pinglab — contents

COBANet

14 May 2026

1. A conductance based neuron equation

The membrane is a capacitor (C_m) pierced by ion channels in parallel. Conservation of charge (Kirchhoff) balances the capacitive current $C_m\,\mathrm{d}V/\,\mathrm{d}t$ against the total ionic current:

$$C_m \frac{\mathrm{d}V}{\mathrm{d}t} = -\sum_{\text{ion}} I_{\text{ion}} \quad (1)$$

Each channel passes an ohmic current — its conductance $g_{\text{ion}} \geq 0$ times the **driving force** $(V - E_{\text{ion}})$, the distance of V from the reversal potential E_{ion} (where the channel's net current vanishes, set by the Nernst equilibrium):

$$I_{\text{ion}} = g_{\text{ion}}(V - E_{\text{ion}}) \quad (2)$$

Because $g_{\text{ion}} \geq 0$, the current's sign lives entirely in the driving force. Summing a leak (g_L, E_L) and synaptic conductances — excitatory (g_e, E_e), inhibitory (g_i, E_i) — gives the general conductance-based (COBA) neuron:

$$C_m \frac{\mathrm{d}V}{\mathrm{d}t} = -g_L(V - E_L) - g_e(V - E_e) - g_i(V - E_i) \quad (3)$$

2. The COBA model

COBANet specialises (3) to two populations — E driven by excitation and inhibition, I by excitation only:

$$C_m^E \frac{\mathrm{d}V^E}{\mathrm{d}t} = -g_L^E(V^E - E_L) - g_e^E(V^E - E_e) - g_i^E(V^E - E_i) \quad (4)$$

$$C_m^I \frac{\mathrm{d}V^I}{\mathrm{d}t} = -g_L^I(V^I - E_L) - g_e^I(V^I - E_e) \quad (5)$$

A neuron spikes at threshold V_{th} and resets to V_{reset} for a refractory period:

$$s_{t+1} = \mathbb{1}[V \geq V_{\text{th}}], \quad V \leftarrow V_{\text{reset}} \text{ if } s_{t+1} = 1 \text{ or refractory} \quad (6)$$

Each synaptic conductance is an exponential trace driven by presynaptic spikes — each spike adds its full weight as an instantaneous jump, then the conductance decays with the channel time constant; there is no E→E connection:

$$\frac{\mathrm{d}g_e^E}{\mathrm{d}t} = -\frac{g_e^E}{\tau_{\text{AMPA}}} + W_{\text{in}} \sum_k \delta\left(t - t_k^{\text{inp}}\right) \quad (7)$$

$$\frac{\mathrm{d}g_i^E}{\mathrm{d}t} = -\frac{g_i^E}{\tau_{\text{GABA}}} + W_{\text{ie}} \sum_k \delta\left(t - t_k^{i}\right) \quad (8)$$

$$\frac{\mathrm{d}g_e^I}{\mathrm{d}t} = -\frac{g_e^I}{\tau_{\text{AMPA}}} + W_{\text{ei}} \sum_k \delta(t - t_k^{e}) \quad (9)$$

(7) is E's excitation from the input W_{in}; (8) its inhibition from I via W_{ie}; (9) the I population's excitation from E via W_{ei}.

3. Discretization

The conductances (7)–(9) and membrane equations (4)–(5) are continuous ODEs. The delta-driven conductances integrate **exactly** over one step: between spikes they decay by $e^{-\Delta t/\tau}$, and any spike landing in the step adds its full weight — the decay-then-add recurrence $g_{t+1} = e^{-\Delta t/\tau} g_t + W s_t$ (with the τ, W and spike train s of each of (7)–(9)). The membrane we integrate by **exponential Euler** — the same algebra for both populations (the I neuron drops g_i).

Collecting on V makes it linear, with total conductance $g_{\text{tot}} = g_L + g_e + g_i$:

$$C_m \frac{\mathrm{d}V}{\mathrm{d}t} = -(g_L + g_e + g_i)V + (g_L E_L + g_e E_e + g_i E_i) \quad (10)$$

Dividing by g_{tot} gives decay-to-steady-state form, naming $\tau_{\text{eff}} = C_m/g_{\text{tot}}$ (shorter than C_m/g_L when synapses are open) and the steady-state voltage V_∞ (the conductance-weighted mean of the reversals):

$$\frac{C_m}{g_{\text{tot}}} \frac{\mathrm{d}V}{\mathrm{d}t} = -\left(V - \frac{g_L E_L + g_e E_e + g_i E_i}{g_{\text{tot}}}\right) \quad (11)$$

A **zero-order hold** freezes the conductances over one step Δt, leaving (11) constant-coefficient with exact solution

$$V_{t+1} = V_\infty + (V_t - V_\infty) e^{-\Delta t/\tau_{\text{eff}}} \quad (12)$$

Per population — I has no g_i, so its g_{tot} and V_∞ drop those terms:

$$g^E_{\text{tot}} = g^E_L + g^E_e + g^E_i, \quad \tau^E_{\text{eff}} = \frac{C^E_m}{g^E_{\text{tot}}}, \quad V^E_\infty = \frac{g^E_L E_L + g^E_e E_e + g^E_i E_i}{g^E_{\text{tot}}} \quad (13)$$

$$g^I_{\text{tot}} = g^I_L + g^I_e, \quad \tau^I_{\text{eff}} = \frac{C^I_m}{g^I_{\text{tot}}}, \quad V^I_\infty = \frac{g^I_L E_L + g^I_e E_e}{g^I_{\text{tot}}} \quad (14)$$

with step (12) for each population $p \in \{E, I\}$: $V^p_{t+1} = V^p_\infty + (V^p_t - V^p_\infty) e^{-\Delta t/\tau^p_{\text{eff}}}$.

Being the *exact* frozen-conductance integral, (12) is **dt-invariant** — N small steps equal one big step, so firing rates and the gamma frequency are physical (Hz) properties, not timestep artifacts (exp044). A forward-Euler step $V_{t+1} = V_t + (\Delta t/C_m) I_{\text{net}(V_t)}$ — not dt-invariant — is kept only as a parity toggle (*COBA_INTEGRATOR*).

Each step runs in fixed order: conductances (7)–(9), then $g_{\text{tot}}, \tau_{\text{eff}}, V_\infty$, then the membrane step (12), then spike + reset (6). **The zero-order hold is this ordering** — the conductances advance once, then stay fixed while the membrane integrates across Δt. E and I advance synchronously, phase-locking the E→I→E gamma cycle to the grid.

Parameters & Units

14 May 2026

All physical quantities in the codebase use the same unit system: **ms for time, mV for voltage, nF for capacitance, μS for conductance, nA for current, Hz for rates**. Time fields carry an explicit *_ms* suffix (*sim_ms*, *ref_ms_E*, *tau_gaba*); CLI flags follow the same convention (*–t-ms 600*). A Δt of 1 means 1 ms, not 1 s.

Quantities

Quantity	Unit	Typical value	Variable
Integration step	ms	0.25	*dt*, *DT_MS*
Simulation length	ms	600	*sim_ms*
Membrane time constant	ms	20 (E), 5 (I)	*tau_m_E*, *tau_m_I*
Refractory period	ms	3 (E), 1.5 (I)	*ref_ms_E*, *ref_ms_I*
AMPA decay	ms	2	*tau_ampa*
GABA decay	ms	9	*tau_gaba*
Resting / leak potential	mV	−65	*E_L*
Spike threshold	mV	−50	*V_th*
Reset potential	mV	−65	*V_reset*
AMPA reversal	mV	0	*E_e*
GABA reversal	mV	−80	*E_i*
Membrane capacitance	nF	1.0 (E), 0.5 (I)	*C_m_E*, *C_m_I*
Leak conductance	μS	0.05 (E), 0.1 (I)	*g_L_E*, *g_L_I*
External drive	μS	0.0006 (async), 0.003 (PING)	*t_e_async*, *t_e_ping*
Input current (CUBA)	nA	20 per spike	*input_scale*
CUBA weight std	nA	32	*W_STD_CUBA*
Max input rate	Hz	25	*max_rate_hz*
Population firing rate	Hz	20–80	*r_E*, *r_I*
Gamma frequency	Hz	30–80	*f_0*

COBA / PING biophysical constants

Used by COBA and PING. Values follow neuroscience conventions (cf. Dayan & Abbott, Gerstner *Neuronal Dynamics*); the E:I asymmetry in $\tau_m, C_m, g_L, \tau_{\text{ref}}$ produces the timescale separation that makes PING dynamics possible.

Parameter	E population	I population
τ_m (ms)	20	5
C_m (nF)	1.0	0.5
g_L (μS)	0.05	0.1
τ_{ref} (ms)	3	1.5
E_L (mV)	−65	−65
V_{th} (mV)	−50	−50

V_{reset} (mV)	-65	-65
E_e (mV, reversal)	0	0
E_i (mV, reversal)	-80	-80

Synapse time constants: $\tau_{\text{AMPA}} = 2$ ms (excitation), $\tau_{\text{GABA}} = 9$ ms (inhibition). These set the ceiling on PING's Δt-stability: once $\Delta t \gtrsim \tau_{\text{GABA}}$ the E→I→E loop cannot complete within one step.

Internal consistency

The chosen units are self-consistent — no conversion factors appear in the integration code. Two equations carry the whole system.

The membrane time constant is $\tau = C/g$. With C in nF and g in µS,

$$\tau_{[\text{ms}]} = \frac{C_{[\text{nF}]}}{g_{[\mu\text{S}]}}$$

so $C_m = 1$ nF and $g_L = 0.05$ µS give $\tau_m = 20$ ms directly.

The LIF voltage update is $\mathrm{d}v = (\Delta t/C)(-g_L(v - E_L) + I)$. With Δt in ms, C in nF, v, E in mV, g in µS, and I in nA,

$$\mathrm{d}v_{[\text{mV}]} = \frac{\Delta t_{[\text{ms}]}}{C_{[\text{nF}]}} \cdot I_{[\text{nA}]}$$

because ms · nA / nF = mV exactly.

Conductance-current products share the same ledger: $g(v - E)$ is µS × mV = nA, so synaptic currents fold into I alongside any direct input current without a scale factor.

Why not SI?

Pure SI (F, S, V, A, s) forces every value to a large negative exponent — $C_m = 10^{-9}$ F, $g_L = 5 \times 10^{-8}$ S, $\Delta t = 2.5 \times 10^{-4}$ s. The neuroscience convention (ms, mV, nF, µS, nA) keeps every typical value between 10^{-3} and 10^{2}, which makes numerical debugging and human intuition faster. The tradeoff is that readers have to trust the unit consistency rather than verify it by plugging into SI formulas — this page is here so that trust is auditable.

Metrics

14 May 2026

This page is the reference for every number and every panel the scope shows. It is written to stand on its own: if this is the first page you read, you should come away understanding what the scope is showing, what each metric measures, and what a "healthy" value looks like.

model

What it is: The model variant driving the run. These sit on a ladder from simplest (snnTorch, a standard feedforward LIF network) to most biophysical (PING, with excitatory and inhibitory populations coupled into a gamma-producing loop). See COBANet for the conductance-based model.

How it's calculated: Read from the *–model* CLI flag. A spinner character prefixes the name on live runs so the eye can tell a frozen frame from one that's still updating.

Common values: snnTorch, *CUBA*, *COBA*, *PING*.

dt

What it is: The integration timestep in ms — how much real time elapses per simulation step. Smaller Δt is more accurate but more expensive (more steps per trial).

How it's calculated: Read from *–dt*. Held fixed across a training run; can be varied at inference to probe Δt-stability.

Common values: 0.1 ms (fine), 0.25 ms (default), 1.0 ms (coarse). PING breaks down above about 1 ms because the excitatory-inhibitory feedback loop can't complete within a single step.

N

What it is: Number of excitatory neurons in the hidden layer. For PING, the inhibitory population is sized at $N_E/4$ (the classical 4:1 cortical ratio).

How it's calculated: Read from *–n-hidden*.

Common values: 256, 512, 1024.

in

What it is: The peak firing rate of the input neurons, in Hz. An MNIST pixel with intensity 1.0 fires at this rate; a dark pixel fires at zero. Controls how strongly the network is driven.

How it's calculated: Each input neuron emits a Bernoulli spike at each step with probability

$$p = r \cdot \Delta t/1000$$

where r is this per-pixel rate (Hz) scaled by pixel intensity. See Training § Input encoding for the full scheme.

Common values: 50 Hz for PING, 100 Hz for simpler models. - in the header when not set.

E

What it is: Firing rate of the excitatory population, averaged over all E neurons and over the trial, in Hz. The single most important live readout of network activity.

How it's calculated: Let $s_E \in \{0,1\}^{T \times N_E}$ be the binary E spike array (T timesteps, N_E neurons; entry is 1 if neuron i spiked at step t), and $T_{\text{sec}} = T\Delta t/1000$ the trial length in seconds. Then

$$r_E = \frac{\sum_{t,i} s_{E[t,i]}}{N_E \cdot T_{\text{sec}}}$$

Common values: 1–30 Hz is sparse-coding territory (healthy for snnTorch when classifying MNIST); 40–80 Hz is the gamma-locked regime (PING target); over $\sim$ 150 Hz means the network has saturated; 0 Hz means it's silent.

I

What it is: Firing rate of the inhibitory population, same definition as *E*. Only meaningful for models that have an I population (COBA, PING); shows - otherwise.

How it's calculated: Same as *E*, on the I spike array s_I and with N_I inhibitory neurons:

$$r_I = \frac{\sum_{t,i} s_{I[t,i]}}{N_I \cdot T_{\text{sec}}}$$

Common values: Typically 2–4× the E rate under balanced PING operation — roughly 80–200 Hz.

f_0

What it is: The dominant oscillation frequency of the E population, in Hz. If the network is gamma-locked the rate signal has a clear peak in its Fourier spectrum; f_0 is the frequency of that peak.

How it's calculated: The E spike raster is binned at 2 ms, the resulting time series $r_E(t)$ is mean-subtracted and FFTed. A peak search runs in the 5–80 Hz band and requires the peak to be at least 3× the band median (an SNR gate). A subharmonic check follows: if there is clear power at half the peak frequency ($\geq$ 30 % of the peak's power), f_0 is set to that lower frequency instead — this catches the case where pyramidal neurons fire every second cycle.

Common values: 0 Hz (non-oscillating; always for non-PING models, which short-circuit the calculation); 30–80 Hz (gamma, the PING target); anything below 30 Hz is sub-gamma beta. Shows - in the header when 0.

CV

What it is: How bursty the E population's firing is. Takes the total E spike count in each 2 ms bin and computes the coefficient of variation (std / mean) of those counts across bins.

How it's calculated: Bin s_E at 2 ms; let n_b be the total E spike count in bin b. Then

$$\text{CV} = \frac{\sigma(n_b)}{\max(\mu(n_b), 10^{-9})}$$

Common values: Around 1 for Poisson-like asynchronous firing; well above 1 (often $\gtrsim 2$) for gamma-locked PING, where spikes cluster into cycle peaks and leave the troughs empty; below $\sim$ 0.3 for a clock-like or saturated network.

I/E

What it is: The ratio of inhibitory to excitatory population rates. A single number that summarises E–I balance.

How it's calculated:

$$\mathrm{I/E} = \frac{r_I}{\max(r_E, 10^{-9})}$$

Common values: 2–4 under balanced PING operation. Much higher means inhibition has run away; much lower means inhibition is failing or E has saturated.

act

What it is: The fraction of excitatory neurons that spiked at least once during the trial. Complements the population rate: a high E could come from a few neurons firing a lot, or many neurons firing a little — *act* tells you which.

How it's calculated:

$$\mathrm{act} = \frac{\#\left\{i : \sum_t s_{E[t,i]} > 0\right\}}{N_E}$$

Common values: 1–3 % is healthy sparse coding (a classical snnTorch regime). 80 %+ with CV near 0.4 is healthy gamma-locked PING. Persistent values below 1 % or above 95 % alongside stalled accuracy are pathological.

e_raster

What it is: The excitatory spike raster over the trial. Each dot is one spike; the y-axis is E-neuron index, the x-axis is time. This is the primary dynamical read and occupies the largest cell in the grid.

How to read it: Vertical bands (near-synchronous firing across the population, at regular intervals) mean gamma-locked PING. A uniformly-speckled cloud means asynchronous firing — normal for CUBA or snnTorch. An empty raster means the network is silent; a solid block means it is saturated.

i_raster

What it is: Same idea as *e_raster* but for the inhibitory population. Empty for models without an I population.

How to read it: Under balanced PING the I raster is denser and at higher rate than the E raster, matching the 2–4× I/E ratio.

drive

What it is: The mean input current (conductance) reaching the membrane, overlaid with the voltage thresholds a neuron needs to reach to spike.

How it's calculated: The input conductance is passed through an exponential synapse with time constant τ_{AMPA} to get the steady-state drive

$$g_{\mathrm{ss}} = \frac{\bar{g}}{1 - e^{-\Delta t/\tau_{\mathrm{AMPA}}}}$$

The bare spike threshold — the drive a neuron needs in the absence of inhibition — is

$$g_{\text{thresh}} = \frac{g_L(V_{\text{th}} - E_L)}{E_e - V_{\text{th}}}$$

The effective threshold adds an inhibition-proportional penalty $g_i \cdot |E_i - V_{\text{th}}| \,/(E_e - V_{\text{th}})$.

How to read it: The shaded region is where drive exceeds the effective threshold — that's when E neurons can fire. If drive never crosses threshold the network is dead by construction; if drive is always above, it is saturated.

weights

What it is: Histograms of the current weight values, one sub-histogram per weight matrix (input-to-hidden, hidden-to-output, and any recurrent E–E / E–I / I–E matrices).

How to read it: Signed weights (standard SNN) look roughly Gaussian around zero. Dale's-law weights are non-negative so they form a half-normal clamped at zero. A long tail means a few synapses dominate; a narrow stack at zero means many weights have died under regularisation.

output

What it is: The readout logits for each output class, over the course of the trial.

How it's calculated: The readout accumulates hidden spikes across time and projects them through a trained linear layer:

$$\hat{y}_t = \left(\sum_{s \le t} s_s^{\text{hid}} \right) W_{\text{out}} + b_{\text{out}}$$

The argmax at the final step is the model's prediction. See Training § Readout.

How to read it: For MNIST there are ten traces; if the network has learned, one trace pulls away from the pack as the trial progresses. If all traces stay flat, or one dominates from t=0, the run is pathological.

psd

What it is: The power spectral density (frequency content) of the E population rate $r_E(t)$. The gamma band (30–80 Hz) is shaded.

How it's calculated: $r_E(t)$ is computed in 2 ms bins over the stimulus window, mean-centred, FFTed, and normalised:

$$\text{PSD}(f) = \frac{|\text{FFT}(r_E - \bar{r}_E)(f)|^2}{\max_f |\text{FFT}(r_E - \bar{r}_E)(f)|^2}$$

How to read it: A flat PSD means asynchronous firing. A peak inside the shaded band means gamma. A peak below 30 Hz means sub-gamma (beta). A peak above 80 Hz is rejected by the f_0 search.

participation

What it is: The distribution of per-neuron spike counts across the E population for this trial.

How to read it: A stack at zero indicates dead neurons; a stack at the per-neuron ceiling indicates saturated neurons. Both can be invisible from the population-level E token, which averages across the group.

acc_curve

What it is: Test accuracy, training loss, and learning rate as a function of epoch — the standard training-progress panel. All three are normalised to their running maximum and plotted on a single 0–100 % axis. Rendered only during training runs.

How to read it: A healthy run has accuracy climbing from ~ 10 % (chance on MNIST) toward 85–98 %, and loss dropping monotonically. A flat accuracy with drifting loss usually means the readout can't separate classes.

rate_curve

What it is: The E and I firing rates as a function of training epoch. Training-only.

How to read it: Stable rates within the healthy regime for the model (see *act* above) means the network is holding its operating point as it learns. Monotonic drift toward 0 or toward a ceiling usually means gradient pressure is pushing the network out of its regime.

grad_flow

What it is: For each weight matrix, the ratio of gradient norm to weight norm — how hard each layer is being updated relative to its current size. Training-only; log y-axis.

How to read it: The shaded band from 10^{-3} to 10^{-2} is the healthy window. Values below mean gradients are vanishing (that layer is not learning); above means they are exploding.

digit_image

What it is: The MNIST digit currently being presented to the network. A small greyscale inset that lets you sync the readout in *output* to the actual input class.

sweep

What it is: A generic progress indicator for parameter-sweep runs. Shows the full sweep range faintly and the completed prefix solidly, with a dot on the current value.

sweep_rates

What it is: E and I firing rates accumulated across an inference sweep — one point per sweep frame, updated as the sweep progresses. Sweep-only.

How to read it: If the trace is flat across the swept variable, the network is stable under that variable. If it fans out by one to two orders of magnitude, it is not — this is the signature read off in a Δt-stability experiment.

sweep_f0

What it is: Peak oscillation frequency f_0 across the sweep, with the gamma band (30–80 Hz) shaded. Sweep-only.

How to read it: A PING run with a robust E→I→E loop holds f_0 inside the shaded band across the whole sweep. Drift out of band indicates the loop is failing at that parameter value.

Spectral radius

What it is: The largest-magnitude eigenvalue of the recurrent weight matrix J. A linear stability indicator: if $\rho(J) > 1$, the linearised network would explode; $\rho(J) \approx 1$ sits at the edge of chaos.

How it's calculated: The recurrent matrix is reassembled post-training with E columns kept positive and I columns flipped to negative (so Dale's law appears as signs), then $\rho(J) = \max_i |\lambda_i|$ via an eigendecomposition.

Common values: $\rho < 1$ is quiescent (no sustained oscillation); $\rho \approx 1$ is the gamma operating regime; $\rho > 1$ is linearly unstable but the spiking nonlinearity can still stabilise it.

Linear f_0 prediction

What it is: A continuous-rate prediction for the oscillation frequency, derived from the eigenvalue of J with the largest imaginary part.

How it's calculated: Pick $\lambda^* = a + bi$ with largest $|b|$, then

$$f_0^{\text{lin}} = \frac{|b|}{2\pi} \cdot 1000 \text{ Hz}$$

Common values: Typically an upper bound on the measured f_0 — the linear mode ignores the spiking nonlinearity and the refractory floor.

Refractory-floor prediction

What it is: A lower bound on the oscillation frequency set by the biology: no spiking network can cycle faster than one excitatory refractory period plus one GABA-synapse decay.

How it's calculated:

$$f_0^{\text{cor}} = \frac{1000}{\tau_{\text{ref}}^E + \tau_{\text{GABA}}} \text{ Hz}$$

Common values: The measured f_0 typically lies between f_0^{cor} and f_0^{lin}.

loss

What it is: The training-set cross-entropy at the end of each epoch — the scalar the optimiser is minimising.

Common values: Starts near $\ln(10) \approx 2.3$ for chance-level classification on MNIST and drops monotonically under healthy training. A flat or climbing loss means the gradient isn't reaching the readout.

test_loss

What it is: Cross-entropy on the held-out test set, evaluated once per epoch.

Common values: Tracks *loss* with a small gap. A widening gap usually means overfitting; a gap that collapses to zero often means data leakage between train and test.

lr

What it is: The current learning rate — the multiplier on each gradient step. Varies if *adaptive-lr* is set; constant otherwise.

Common values: 10^{-2} for snnTorch / CUBA; 10^{-4} for COBA / PING, whose conductance-based gradients are much steeper (see Gradient Stabilisation).

grad_norm

What it is: The mean gradient norm across all parameters, taken after gradient clipping.

Common values: Stable to within an order of magnitude across a run. A sudden jump usually precedes an NaN loss.

grad_ratios

What it is: A dictionary mapping each weight matrix to its $\|\nabla\| / \|W\|$ — the quantity plotted by *grad_flow*.

Common values: Healthy band 10^{-3} to 10^{-2} per layer; below is vanishing, above is exploding.

acc

What it is: Test-set accuracy (%) at the end of each epoch — the canonical "how well is the network classifying" number.

Common values: 10 % is chance on MNIST; 85–90 % is typical at a small training tier; 95–98 % at medium or larger; anything above 98 % flags a leaked test split or a bug.

best_acc

What it is: The maximum of *acc* over all epochs seen so far. Tracked alongside *best_epoch* (the epoch that set it) and a *new_best* flag (true on the epoch that set it).

Common values: This is the headline number cited in notebook entries. If *best_epoch* is near the final epoch the run was still improving; if much earlier, it had plateaued.

On inference runs, *best_acc* is just the single-shot test accuracy; the raw count *n_correct* is saved alongside, and per-sample calls go to *test_predictions.json* (one row per sample, carrying *idx*, *true*, *pred*, *correct*, *logits* for confusion-matrix and calibration downstream).

Training

14 May 2026

This is the shared training recipe every model on the ladder uses. It runs, in order, through how gradients flow through time, the surrogate that lets them pass through spikes, the one flag that stops them exploding (its own article, Gradient Stabilisation), the loss and optimiser, the readout options, the firing-rate regulariser, weight initialisation, and the tasks the networks are trained on.

Backpropagation through time

Every model here is a recurrent system run forward in time, so gradients come from Backpropagation Through Time (BPTT): unroll the recurrence into a deep feedforward graph — one layer per timestep, all sharing the same weights — and backpropagate through it.

Take a hidden state h^t that evolves as

$$h^t = f(h^{t-1}, x^t; \theta), \quad y^t = g(h^t; \theta)$$

with input x^t, output y^t, and parameters θ shared across time. Running T steps gives a chain $h^0 \to h^1 \to \cdots \to h^T$, which for gradients we treat as a depth-T feedforward network with tied weights.

The gradient with respect to a parameter θ_k then sums over every timestep it touched:

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial h^t} \cdot \frac{\partial h^t}{\partial \theta_k}$$

The catch is the product of per-step Jacobians $\partial h^{t+1}/\partial h^t$ running through the chain: if their norms sit above 1 the product explodes in T, if below 1 it vanishes.

SNNs fit BPTT naturally — one simulation step is one step of the recursion, with the hidden state holding membrane potentials, synaptic conductances, and refractory counters. A 200 ms trial at $\Delta t = 0.1$ ms unrolls to $T = 2000$ steps. Because the state variables carry physical units (mV, µS), the per-step Jacobians are wildly scaled: voltage updates carry tiny factors like $\Delta t / C_m$ while surrogate gradients through spikes are $O(1)$. That mismatch is exactly what the gradient-stabilisation flag exists to fix — derived in full in Gradient Stabilisation.

Surrogate gradients

The spike function $S = \mathbf{1}[U \geq \theta]$ has zero gradient almost everywhere, so the backward pass substitutes a smooth surrogate. Pinglab uses the fast-sigmoid surrogate everywhere. Forward is the hard step; backward is

$$\frac{\partial \tilde{S}}{\partial U} = \frac{k}{(1 + k\,|U - \theta|)^2}$$

This matches snntorch's *FastSigmoid*, so equal-k comparisons against the snntorch reference test the update rule, not the surrogate.

It takes its slope from *SURROGATE_SLOPE = 5.0*, overridable per-run with *–surrogate-slope*.

Gradient stabilisation

Conductance-based networks (COBA, PING) need one extra ingredient to train: the recurrent E↔I loop makes the backpropagated gradient explode during BPTT, and a single flag, `--v-grad-dampen`, tames it. The full derivation — why the gradient diverges once per gamma cycle, and why per-step voltage damping fixes it without touching the forward pass — has its own article: **Gradient Stabilisation**.

The training loop

Logits from the readout go into cross-entropy loss:

$$L_{\text{CE}} = -\frac{1}{B}\sum_{b=1}^{B} \log \frac{\exp\left(\hat{y}_{b,c_b}\right)}{\sum_k \exp\left(\hat{y}_{b,k}\right)}$$

with batch size B, logit vector $\hat{y}_b$, and true class c_b; chance-level loss on a 10-class problem is $\ln 10 \approx 2.30$. The optimiser is Adam, with gradients clipped to unit norm (`GRAD_CLIP = 1.0`) before each step. The saved `weights.pth` is the *best-epoch* state by test accuracy, not the final epoch.

Readout

The readout collapses the last hidden layer's activity into class logits; `--readout` picks how. Four modes:

- `spike-count` — sum last-hidden spikes over the trial and project linearly, $\hat{y} = \left(\sum_t s_t^{\text{hid}}\right) W_{\text{out}} + b_{\text{out}}$. Equivalent to spike-rate up to a constant.
- `mem-mean` — pass spikes through a final non-resetting LIF and average its membrane potential over the trial. Default for the COBA/PING recipes; used by exp025 and the streaming entries.
- `li` — leaky integrator: a non-spiking LIF whose final-step membrane potential is the logit.
- `rate` — softmax over per-trial spike rates.

The choice matters because it sets where the gradient enters the network: `mem-mean` lets it flow through the output LIF's membrane at every timestep, while `spike-count` only sees the aggregate.

Firing-rate regularisation

Many recipes penalise too much or too little hidden firing via `--fr-reg-upper-theta`, `--fr-reg-upper-strength`, and the matching `lower` pair:

$$\mathcal{L}_{\text{fr}} = s_u \cdot \text{ReLU}(\bar{r} - \theta_u) + s_l \cdot \text{ReLU}(\theta_l - \bar{r})$$

where $\bar{r}$ is the per-layer mean firing rate (per-neuron or population, set by `--fr-reg-mode`). This is the mechanism behind the θ_u sweeps in exp025 and the rate-floor framing in ar009.

Weight init

Feedforward weights are sampled fan-in-normalised, either half-normal (Dale's law) or normal (signed):

$$W \sim \mathcal{N}(\mu, \sigma^2), \quad W \leftarrow W / N_{\text{pre}}$$

with optional sparsity $s \in [0, 1)$: a fraction s of entries are zeroed and the survivors rescaled by $1/(1 - s)$, so the expected synaptic input per post-neuron is preserved.

Dale's law under Adam

When Dale's law is on, the feedforward matrices W_{ff} are clamped to $W \geq 0$ when they are read by the forward pass and every trainable constrained matrix is projected back into the non-negative cone by `project_dales()` after each optimiser step. The recurrent conductance matrices W_{ee}, W_{ei}, W_{ie}, and W_{ii} are not forward-clamped: they are initialised non-negative and, when trainable, kept non-negative by the post-step projection. Their entries are conductance magnitudes; pathway-specific reversal potentials, rather than a negative stored W_{ie}, determine whether a synapse is excitatory or inhibitory.

Bayesian Literature Review

19 May 2026

Paper	Year
Ma, Beck, Latham & Pouget — *Bayesian Inference with Probabilistic Population Codes*	2006
Pouget, Dayan & Zemel — *Inference and Computation with Population Codes*	2003
Fiser, Berkes, Orbán & Lengyel — *Statistically Optimal Perception and Learning: From Behavior to Neural Representations*	2010
Orbán, Berkes, Fiser & Lengyel — *Neural Variability and Sampling-Based Probabilistic Representations in the Visual Cortex*	2016
Aitchison & Lengyel — *The Hamiltonian Brain: Efficient Probabilistic Inference with Excitation-Inhibition Networks*	2016
Echeveste, Aitchison, Hennequin & Lengyel — *Cortical-like Dynamics in Recurrent Circuits Optimized for Sampling-Based Probabilistic Inference*	2020
Padamsey, Katsanevaki, Dupuy & Rochefort — *Neocortex Saves Energy by Reducing Coding Precision during Food Scarcity*	2022

Terms to know first

The papers above share a small vocabulary that runs across all of them. If a term feels hazy, look it up before reading — they are not redefined in each paper.

Bayesian fundamentals.

- *Prior, likelihood, posterior.* $p(\theta)$, $p(x \mid \theta)$, $p(\theta \mid x) \propto p(x \mid \theta)p(\theta)$. The posterior is what every paper here claims the brain represents in some form.
- *Generative model.* The forward model the brain is assumed to have inverted: latent causes $\rightarrow$ sensory observations.
- *Recognition / inference.* The inverse problem — recovering $p(\theta \mid x)$ from x — performed approximately by the circuit.
- *Marginalisation.* Integrating out nuisance variables. Cortical circuits are claimed to do this in different ways depending on whether the representation is parametric or sample-based.

Two flavours of neural representation of probability. This is the central split in the reading list and the two camps disagree on it.

- *Parametric (probabilistic population code, PPC).* The instantaneous population firing rate $\boldsymbol{r}$ *encodes the parameters* of a distribution over θ — for an exponential-family code, $p(\theta \mid x) \propto \exp(\boldsymbol{h}(\theta)^\top \boldsymbol{r})$. Posterior uncertainty is read off the population gain. Ma et al. 2006 and Pouget et al. 2003 are the canonical PPC papers.
- *Sampling-based.* The instantaneous activity $\boldsymbol{r}(t)$ *is* a sample from the posterior, $\boldsymbol{r}(t) \sim p(\theta \mid x)$. Uncertainty is read off the trial-to-trial variability over time. Fiser et al. 2010, Orbán et al. 2016, and the Lengyel-group papers (Aitchison & Lengyel 2016, Echeveste et al. 2020) are the canonical sampling-camp papers.
- *Variational inference (background).* A third flavour — fit a parametric $q(\theta)$ to the posterior by minimising KL divergence. Worth knowing as the ML reference point, though none of these papers commit to it as the brain's algorithm.

Population-code mechanics.

- *Tuning curve* $f_i(\theta)$ — neuron i's mean rate as a function of stimulus θ.
- *Fisher information* $I_F(\theta)$ — the precision a code carries about θ; for an unbiased estimator, $\mathrm{Var}\left(\hat{\theta}\right) \geq 1/I_F(\theta)$ (Cramér–Rao). Often discussed per spike, to compare codes at matched energy.
- *Linear Fisher information.* The piece of I_F accessible by a linear readout — what a downstream neuron with synaptic weights can actually extract.
- *Noise correlations* c_{ij} — trial-to-trial covariance between neurons at fixed stimulus. Information-limiting correlations are the slice of c_{ij} aligned with the tuning gradient $\partial f/\partial\theta$; they cap I_F no matter how many neurons you average.
- *Poisson variability.* Spike-count variance ≈ mean. The baseline noise model PPCs are built on.

Sampling-camp specifics.

- *Langevin dynamics.* Stochastic gradient on $-\log p(\theta \mid x)$ plus white noise — the simplest dynamical system whose stationary distribution is the posterior. The recurrent-circuit papers are versions of this idea with biological constraints.
- *Hamiltonian Monte Carlo (HMC).* Sampling with momentum — propose moves along trajectories of a fictitious Hamiltonian, accept by Metropolis. Aitchison & Lengyel argue E/I circuits implement an HMC-like sampler with the inhibitory population playing the momentum role.
- *Inhibition-stabilised network (ISN).* A recurrent E/I circuit whose excitatory subnetwork is *unstable* on its own and stabilised only by inhibition. Empirically common in cortex; the sampler papers exploit ISN dynamics for fast, well-mixed sampling.
- *Mixing time, autocorrelation time.* How quickly a sampler decorrelates from its current state — sets how fast a circuit can revise its uncertainty estimate. Echeveste et al. argue cortical-like transients arise from optimising for fast mixing.
- *Stochastic stability vs. transients.* A sampler must be globally stable but locally fluctuating. The non-trivial transient responses cortical circuits show are read as evidence of a sampler near, not at, equilibrium.

Coding-cost and metabolism.

- *Bits per spike.* Mutual information per spike between stimulus and response — the efficiency the energetics papers ask about.
- *Coding precision.* The width of the represented posterior; can be widened to save metabolic cost without changing the represented mean. Padamsey et al. 2022 show this happens *in vivo* under food scarcity — directly tying the Bayesian and metabolic stories together.
- *Metabolic cost of a spike.* Sodium-pump ATP per action potential; the currency the precision–cost trade-off is denominated in.

Useful background that none of these papers stop to define.

- *Exponential family, sufficient statistics, natural parameters.* PPCs lean on this — the firing rates are the natural parameters of an exponential-family posterior.

- *KL divergence.* The asymmetric distance between distributions, used to score how good an approximate posterior is.
- *Stationary distribution of a Markov chain.* The distribution the chain converges to and samples from at long times. The sampling-camp claim is that the cortical chain's stationary distribution is the posterior.

Manuscript

21 June 2026

Abstract

Gamma oscillations are widespread in cortical activity but are largely absent from trained spiking neural networks, which typically operate in a current-based regime or impose oscillations as an external input. We train a spiking network with a fixed pyramidal–interneuron gamma (PING) loop on MNIST under surrogate-gradient descent, with the recurrent E↔I weights held at biophysical values. At matched test accuracy on the accuracy–rate frontier, the post-training excitatory firing rate is roughly an order of magnitude below a conductance-based control (7-fold by total population spike count once the higher-rate inhibitory pool is included), and the trained rate is well described by an affine relation $r_E \approx 1.15 + 0.183 f_\gamma$ with the measured gamma frequency ($R^2 = 0.994$). When the loop weights are released for training under a Dale's-law clamp, the rhythmicity collapses within one epoch and is not recovered from any initial condition tested. The trained network classifies a continuously concatenated digit stream without retraining or a segmentation cue; streaming accuracy is approximately governed by the product of presentation duration and input rate, remains below $\approx 80\%$ for durations of 15 ms or less, and at 200 ms rises from chance below 0.5 Hz to clearly informative performance by 2 Hz. These results are consistent with an interpretation of gamma as a structural constraint on excitatory firing rates that does not require learned tuning of the inhibitory connectivity.

1. Introduction

Gamma oscillations in the 30–80 Hz band have been associated with attention, binding, and gating in cortical activity [**1, 2, 3**], with the original visual-cortex observation reported by [**4**]. Two generation mechanisms are commonly distinguished, ING and PING [**5, 6**] the present work focuses on PING. PING arises from the dynamics of a recurrent excitatory–inhibitory (E↔I) loop [**7**]. Optogenetic activation of fast-spiking interneurons in intact cortical circuits drives gamma rhythms [**8, 9, 10, 11**]. Earlier in vitro recordings [**12**] and biophysical models [**13**] characterised interneuron-driven gamma in isolated inhibitory networks, and the synaptic mechanisms that synchronise the interneuron pool have been described [**14**].

PING has been studied extensively in biophysical [**15, 16, 17**] and neural-mass / mean-field models [**18, 19, 20, 21**], but these models are descriptive: they are not trained on a task. The parallel literature on trainable spiking neural networks uses surrogate-gradient descent for end-to-end optimisation [**22, 23, 24**] the resulting networks are typically current-based and non-rhythmic. The rhythmic variants either impose the oscillation as an external input [**25**] or obtain it as an emergent property of unconstrained surrogate-gradient training [**26**] in neither case is the rhythm carried by a fixed, biophysically-calibrated PING loop, and the present work is to our knowledge the first task-trained spiking network of that kind.

Cortical pyramidal cells fire at low rates (typically below 10 Hz) under strong recurrent input [**27**], and the cortical metabolic budget is dominated by excitatory spike generation, with inhibitory spikes substantially less costly per spike [**28, 29**]. The mechanism that constrains pyramidal firing rates under these conditions is not fully understood. We test the hypothesis that a fixed E↔I loop, by generating a gamma rhythm, constrains the post-training

excitatory firing rate. A spiking network with a fixed PING loop is trained on MNIST and the post-training firing rate is compared with a non-rhythmic baseline at matched accuracy. In the architecture studied here the measured rhythm and firing rate are tightly linked, so rate and timing are synergistic descriptions of a single dynamics rather than independent codes [**30**].

The remainder of the paper is organised as follows. §2.1 describes the model; §2.2 characterises the gamma onset in theory and simulation; §2.3 reports the trained accuracy–rate frontier; §2.4 reports two experiments that test whether the firing-rate reduction is acquired during training; §2.5 reports the relationship between the gamma frequency and the post-training firing rate; §2.6 reports robustness of the post-training firing rate to spike perturbations and to integration timestep; and §2.7 reports a streaming-classification protocol.

2. Results

2.1 The model: COBA baseline and the PING loop

The network is a two-population conductance-based spiking model with a single hidden layer of N_E excitatory (E) and N_I inhibitory (I) leaky integrate-and-fire (LIF) units, driven by feedforward input weights W_{in} and read out by a non-spiking leaky-integrator layer over the excitatory population (§5.2). Recurrence is confined to one excitatory–inhibitory (E↔I) loop: E projects to I through the weight W^{EI} and I back to E through W^{IE}, with no E→E or I→I connection. Throughout the paper we compare two configurations of this single architecture: with the loop disabled ($W^{EI} = W^{IE} = 0$) it is a conductance-based (COBA) control in which input drives the excitatory population alone, and with the loop engaged it is the pyramidal–interneuron gamma (PING) configuration. The same network is trained on MNIST by surrogate-gradient descent from §2.3 onward (§5.4); in this section we characterise its free-running dynamics.

Free-running, the COBA configuration fires asynchronously: its power spectral density (PSD) has no peak in the gamma band, and the excitatory firing-rate–current (f–I) curve rises to ≈ 485 Hz under the strongest drive tested. Engaging the E→I→E loop instead produces synchronous inhibitory bursts and gamma-banded excitatory rasters with a PSD peak at $f_\gamma \approx 42$ Hz, and holds the excitatory firing rate approximately an order of magnitude below the COBA baseline across two decades of input drive (Figure 1).

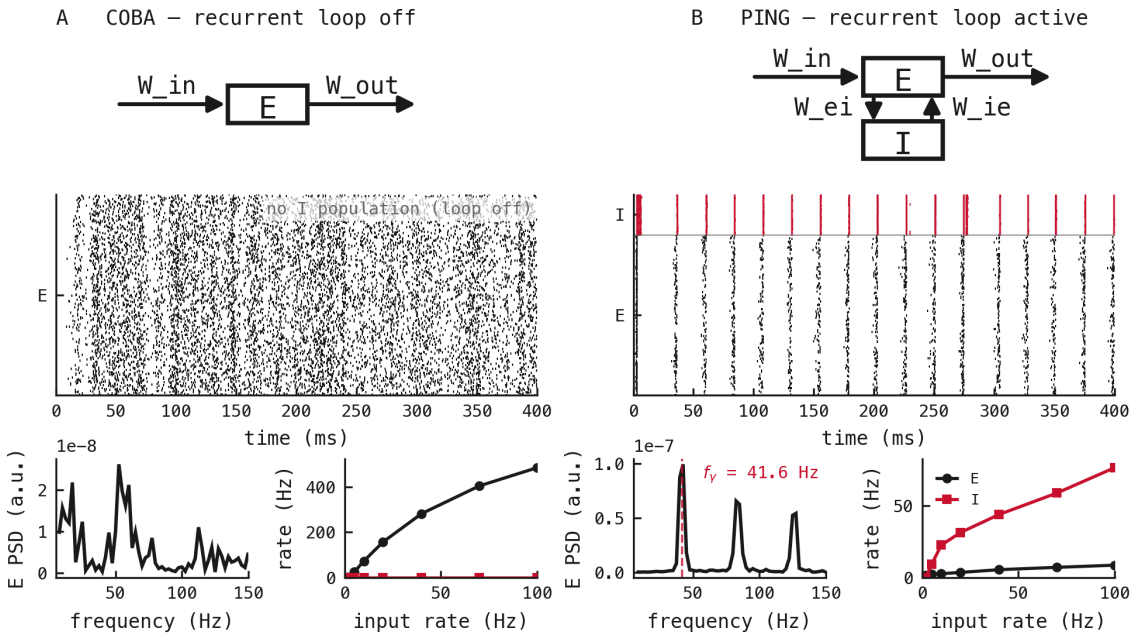


Figure 1: **A single recurrent E→I→E loop simultaneously generates a gamma rhythm and clamps the excitatory firing rate.** Free-running activity of the two-population conductance-based network (excitatory pool $N_E = 1024$, inhibitory pool $N_I = 256$; canonical biophysical parameters, §5.1) under matched Poisson drive, in two configurations. Each column shows, from top: a wiring schematic, a single-trial spike raster, the excitatory power spectral density (PSD), and the excitatory f–I curve. **(A)** COBA baseline with the recurrent loop disabled ($W^{EI} = W^{IE} = 0$): input projects to the excitatory (E) population only, with no inhibitory (I) population (schematic). The E spike raster is asynchronous, the Welch PSD of the summed E population shows no gamma-band peak, and the excitatory f–I curve rises to ≈ 485 Hz under the strongest drive tested. **(B)** PING configuration with the loop engaged (schematic: E→I via W^{EI}, I→E via W^{IE}; no I→I or E→E synapse): the inhibitory (I) population fires synchronous bursts, the E raster forms gamma bands, the PSD shows a discrete peak at $f_\gamma \approx 42$ Hz, and on axes shared with (A) the E rate is held approximately an order of magnitude lower across two decades of input drive while the I rate rises to ≈ 76 Hz. Source: exp023.

2.2 Gamma onset across the $W^{EI} \times W^{IE}$ plane

Across the $W^{EI} \times W^{IE}$ coupling plane the excitatory firing rate decreases, the inhibitory firing rate increases, and the lobe–trough rhythmicity score R (a normalised, dimensionless measure of periodicity in the population autocorrelation, bounded in $[0, 1)$; §5.5) increases monotonically with coupling strength: $R \approx 0$ along the COBA edges and $R \approx 0.98$ at strong coupling. The four-dimensional mean-field reduction (§5.3) predicts a supercritical Hopf bifurcation at external drive $I^\star_{\text{ext}} = 0.6$ nA with crossing frequency $f^\star \approx 27.8$ Hz. The classification as supercritical is supported by quasi-static up/down ramps with peak hysteresis below 10^{-5} in rate units and by the linear scaling of the squared steady-state oscillation amplitude A, $A^2 \propto (I_{\text{ext}} - I^\star_{\text{ext}})$ ($R^2 = 0.999$). The predicted gamma frequency is in qualitative agreement with the spiking measurement across the GABA synaptic decay time constant $\tau_{\text{GABA}} \in [4.5, 27]$ ms (Figure 2).

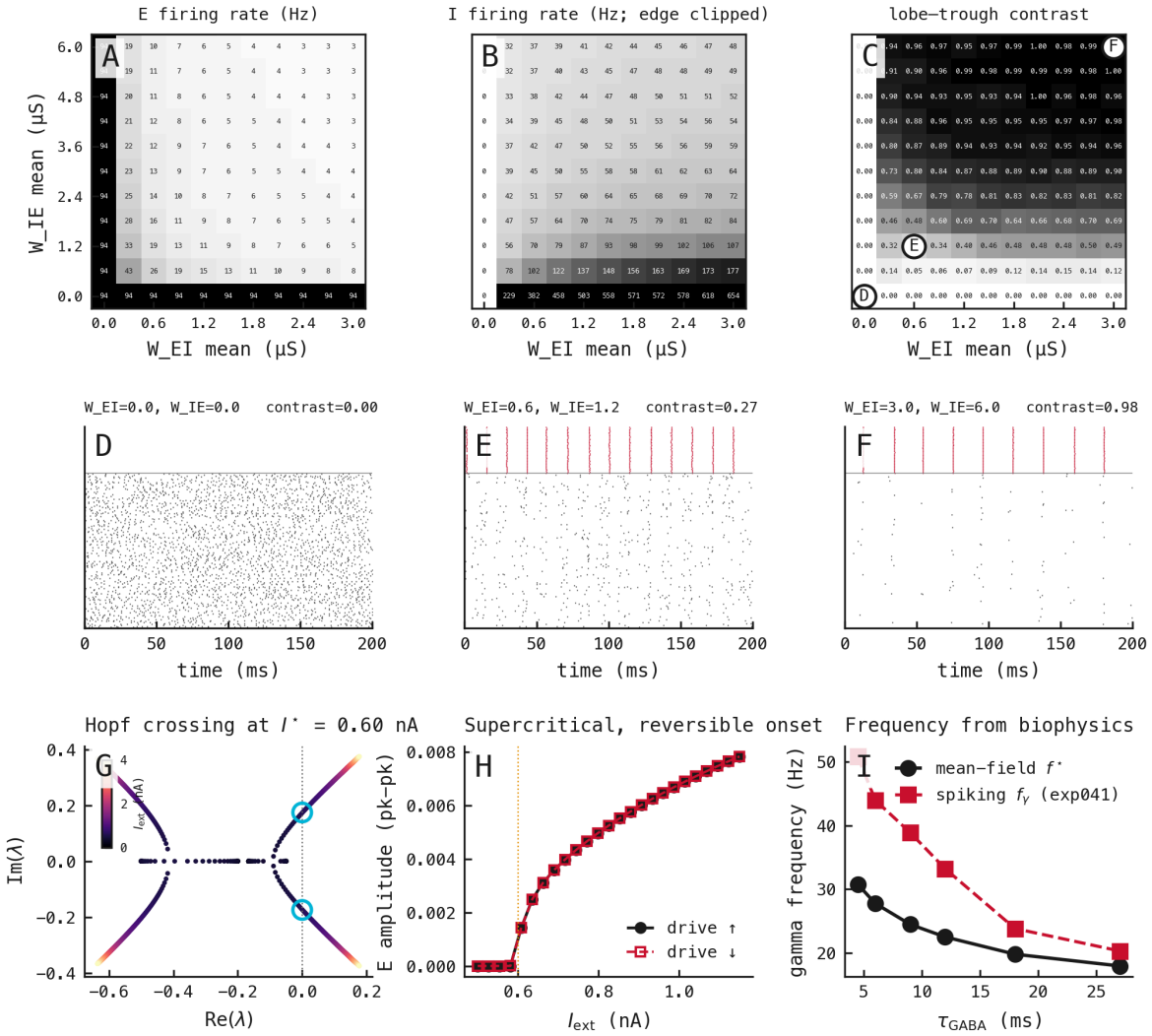


Figure 2: **Gamma emerges through a smooth, reversible onset across the coupling plane, consistent with a supercritical Hopf bifurcation of the mean-field reduction.** Nine panels (A–I). **(A–C)** Steady-state measurements across the 11×11 $W^{EI} \times W^{IE}$ coupling plane: mean excitatory firing rate (A), mean inhibitory firing rate (B), and the lobe–trough rhythmicity contrast R (C, §5.5), which is ≈ 0 along the two COBA edges and rises to ≈ 0.98 toward strong coupling. **(D–F)** Single-trial E (black) and I (red) rasters at three points sampled along the $W^{IE} = 2W^{EI}$ diagonal (circled in C): the loop-disabled origin (D, asynchronous), weak coupling ($R < 0.5$, E), and strong coupling (F, sharp gamma volleys). **(G–I)** The four-dimensional conductance mean-field reduction (§5.3): a complex-conjugate eigenvalue pair crosses into the right half-plane at external drive $I^\star = 0.6$ nA (G), locating a Hopf bifurcation at $f^\star \approx 27.8$ Hz; the steady-state oscillation amplitude grows continuously across the onset with coincident up- and down-ramp branches (H), the signature of a supercritical, reversible transition; and the predicted gamma frequency falls with τ_{GABA} in qualitative agreement with the spiking measurement (I). Source: exp054 (coupling-plane maps and mean-field, incorporating exp033); spiking f_γ from exp041.

2.3 Trained PING attains COBA accuracy at $\approx 10\times$ fewer spikes

Both architectures, trained on MNIST under surrogate-gradient descent (§5.4), converge to approximately 91% test accuracy. Sweeping the spike-budget penalty θ_u generates an

accuracy–rate frontier; at every value of θ_u tested, PING attains higher accuracy at lower mean hidden-E firing rate than COBA. At the operating point with θ_u disabled, PING reaches $\approx 91\%$ accuracy at ≈ 12.3 Hz mean hidden-E rate, against $\approx 91\%$ at ≈ 181 Hz for COBA (Figure 3). The operating-point endpoints are reported as means over three seeds (42, 43, 44) and are approximately seed-invariant; interior points of the θ_u sweep that trace the frontier are run at a single seed (42). Seed-invariance of the post-training firing rate at finer resolution is established in §2.5–§2.7, where three seeds per condition are reported throughout. The PING rate does not decrease further when θ_u is lowered, consistent with a structural lower bound on the rate.

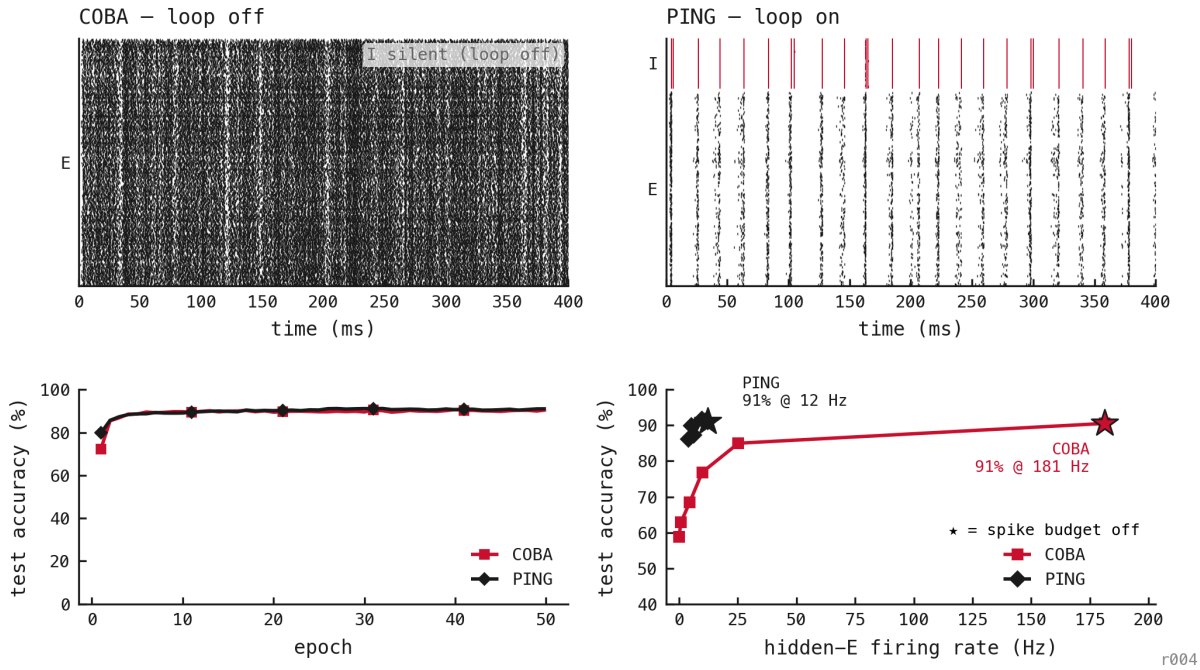


Figure 3: **Trained PING matches COBA classification accuracy while operating at an order-of-magnitude lower excitatory firing rate.** Both configurations were trained on MNIST by surrogate-gradient descent (§5.4). Top: representative single-trial rasters of the trained networks: COBA fires densely and asynchronously with the inhibitory population silent, whereas PING fires in gamma bands with synchronous inhibitory bursts (red) above excitatory spikes (black). For visualisation, each raster is an extended 400 ms replay of one digit, twice the 200 ms presentation used for training and quantitative evaluation. Bottom left: test accuracy per epoch, both configurations converging to $\approx 91\%$. Bottom right: accuracy–rate frontier traced by sweeping the per-neuron spike-budget penalty θ_u (§5.4); each marker is a trained network, plotting mean hidden-E firing rate (abscissa) against test accuracy (ordinate). PING lies above and to the left of COBA across the sweep. At the operating point with θ_u disabled (starred), PING reaches $\approx 91\%$ at ≈ 12.3 Hz, against COBA's $\approx 91\%$ at ≈ 181 Hz. Source: exp025 rate-attractor analysis in exp024.

2.4 The firing-rate reduction does not require trained loop weights

Whether the firing-rate reduction in §2.3 is acquired during training or follows from the canonical loop weights is tested by two complementary experiments. We distinguish the two senses of "architectural": (a) the reduction occurs without gradient-based tuning of the loop weights (which §2.4 establishes), as opposed to (b) the reduction emerges from generic E↔I

structure without any hand-set values (which is not tested; the loop weights are held at the canonical biophysical values of [7]). The claim in this work is (a): the inductive bias is paid for at design time, not during training.

In the first, the recurrent loop is activated at inference on a trained COBA network, without retraining any weight. The network immediately produces gamma-banded rasters; the mean excitatory rate falls by approximately a factor of 15 ($\approx 133 \rightarrow \approx 9$ Hz), and test accuracy falls by ≈ 36 pp (Figure 4). The rate reduction therefore occurs without training. The accuracy loss is consistent with the absence of a learned compensation in the feedforward weights, which were optimised in the absence of the loop.

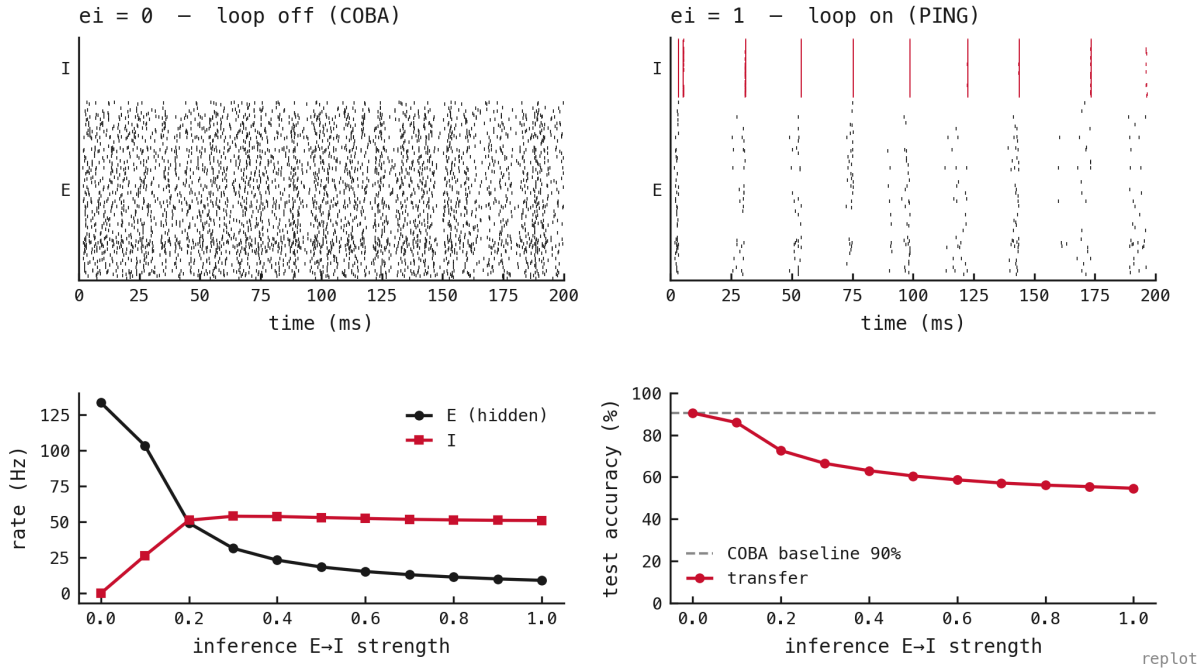


Figure 4: **Engaging the recurrent loop at inference on a trained COBA network reproduces the PING firing-rate reduction with no weight update.** A network trained in the COBA configuration is evaluated with the recurrent E→I coupling scaled from $ei = 0$ (loop off, as trained) to $ei = 1$ (canonical loop strength), without retraining. Top: single-trial rasters at $ei = 0$ (dense, asynchronous, inhibitory population silent) and $ei = 1$ (gamma bands, synchronous inhibitory bursts). Bottom left: mean excitatory (black) and inhibitory (red) firing rates versus inference coupling strength; the excitatory rate falls approximately 15-fold (from ≈ 133 to ≈ 9 Hz) as the inhibitory rate rises to ≈ 51 Hz. Bottom right: test accuracy versus coupling strength, falling from the $\approx 90\%$ COBA baseline to $\approx 55\%$ at $ei = 1$ (a ≈ 36 pp cost). The rate gating appears the instant the loop is wired in, whereas the accuracy loss reflects the absence of a feedforward compensation that only training in the presence of the loop provides. Source: exp038.

In the second, the loop weights W^{EI}, W^{IE} are released for training under the Dale's-law clamp. Both matrices represent non-negative conductance magnitudes; pathway identity fixes their reversal potentials, and the negative GABA reversal potential makes the I→E pathway inhibitory (§5.4). After each optimiser step, the trained magnitudes are projected onto the non-negative cone. In all conditions tested, the rhythmicity score collapses within a single training epoch: the first logged metric, after epoch 1, shows $R \approx 0.08$, compared with the

canonical initial value $R \approx 1$ (Figure 5). The collapse is faster than the per-epoch logging interval, so no intermediate state is recorded. From every initial condition tested (canonical PING values, zero, and 0.1 × canonical), the inhibitory firing rate remains near zero, R stays in 0.03–0.14, and final test accuracy is approximately 91% at ≈ 10 Hz mean E rate for the frozen-PING control (Figure 5). These numbers differ from the §2.3 frontier endpoint (≈91% at ≈12.3 Hz) because the §2.4 setup omits the spike-budget penalty θ_u and isolates the within-experiment contrast between frozen and trainable conditions rather than tracing the accuracy–rate frontier. Within this setup, gradient descent does not preserve or recover effective E→I recruitment from any tested initial condition.

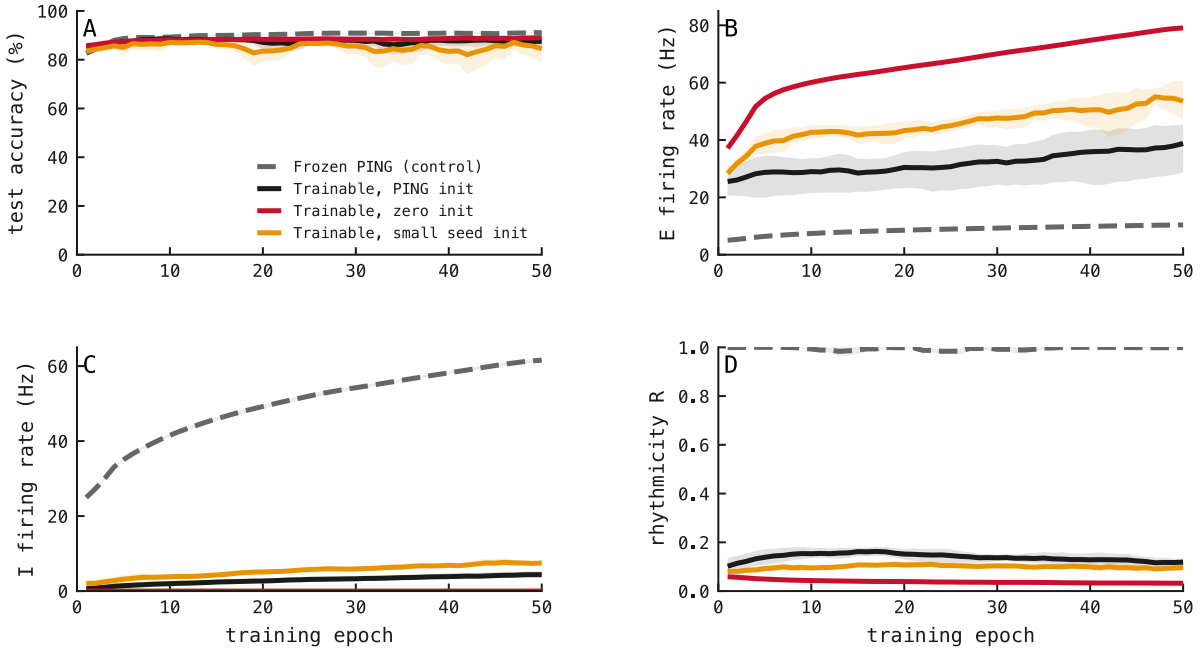
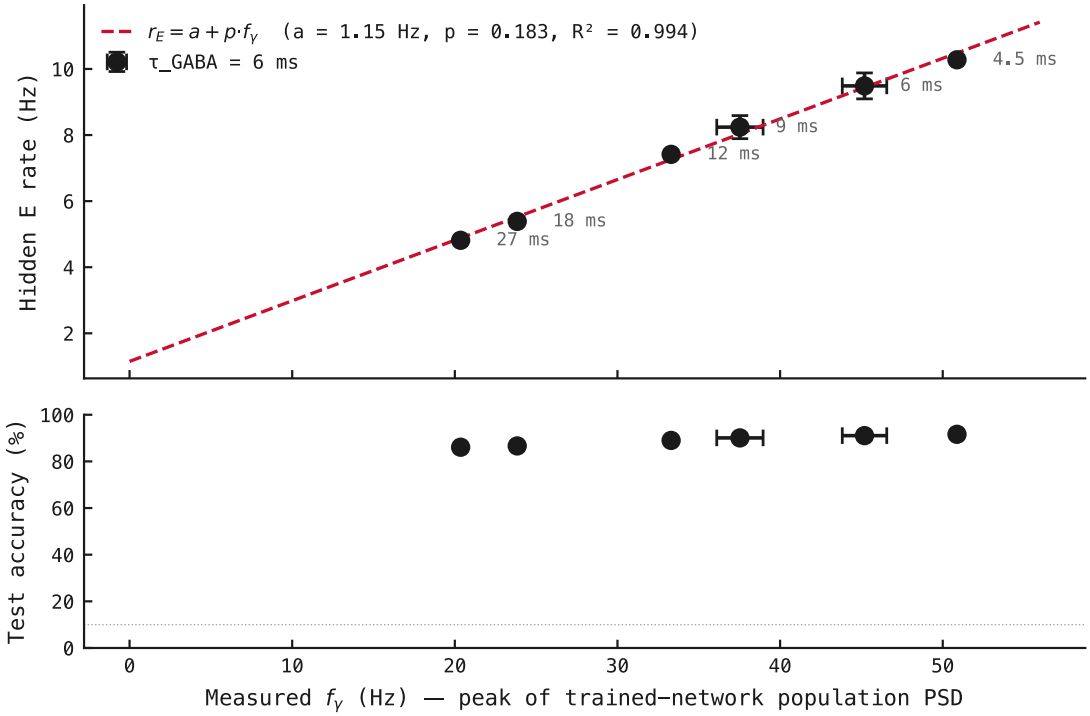


Figure 5: **Releasing the loop weights to gradient descent prunes the rhythm within a single epoch, from every initialisation.** Per-epoch training metrics with the recurrent weights W^{EI}, W^{IE} made trainable under the Dale's-law clamp, over 50 epochs on MNIST. Lines are the mean of three seeds (42–44); shading is the across-seed range. Conditions differ only in initialisation of the loop weights: canonical PING values (black), zero (red), and 0.1 × canonical (amber), against a frozen-PING control (grey, dashed). **(A)** Test accuracy: all conditions overlap at ≈ 90–92%. **(B)** Mean excitatory firing rate: the trainable conditions rise to ≈ 37–76 Hz as the loop is dismantled, while the frozen control remains gated near 10 Hz. **(C)** Mean inhibitory firing rate: the trainable conditions collapse to ≈ 0 Hz within a few epochs, while the frozen control's inhibitory rate rises to ≈ 57 Hz as its readout trains. **(D)** Lobe–trough rhythmicity contrast: the frozen control holds at ≈ 1 while every trainable initialisation drains to ≈ 0.03–0.14. Source: exp049.

The second result is conditional on the gradient-damping scheme that stabilises PING training (the gradient flowing through the loop is attenuated by a factor $1/d$ on the backward pass, with $d = 1000$; §5.4); a constrained-training scheme (§4 future directions) would test whether the loop's pruning depends on the damping regime. The first experiment (inference-time activation) uses no gradients and is unaffected by this caveat, and carries most of the weight of the §2.4 conclusion.

2.5 Post-training E rate covaries with gamma frequency

The post-training excitatory firing rate covaries approximately affinely with the measured gamma frequency. Across a sweep of τ_{GABA}, which jointly changes the inhibitory decay kinetics, integrated inhibitory influence, and realised f_γ, the trained r_E is well fit by $r_E = 1.15 + 0.183 f_\gamma$ ($R^2 = 0.994$, three seeds per point; Figure 6). Mean test accuracy declines from 91.6% at $\tau_{\mathrm{GABA}} = 4.5$ ms to 86.1% at 27 ms, a 5.5 percentage-point tradeoff across the sweep. The association has a cycle-resolved counterpart. Resolving spikes by (cell, cycle) pair, 98.87% contain at most one spike across the full τ_{GABA} sweep: $P(0) \approx 79\%$, $P(1) \approx 20\%$, and the multi-spike fraction (≥ 2) is $\approx 1\%$ (Figure 7). Because τ_{GABA} changes more than frequency alone, these experiments do not identify f_γ as the sole causal variable.



r007

Figure 6: **Post-training excitatory rate covaries affinely with gamma frequency across a sweep of inhibitory decay kinetics.** Networks were trained from scratch at each of six values of the GABA decay constant $\tau_{\mathrm{GABA}} \in \{4.5, 6, 9, 12, 18, 27\}$ ms. Changing τ_{GABA} alters both the realised gamma frequency and the duration and integrated influence of inhibition (§5.4). Top: mean post-training excitatory firing rate against measured f_γ; markers are per-condition means over three seeds, with error bars ($\pm$ SD) on both axes. The linear fit $r_E = 1.15 + 0.183 f_\gamma$ passes through every error bar ($R^2 = 0.994$). Bottom: mean test accuracy declines from 91.6% at $\tau_{\mathrm{GABA}} = 4.5$ ms to 86.1% at 27 ms, a 5.5 percentage-point tradeoff. The sweep establishes covariance with realised frequency, not an independent causal effect of frequency alone. Source: exp041.

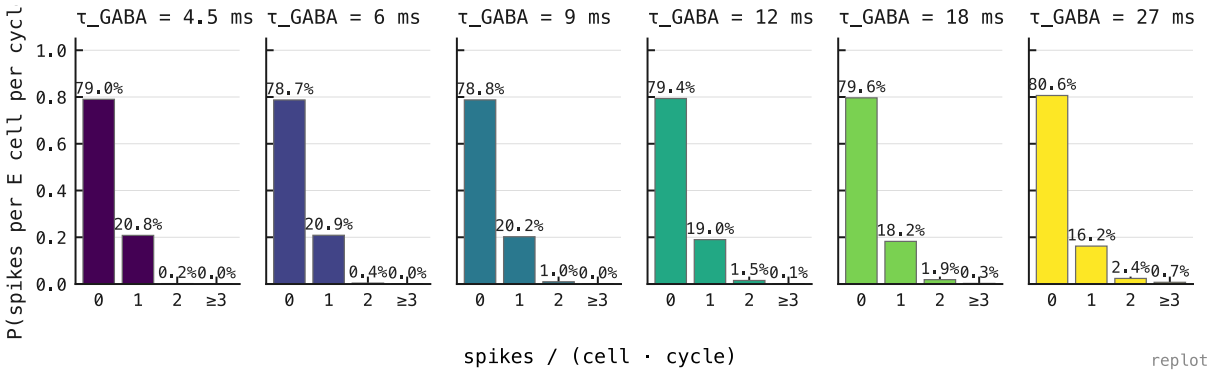


Figure 7: **The affine rate law follows from a near-binary per-cycle firing statistic: each excitatory cell contributes at most one spike per gamma cycle.** Excitatory spikes were resolved into (cell, cycle) pairs by assigning each spike to the gamma cycle inferred from peaks of the population inhibitory-burst rate (§5.5). Each panel shows the distribution of spikes per pair (0, 1, 2, or ≥ 3) at one value of τ_{GABA}. Across the sweep, $P(0) \approx 79\%$, $P(1) \approx 20\%$, and the multi-spike fraction (≥ 2) stays near $\approx 1\%$; pooled over all conditions, 98.87% of pairs contain at most one spike. Source: exp046.

2.6 Dynamics and robustness

The gating depends on the timing of inhibition, not its mean level. Perturbations of the trained PING network at inference reveal a deletion-versus-addition asymmetry. From an unperturbed baseline of approximately 91% accuracy, PING retains approximately 85% accuracy under deletion of 80% of emitted spikes (little degradation); addition of off-phase Poisson noise instead drives accuracy down to chance as the injected rate grows, because those spikes recruit the inhibitory pool at arbitrary phase and disrupt the rhythm [**31**]. COBA exhibits the opposite asymmetry (Figure 8). Two inference-time jitter perturbations of the inhibitory spike train, both holding the mean inhibitory rate fixed, produce opposite effects on the excitatory rate: per-cell jitter smears each burst into a continuous shunt and reduces the excitatory rate to zero, while cycle-coherent jitter preserves within-burst synchrony and raises the excitatory rate from ≈ 9 Hz to ≈ 36 Hz (Figure 9) [**32**].

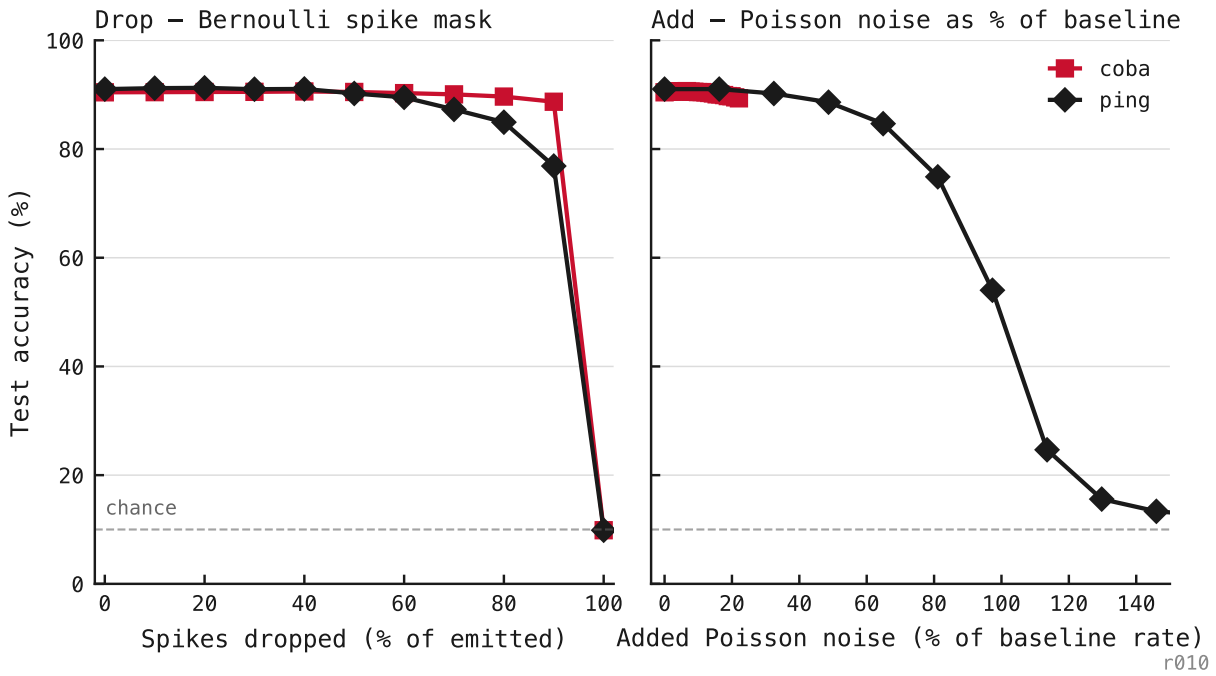


Figure 8: **The gating is robust to spike deletion but fragile to spike addition, the expected signature of a phase-based code.** A trained PING network is perturbed at inference (no retraining); accuracy is plotted against perturbation level. Left: deletion of a random fraction of emitted spikes. PING retains ≈ 85% accuracy through 80% deletion, close to its unperturbed value, degrading only as the network approaches silence. Right: addition of Poisson spikes, expressed as a fraction of each population's baseline rate. PING accuracy falls to chance as added noise grows, because off-phase spikes drive the inhibitory pool at arbitrary phase and dissolve the rhythm. COBA (red), having no rhythm to protect, shows the mirror-image asymmetry: tolerant to addition, intolerant to deletion. Source: exp037.

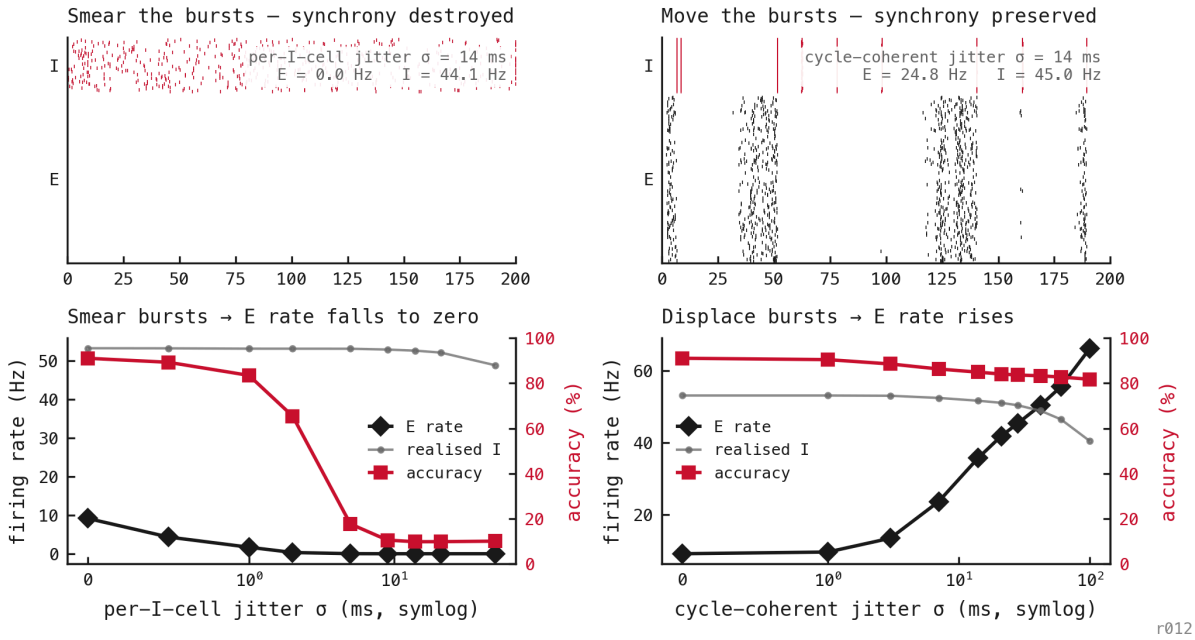


Figure 9: **Two inhibitory-jitter manipulations at the same magnitude that hold the mean inhibitory rate fixed drive the excitatory rate in opposite directions, isolating timing from level.** The trained PING inhibitory spike train is perturbed at inference while the mean per-cell inhibitory rate is held constant. Both arms use the same jitter magnitude, $\sigma = 14$ ms — only the *kind* of jitter differs. Top: single-trial rasters (E black, I red). Bottom: mean excitatory rate (black) and accuracy (red) versus jitter magnitude σ, with the realised mean inhibitory rate overlaid (grey). Left: per-cell jitter smears each burst into a continuous shunt; the excitatory rate falls to ≈ 0 Hz (inhibitory rate ≈ 53 Hz) and accuracy falls to chance. Right: cycle-coherent jitter displaces whole bursts while preserving within-burst synchrony; the excitatory rate rises from ≈ 9 to ≈ 36 Hz (inhibitory rate ≈ 52 Hz, matched to the left arm) and accuracy holds high. Both arms are read where the realised inhibitory rate is matched to within a few percent; at larger σ the cycle-coherent excitatory rate climbs further, but the finite trial window truncates the most-displaced bursts and realised inhibition falls, so the strict comparison is anchored here. Identical mean inhibition, opposite excitatory outcome: the gate is the phase structure of inhibition, not its level. Source: exp042.

The post-training firing rate is also approximately invariant under change of integration timestep: across $\Delta t \in [0.05, 1.0]$ ms (a $20\times$ range), the trained excitatory rate stays in 9.1–13.6 Hz and accuracy varies by less than 1.1 pp (Figure 10). The rate is therefore a property of the continuous dynamics, not an artefact of the discretisation.

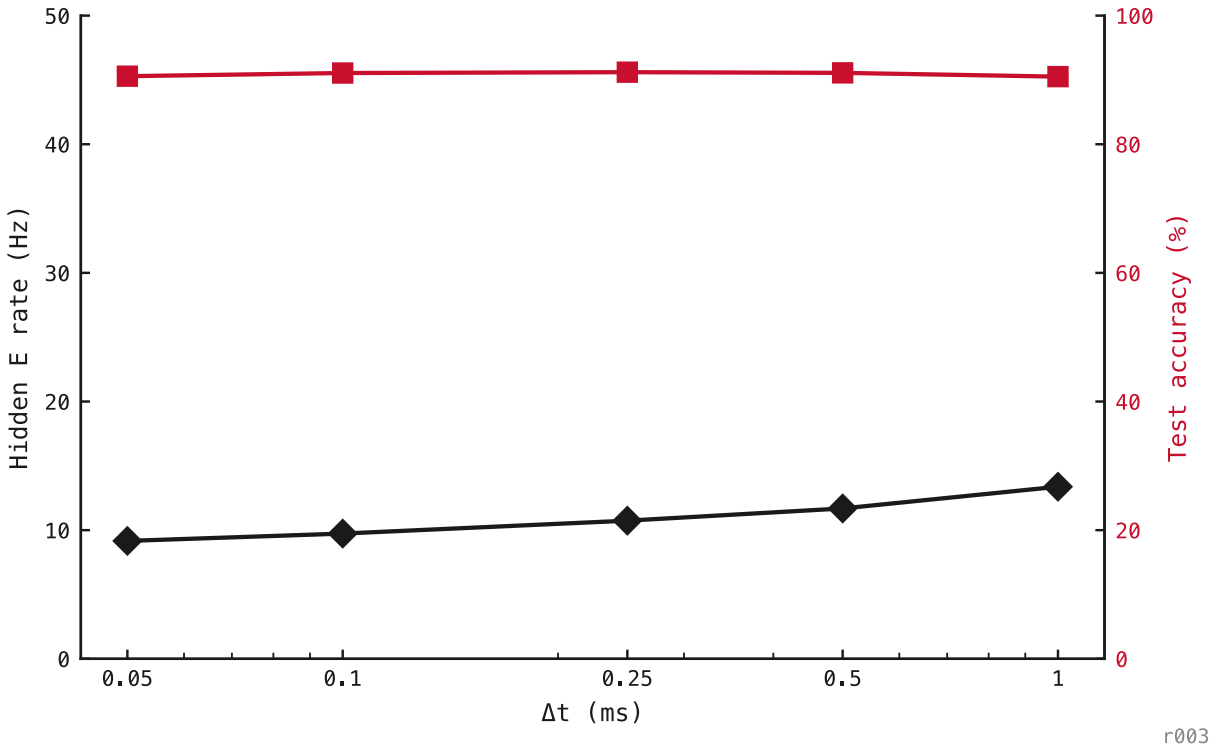


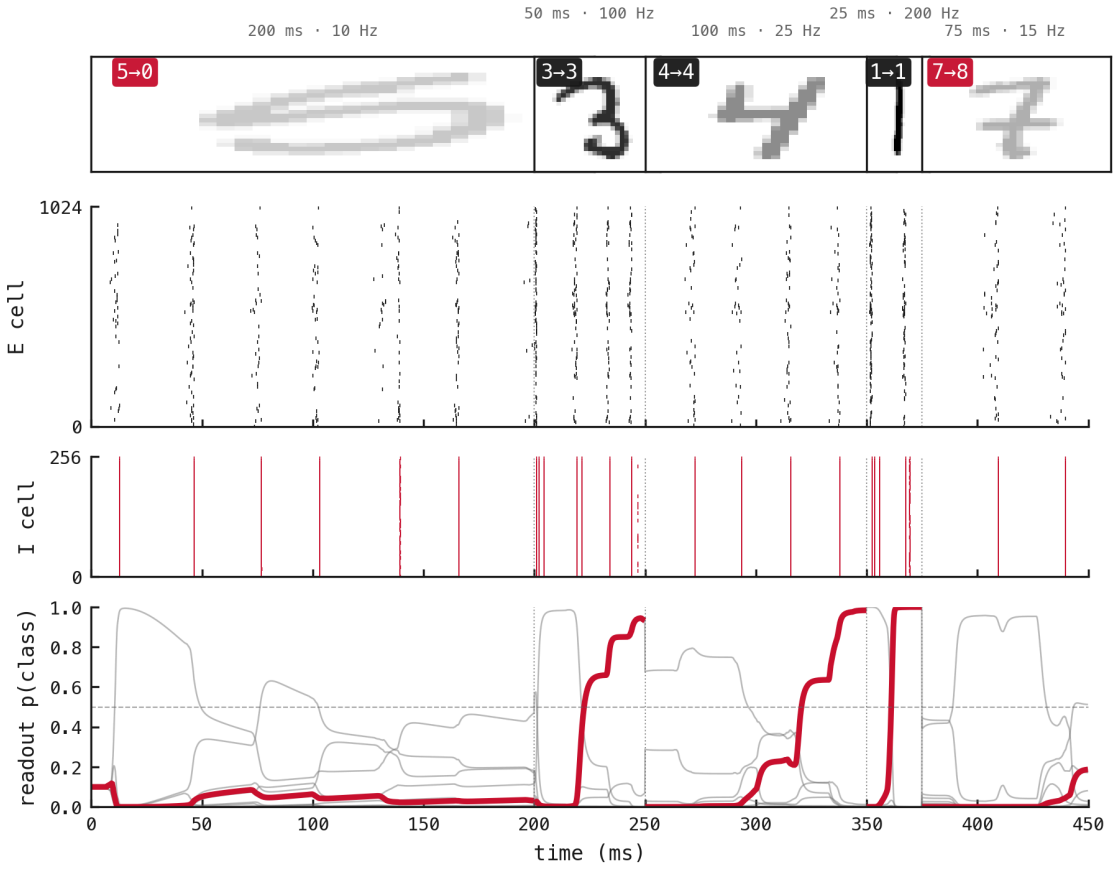
Figure 10: **The firing-rate reduction is a physical-time property, invariant to the integration timestep over a twentyfold range.** The network was trained and evaluated at matched integration timestep Δt for each value across $[0.05, 1.0]$ ms (logarithmic abscissa). Left ordinate (black diamonds): post-training mean excitatory rate, confined to a 9.1–13.6 Hz band and non-monotonic in Δt, so finer stepping does not buy a lower rate. Right ordinate (red squares): test accuracy, flat within 1.1 pp (90.4–91.4%). Because both training and inference use the same Δt at each point, the figure tests invariance of the training-plus-inference pipeline, not the generalisation of one trained network to a varied inference step. Source: exp044.

2.7 Streaming classification on continuous input

The preceding subsections evaluate the network on isolated single-digit presentations. The streaming protocol tests whether the firing-rate reduction is preserved under continuous input.

A PING network trained on single-digit MNIST classifies a continuously concatenated input stream without retraining and without a segmentation cue. The streaming protocol (§5.7) uses a time-averaged non-spiking LIF readout over a sliding window whose duration is matched exactly to each segment's presentation duration. On a representative stream of five digits, each with its own duration (25–200 ms) and input rate (10–200 Hz), 3 of 5 are classified correctly (Figure 11). Across the (τ, input-rate) grid, accuracy is approximately a function of the product $\tau \cdot$ rate. Accuracy does not exceed $\approx 80\%$ for $\tau \leq 15$ ms regardless of input rate (Figure 12A). For reference, 15 ms is approximately 0.6 times the canonical gamma period $T_\gamma \approx 27$ ms. Accuracy saturates by $\tau \approx 40$–50 ms, and the trained operating point ($\tau = 200$ ms, rate $= 25$ Hz) reaches 93% accuracy. Extending the 200 ms slice below the grid's 5 Hz minimum locates a separate encoder floor: performance remains at chance through 0.5 Hz, becomes clearly informative by 2 Hz, and reaches 79.1% at 5 Hz (Figure

12B). The failed 200 ms, 10 Hz segment in Figure 11 occurs at a population-level accuracy of 87.3%, so it is a natural weak-evidence classification error rather than evidence that the encoding rate is intrinsically nonviable. Together the panels establish requirements for sufficient integration time and sufficient encoded input evidence; because gamma frequency is not independently varied, they do not identify the gamma cycle as the cause or temporal unit of either requirement.



exp048-replot

Figure 11: **A PING network trained on isolated digits classifies a continuous, unsegmented stream whose presentation timing varies from segment to segment.** A single input stream concatenates five MNIST digits, each with its own presentation duration (25–200 ms) and Poisson input rate (10–200 Hz); the network is given no segmentation cue and is not retrained. Top: the five digit thumbnails with their per-segment (duration, rate) and predicted labels; thumbnail opacity indicates input rate. Below: hidden excitatory and inhibitory rasters, showing gamma cycles maintained throughout (sparser under weak drive, denser under strong drive) and the sliding leaky-integrator readout traces. 3 of 5 digits are classified correctly; the two errors occur in the weakest-drive segments and are interpreted against the population curve in Figure 12B. Source: exp048.

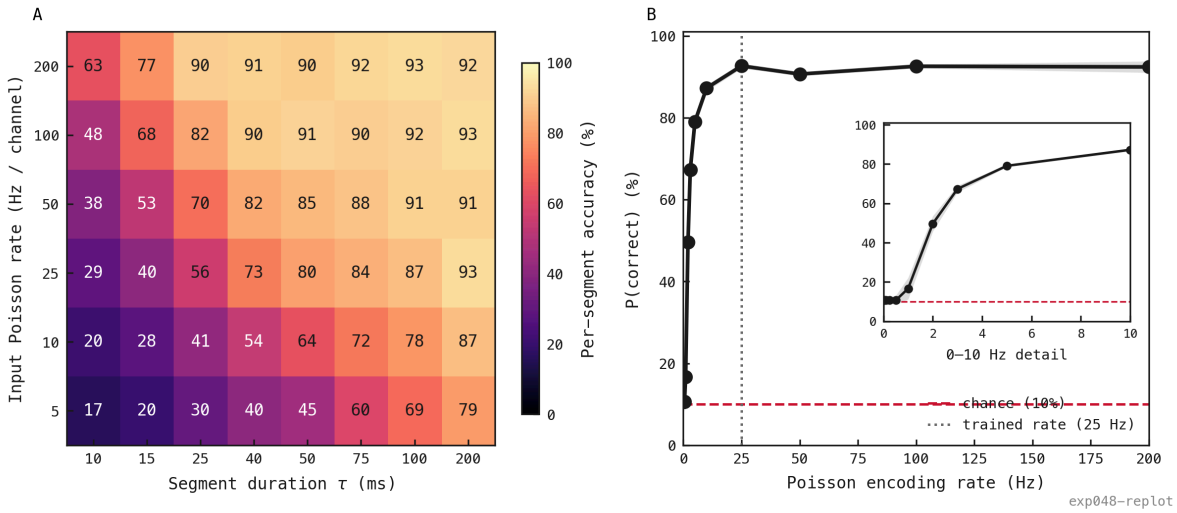


Figure 12: **Streaming accuracy has distinct integration-time and encoding-rate evidence floors. (A)** Per-segment accuracy across presentation duration τ and input rate, averaged over three seeds and 1,200 segments per grid cell. For $\tau \leq 15$ ms, accuracy does not exceed $\approx 80\%$ at any input rate; above that floor, diagonal iso-accuracy contours show an approximate dependence on $\tau \cdot \text{rate}$. The trained operating point ($\tau = 200$ ms, 25 Hz) reaches 93%. **(B)** Probability of a correct classification versus encoding rate with both presentation duration and readout window fixed at 200 ms. All points belong to the same psychometric curve. Performance remains at chance through 0.5 Hz, is clearly informative by 2 Hz, and reaches 79.1% at 5 Hz; the dotted line marks the 25 Hz training rate. Thus the Figure 11 error at 200 ms and 10 Hz occurs in a condition with 87.3% population accuracy, above the nonviable encoder regime. For scale, $\tau = 15$ ms is approximately 0.6 times the canonical gamma period $T_\gamma \approx 27$ ms, but gamma frequency is not manipulated independently. Source: exp048.

3. Discussion

A recurrent E↔I loop, held fixed during training, generates a gamma rhythm and reduces the post-training excitatory firing rate by approximately an order of magnitude relative to the COBA baseline at matched accuracy (§2.3). The reduction does not require gradient-based learning of the loop weights: activating the loop at inference on a trained COBA network reduces the firing rate without retraining (§2.4), and gradient descent does not preserve the loop within a single epoch when its weights are released (§2.4, conditional on the §5.4 damping scheme). The inductive bias is paid for at design time, via the canonical biophysical loop weights [**7**], not during training. Within the mean-field reduction, the gamma onset is a supercritical Hopf bifurcation (§2.2), and the empirical data support this classification. Its continuous, reversible character fits the inference-time loop activation of §2.4, which produces a graded change in firing rate without hysteresis; a direct test would require an inference-time hysteresis sweep on the E→I gain (§4).

The mechanism of the gate is the temporal structure of inhibition, not its mean level. The jitter perturbation experiment (Figure 9) is a direct test: holding the mean per-cell inhibitory rate fixed and varying its temporal structure produces opposite effects on the excitatory rate depending on whether within-burst synchrony is preserved, consistent with prior

characterisations of temporal-synchrony patterns within PING circuits [**32**]. The robustness asymmetry under spike addition versus deletion (Figure 8) follows from this dependence on phase: removal of spikes does not alter the phase structure of the population output, while addition of off-phase spikes does. The asymmetry constitutes a testable prediction for biological gamma circuits and would speak to long-standing rate-vs-timing debates [**33, 34**]. It echoes prior reports that oscillations sharpen spike-timing precision [**31**].

Across the τ_{GABA} sweep, post-training rate covaries with the realised gamma frequency, and the majority of (cell, cycle) pairs contain at most one spike (Figures 6–7). This is consistent with cycle-structured rate control, but τ_{GABA} simultaneously changes inhibitory decay, integrated conductance, and burst duty cycle; the experiment does not isolate frequency as the sole cause of the rate change. An independent manipulation of oscillation frequency at matched inhibitory influence would be needed for that attribution. The streaming experiment (Figure 12) instead separates two evidence bounds: panel A shows low accuracy at short presentation durations and an approximate dependence on $\tau \cdot \text{rate}$, whereas panel B locates the 200 ms encoder floor below 0.5 Hz. Errors above that floor are ordinary trial-level failures under weak evidence, not evidence that the rate is categorically unusable. The numerical proximity of the short-duration floor to the canonical gamma period is descriptive, not mechanistic, because gamma frequency is not independently varied. A gamma-frequency sweep or a cycle-aligned analysis showing discontinuities at integer multiples of T_γ would be required to test whether gamma defines a temporal unit for classification.

PING is a structured alternative to the asynchronous balanced state [**36, 37**]. The architectural treatment of the loop adopted here differs from the inhibitory-plasticity literature, in which the inhibitory connectivity is plastic and learns E/I balance [**38, 39, 40, 41**]. The §2.4 result, that gradient descent does not preserve the loop from any tested initial condition (under the §5.4 damping regime), provides empirical support for the architectural treatment in this setting. We propose a functional interpretation: gamma may act as a structural constraint on excitatory firing rates without requiring learned tuning of the inhibitory connectivity.

Relative to the two closest recent trainable-SNN precedents in the bibliography, the present work differs in how the rhythm is obtained rather than in claiming better performance. [**25**] imposes the oscillation as an external input to spiking neurons; [**26**] trains an adaptive-LIF network on speech, all parameters free, and reports that oscillatory synchronisation and cross-frequency coupling *emerge* from end-to-end optimisation, correlating with task performance. The §2.4 result that gradient descent does not preserve the loop is therefore not a claim that surrogate-gradient training cannot discover rhythm in general ([**26**] shows it can) but a narrower one: it does not maintain a *fixed, biophysically-calibrated PING loop* whose weights are released under the Dale's-law clamp and the damping regime of §5.4. The two findings are complementary poles of the same question: rhythm acquired by training versus rhythm supplied by architecture. Neither [**25**] nor [**26**] uses a conductance-based E↔I loop, and neither attributes a firing-rate reduction to the rhythm via inference-time activation, frequency tuning, or jitter perturbation as §2.4–§2.6 do; their firing rates are roughly an order of magnitude higher than the rates reported here, and neither frames a per-spike economy. We do not provide a head-to-head numerical comparison: [**25**] and [**26**]

evaluate on temporally structured tasks (SHD; speech perception) where the present static-MNIST protocol is not directly comparable. The contribution is mechanistic, not benchmark-driven.

The reduction reported in §2.3 should be considered net of the inhibitory contribution. PING uses a smaller, higher-rate inhibitory population, so the reduction in total spike count ($N_E\langle r_E\rangle + N_I\langle r_I\rangle$) is approximately a factor of 7 rather than a factor of 15. Excitatory glutamatergic signalling accounts for a larger share of the cortical energy budget than inhibitory transmission [**28, 29**] weighting spikes by metabolic cost recovers the order-of-magnitude figure. The argument is complicated by the substantial per-cell metabolic demands of fast-spiking interneurons, which sustain high firing rates and dense synaptic activity [**42**] for this reason we treat the uniform-spike-counting reduction (approximately 7-fold) as the more conservative claim. We do not attempt a quantitative metabolic comparison with cortex, given the architectural differences (no W_{ee} or W_{ii}, idealised synapse counts).

Several limitations apply. The evaluation uses a single dataset (MNIST), a single readout, and a fixed loop topology. MNIST is a simple, near-saturated benchmark on which many architectures reach comparable accuracy, so the classification results here should be read as evidence that the firing-rate reduction survives training to competence, not as a claim about task difficulty or about generalisation to harder problems; whether the mechanism holds on datasets with intrinsic temporal structure is left to future work (SHD, §4). The rhythmicity metric R is one of several available options. The mean-field reduction is biophysically calibrated but is a reduction of the full network. The architecture excludes W_{ee} and W_{ii}, conduction delays, and cell-type heterogeneity; the implications for cortical microcircuits with richer connectivity are open [**15**]. The streaming evaluation does not include temporally structured inputs in which classification depends on input timing. The capacity of a single gamma cycle and its scaling with assembly size are not characterised here [**43**]. Classification accuracy on MNIST is approximately 83–90%; the contribution of the present work concerns the mechanism by which the firing rate is reduced rather than the absolute accuracy. The §2.4 released-loop result depends on the gradient-damping scheme ($d = 1000$, §5.4) and the Dale clamp; whether the loop's collapse persists under weaker damping is not addressed here. The paper does not benchmark against rhythmic-SNN baselines [**25, 26**] the PING-specific attribution rests on the within-architecture experiments of §2.4–§2.6 rather than on an external rhythmic-SNN comparison.

4. Conclusion and Future Directions

A recurrent E↔I loop held fixed during training reduces the post-training excitatory firing rate by approximately an order of magnitude relative to the COBA baseline at matched accuracy, and approximately 7-fold by total spike count when the higher-rate inhibitory pool is included (§3 para 5). The reduction is invariant to the integration timestep across a 20 × range (Figure 10, retrained at each Δt) and is preserved under evaluation on continuously concatenated input streams (Figures 11–12).

Empirical extensions include evaluation on the SHD dataset [**44**], which tests the gating on a spiking benchmark with intrinsic temporal structure, and tasks in which classification accuracy depends on input timing, which could test whether the gamma cycle acts as a temporal unit. This is the regime in which the emergent-oscillation route of [**26**] operates; a

direct comparison there would set the imposed, biophysically-calibrated PING loop studied here against a network free to learn its own rhythmic structure, on the same temporally structured task. A more comprehensive characterisation of the $W^{EI} \times W^{IE}$ plane would test the supercritical Hopf classification directly.

Theoretical extensions include multi-layer PING architectures, an independent two-dimensional sweep of the two loop-weight gains $\alpha_{EI} \times \alpha_{IE}$, multi-rhythm and theta-nested gamma models [**19, 20**], and characterisation of capacity limits for single cycles and minimal assemblies [**43**]. A constrained-training scheme, in which the loop weights are regularised toward biophysical values rather than held fixed, would connect the present result to the inhibitory-plasticity literature [**38**]. The rate law of §2.5 admits a testable biological prediction: perturbing the timing of inhibitory neurons in vivo (e.g. by optogenetic stimulation, as in [**8, 9, 10**]) should change the excitatory firing rate without altering the mean inhibitory rate, the in vivo analogue of Figure 9.

Gamma may act as a structural mechanism by which the cortical microcircuit maintains sparse excitatory firing rates at low metabolic cost. The present work provides one architecture in which this mechanism is realised explicitly.

5. Methods

5.1 Single-neuron and synapse dynamics

The model uses a conductance-based leaky integrate-and-fire (LIF) representation with two populations, excitatory (E) and inhibitory (I). The sub-threshold membrane potential of each population evolves as

$$C_m^E \frac{\mathrm{d}V^E}{\mathrm{d}t} = -g_L^E(V^E - E_L) - g_e^E(V^E - E_e) - g_i^E(V^E - E_i)$$
$$C_m^I \frac{\mathrm{d}V^I}{\mathrm{d}t} = -g_L^I(V^I - E_L) - g_e^I(V^I - E_e)$$

where C_m is the membrane capacitance, g_L the leak conductance, E_L the leak reversal potential, and E_e, E_i the excitatory and inhibitory synaptic reversal potentials. The I population has no inhibitory term because there is no I→I connection in this architecture (§5.2).

A neuron emits a spike when V crosses threshold V_{th} from below; the membrane potential is then reset to V_{reset} for a refractory period τ_{ref}:

$$s_{t+1} = \mathbf{1}[V \geq V_{\text{th}}], \quad V \leftarrow V_{\text{reset}} \text{ if } s_{t+1} = 1 \text{ or refractory.}$$

Each synaptic conductance is an exponential trace driven by presynaptic spikes; a presynaptic spike adds its full weight W as an instantaneous jump in conductance, which then decays with the relevant channel time constant (τ_{AMPA} for AMPA-like excitation, τ_{GABA} for GABA-like inhibition):

$$\frac{\mathrm{d}g_e^E}{\mathrm{d}t} = -\frac{g_e^E}{\tau_{\mathrm{AMPA}}} + W_{\mathrm{in}} \sum_k \delta\big(t - t_k^{\mathrm{inp}}\big)$$

$$\frac{\mathrm{d}g_i^E}{\mathrm{d}t} = -\frac{g_i^E}{\tau_{\mathrm{GABA}}} + W^{IE} \sum_k \delta\big(t - t_k^{i}\big)$$

$$\frac{\mathrm{d}g_e^I}{\mathrm{d}t} = -\frac{g_e^I}{\tau_{\mathrm{AMPA}}} + W^{EI} \sum_k \delta(t - t_k^{e})$$

The first equation describes input-driven excitation onto E via feedforward weights W_{in}; the second is inhibition onto E from I via W^{IE}; the third is excitation onto I from E via W^{EI}. There is no equation for g_i^I (no I→I connection) and no recurrent contribution to g_e^E (no E→E connection). Canonical parameter values are listed in the parameters table; the E and I populations differ in membrane capacitance, leak conductance, membrane time constant, and refractory period.

Canonical parameter values for the spiking network are given below. Where E and I populations use different values they are listed as "E / I"; otherwise the value is shared. The $\tau_{\mathrm{GABA}} = 9$ ms value is the canonical PING value of [**7**].

Symbol	**Description**	**Value**
C_m	Membrane capacitance	1.0 / 0.5 nF
g_L	Leak conductance	0.05 / 0.1 μS
τ_{ref}	Refractory period	3.0 / 1.5 ms
E_L	Leak reversal potential	−65 mV
V_{th}	Spike threshold	−50 mV
V_{reset}	Reset potential	−65 mV
E_e	AMPA reversal potential	0 mV
E_i	GABA reversal potential	−80 mV
τ_{AMPA}	AMPA decay time constant	2 ms
τ_{GABA}	GABA decay time constant	9 ms
Δt	Integration timestep	0.1 ms (train) / 0.25 ms (inference)
N_E	Hidden excitatory pool size	1024
N_I	Inhibitory pool size	256

5.2 Network architecture

The network has one hidden layer with N_E excitatory and N_I inhibitory units, and a non-spiking leaky-integrator readout layer with weights W_{out} over the excitatory population. The readout units follow LIF membrane dynamics with no spike, reset, or refractory period; per-class logits are computed as the membrane potentials averaged over the presentation window. Input spikes drive the excitatory population via feedforward weights W_{in}. Recurrence is restricted to the E↔I loop: E projects to I via W^{EI}, and I projects back to E via W^{IE}. There is no W_{ee} and no W_{ii}.

The restriction to the E↔I loop is intended to make the rhythm unambiguously PING: excluding W_{ii} rules out ING, and excluding W_{ee} rules out recurrent-E driven oscillation [**7,**

45]. The conductance-based (COBA) baseline used as a non-PING control is the loop-off limit of the same architecture, obtained by setting $W^{EI} = W^{IE} = 0$.

The loop weights W^{EI} and W^{IE} are held fixed (untrained) in the experiments reported in §2.3 and §2.5–§2.7. The loop is treated as a structural prior, consistent with the inhibitory-plasticity literature, in which inhibitory synapses serve an experience-dependent E/I balance role rather than carrying the feedforward computational features that the excitatory pathway acquires [**38, 39**]. The choice is also supported empirically by the result in §2.4 (Figures 4–5) that gradient descent does not preserve effective E→I recruitment when the recurrent conductances are released. A Dale's-law clamp constrains all trained conductance magnitudes to remain non-negative throughout training (§5.4).

5.3 Mean-field reduction

To locate the gamma-onset bifurcation analytically (§2.2), the spiking network is reduced to a four-dimensional rate model in the state

$$\boldsymbol{x}(t) = \big(\bar{E}, \bar{I}, \bar{g}_e^I, \bar{g}_i^E\big),$$

where $\bar{E}$ and $\bar{I}$ are the population-mean firing rates of the E and I cells (in spikes per millisecond), and $\bar{g}_e^I$, $\bar{g}_i^E$ are the population-mean cross-population synaptic conductances onto the I and E populations. The two cross-population conductances are sufficient because the architecture has no W_{ee} and no W_{ii} (§5.2); the within-population conductances vanish identically.

The reduction replaces the spike-driven conductance dynamics with rate-driven first-order filters, and replaces each cell's stochastic spike output with the population-averaged firing rate of a noise-driven LIF cell at a given mean input current. The closed 4D system is

$$\begin{aligned}
\tau_E \dot{\bar{E}} &= -\bar{E} + \Phi_E(I_{\text{ext}} - \bar{g}_i^E \Delta V_{\text{inh}}) \\
\tau_I \dot{\bar{I}} &= -\bar{I} + \Phi_I(\bar{g}_e^I \Delta V_{\text{exc}}) \\
\tau_{\text{AMPA}} \dot{\bar{g}}_e^I &= -\bar{g}_e^I + \tau_{\text{AMPA}} W^{EI} \bar{E} \\
\tau_{\text{GABA}} \dot{\bar{g}}_i^E &= -\bar{g}_i^E + \tau_{\text{GABA}} W^{IE} \bar{I}
\end{aligned}$$

where I_{ext} is an external tonic drive to E (the bifurcation control parameter, below), and the driving forces are $\Delta V_{\text{exc}} = E_e - E_L = 65$ mV and $\Delta V_{\text{inh}} = E_L - E_i = 15$ mV evaluated at rest. The membrane time constants are the passive ratios $\tau_E = C_m^E / g_L^E = 20$ ms and $\tau_I = C_m^I / g_L^I = 5$ ms, computed from the capacitances and leak conductances in the §5.1 parameters table; the synaptic time constants τ_{AMPA}, τ_{GABA} are taken directly from that table. The fan-in-normalised coupling strengths are $W^{EI} = 1.0$ μS and $W^{IE} = 2.0$ μS, inherited from the spiking network.

The population gain functions Φ_E and Φ_I are the noise-driven LIF rate functions in Ricciardi–Siegert form [**46**]. For mean input current μ delivered to a cell with leak conductance g_L, leak reversal E_L, threshold V_{th}, reset V_{reset}, membrane time constant τ_m, and refractory period τ_{ref},

$$\Phi(\mu) = \left[\tau_{\mathrm{ref}} + \tau_m\sqrt{\pi}\int_{(V_{\mathrm{reset}}-\mu_V)/\sigma_V}^{(V_{\mathrm{th}}-\mu_V)/\sigma_V} e^{u^2}(1+\operatorname{erf} u)\,\mathrm{d}u\right]^{-1},$$

with mean membrane potential $\mu_V = E_L + \mu/g_L$ and effective membrane-noise scale σ_V. The integral is evaluated by numerical quadrature. All parameters of Φ are taken from the §5.1 parameters table (with $\tau_m = C_m/g_L$) separately for the E and I populations. The noise scale σ_V is set to 4 mV; the predicted Hopf frequency varies by less than 1 Hz across $\sigma_V \in [3, 6]$ mV.

The silent (non-oscillating) fixed point is tracked as I_{ext} is swept from 0 to 4 nA in 10 µA steps. At each I_{ext}, the fixed point is obtained by solving the algebraic system in $(\bar{E}, \bar{I})$ (at steady state the two conductances are determined by the rates, $\bar{g}_e^I = \tau_{\mathrm{AMPA}} W^{EI}\bar{E}$ and $\bar{g}_i^E = \tau_{\mathrm{GABA}} W^{IE}\bar{I}$) using *scipy.optimize.fsolve*. The numerical Jacobian of the full 4D system at each fixed point is computed by central finite differences. The Hopf threshold $I_{\mathrm{ext}}^\star$ is the smallest I_{ext} at which the eigenvalue $\lambda^\star$ with largest real part crosses zero with non-zero imaginary part; the crossing frequency is $f^\star = |\mathrm{Im}\ \lambda^\star|\ /(2\pi)$.

The onset is classified numerically by a quasi-static amplitude sweep. I_{ext} is ramped up across $I_{\mathrm{ext}}^\star$ from a small perturbation of the silent fixed point and then ramped down; at each step the 4D system is integrated to its steady-state oscillation amplitude A. The onset is classified as supercritical when (i) the up- and down-ramp branches coincide within a peak hysteresis of 10^{-4} in rate units, and (ii) the squared amplitude scales linearly with the bifurcation distance,

$$A^2 \propto (I_{\mathrm{ext}} - I_{\mathrm{ext}}^\star),$$

with $R^2 > 0.9$. For the canonical parameter set the criterion is met with hysteresis below 10^{-5} and $R^2 = 0.999$.

The mean-field prediction is compared with the gamma frequency measured in the spiking network, extracted as in §5.5, across a sweep of $\tau_{\mathrm{GABA}} \in \{4.5, 6, 9, 12, 18, 27\}$ ms (Figure 2). Both curves decrease monotonically with τ_{GABA}; the spiking measurement is consistently higher than the rate-equation prediction across the sweep. The reduction captures the qualitative dependence on the GABA decay time, which is the use to which it is put. The treatment follows the Wilson–Cowan tradition for cortical-rhythm modelling [**18**] and the broader population-dynamics and next-generation neural-mass literature [**19, 47, 48**].

5.4 Training

The network is trained on MNIST by surrogate-gradient descent through backpropagation in time [**22, 23**]. Each digit is rate-encoded as a Poisson spike train over a 200 ms presentation window at a peak rate of 25 Hz per active pixel (Bernoulli per timestep with $p = r_{\max}\Delta t$). The loss is cross-entropy on the time-averaged membrane potentials of the non-spiking LIF readout, with a per-neuron firing-rate penalty active when a unit's mean rate exceeds a soft ceiling θ_u:

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + s_u \sum_{i\in E} \mathrm{ReLU}(\langle r_i\rangle - \theta_u)^2,$$

where $\langle r_i \rangle$ is the per-trial mean spike count of E neuron i, θ_u is a per-neuron rate ceiling expressed in spikes per trial, and $s_u = 10^{-3}$ is the penalty strength. Sweeping θ_u over $\{\text{off}, 5, 2, 1, 0.5, 0.2\}$ spikes per 200 ms trial (equivalent peak rates of $25, 10, 5, 2.5, 1$ Hz) generates the accuracy–rate frontier reported in §2.3.

The discrete spike nonlinearity has zero gradient almost everywhere; in the backward pass it is replaced by a fast-sigmoid surrogate of the distance to threshold $u = V - V_{\text{th}}$,

$$\frac{\partial \mathbf{1}[V \geq V_{\text{th}}]}{\partial V} \equiv \frac{s}{(1 + s\ |u|)^2},$$

with slope $s = 1$ **[23, 49]**. The forward pass evaluates the Heaviside exactly. Optimisation uses Adam **[50]** with learning rate 4×10^{-4}, batch size 256, and 50 epochs. Each baseline condition is repeated across three seeds (42, 43, 44); the spike-budget sweep uses a single seed (42).

Gradient damping for the PING configuration. Surrogate-gradient training of the PING configuration is unstable without intervention on the gradient. A single loop weight (W^{EI} or W^{IE}) contributes to the membrane-voltage update of every cell at every subsequent timestep within a trial; combined with the spike-driven impulse updates of the synaptic conductances and the non-zero surrogate gradient at the spike threshold, the gradient propagates through a tightly coupled feedback loop with millisecond-scale conductance dynamics. Over the 2,000-step backpropagation-through-time window of a single 200 ms trial at $\Delta t = 0.1$ ms, multiplicative contributions across timesteps cause gradient norms to grow by many orders of magnitude, and training does not converge. The mechanism is the same multiplicative compounding through long unrolled recurrences that characterises the exploding-gradient pathology in standard recurrent networks **[51]**.

We address this by attenuating the gradient flowing through the membrane-voltage increment $\mathrm{d}V$ on the backward pass by a factor of $1/d$, implemented as a straight-through identity that scales the gradient without modifying the forward value:

$$\text{damp}_d(x) = \frac{1}{d}x + \left(1 - \frac{1}{d}\right) \text{stopgrad}(x).$$

The operator is applied to $\mathrm{d}V$ at every integration step in every layer; the forward simulation is exact (the network still solves the same ODE), and the gradient flowing through that update is attenuated by a factor of $1/d$ per step, eliminating the multiplicative compounding. All experiments reported here use $d = 1000$, applied identically to the PING and COBA training pipelines. The COBA configuration is trainable at the module default $d = 80$; the PING configuration is not.

Dale's-law clamp. The synaptic matrices store conductance magnitudes, not signed currents. After each optimiser step, W_{in}, W^{EI}, and W^{IE} are therefore projected onto the non-negative cone **[52, 53]**. Excitatory or inhibitory action is set by the pathway-specific reversal potential in the membrane current: the I→E term $g_I(E_I - V)$ is hyperpolarising for the GABA reversal potential $E_I = -80$ mV. The projection permits either recurrent conductance to grow while preventing an unphysical negative conductance. The §2.4 collapse primarily reflects weakened or absent E→I recruitment and loss of inhibitory firing, not W^{IE} crossing into a negative sign.

5.5 Measurement and analysis

Mean firing rates $\langle r_E \rangle$ and $\langle r_I \rangle$ are computed as time-averaged spike counts per neuron over the presentation window. The gamma frequency f_γ is extracted as the peak of the Welch power spectral density [**54**] of the summed population E spike train at sampling rate $f_s = 1/\Delta t = 4000$ Hz, using a single segment of length equal to the trial, mean-centred (not z-scored) signal, no detrending, and a peak search restricted to the gamma band $[5, 150]$ Hz; the peak frequency is refined by parabolic sub-bin interpolation.

The rhythmicity score R is the Michelson contrast between the first side lobe and the first trough of the autocorrelation of the binned population E spike count, computed via zero-padded FFT and normalised by the squared mean of the rate. After a 3-point smoothing kernel $[0.25, 0.5, 0.25]$ is applied to the autocorrelogram, the first trough is identified as the first local minimum starting from lag 2, and the lobe as the maximum between lag 1 and the trough; then

$$R = \frac{\text{lobe} - \text{trough}}{\text{lobe} + \text{trough}} \in [0, 1).$$

R is bounded and dimensionless. We use it in preference to spectral-peak measures because the latter become unreliable at the low firing rates encountered in some of the regimes of interest. R is qualitatively consistent with the spectral-peak and population-coherence measures used elsewhere in the gamma literature [**55, 56**].

The spikes-per-cycle distribution (Figure 7) is constructed by binning each cell's spikes into gamma cycles inferred from peaks of the population I-burst rate (Gaussian-smoothed with $\sigma = 1$ ms), detected with *scipy.signal.find_peaks* using a minimum inter-peak separation of half the expected gamma period and a height threshold of 5% of the maximum. Cycle boundaries are placed at midpoints between consecutive I-burst peaks; each E spike is assigned to its enclosing cycle.

Spike-economy claims report both the mean E rate and the total population spike count $N_E\langle r_E \rangle + N_I\langle r_I \rangle$, so the reduction is stated net of the inhibitory contribution. The metabolic argument that excitatory spikes incur larger costs than inhibitory spikes [**28, 29**] is invoked where relevant but is not modelled quantitatively.

5.6 Integration and parameters

The membrane and synaptic-conductance ODEs are integrated by an exponential-Euler scheme [**57**] with zero-order hold on the synaptic conductances over each step. With $g_{\text{tot}} = g_L + g_e + g_i$ the total instantaneous conductance, effective time constant $\tau_{\text{eff}} = C_m/g_{\text{tot}}$, and instantaneous steady state $V_\infty = (g_L E_L + g_e E_e + g_i E_i)/g_{\text{tot}}$, the closed-form update is

$$V_{t+1} = V_\infty + (V_t - V_\infty)e^{-\Delta t/\tau_{\text{eff}}}.$$

Training uses $\Delta t = 0.1$ ms; smaller timesteps are required for numerical stability of the backpropagation through the recurrent E↔I dynamics. Inference uses $\Delta t = 0.25$ ms. Firing rates and frequencies are reported in Hz. The §2.6 result (Figure 10) is obtained by retraining the network at each Δt value in the swept range, and confirms invariance of training+inference at matched Δt within the range tested (not invariance of inference at a fixed-Δt-trained network to a varied inference Δt).

Default parameters used by all experiments are listed in the parameters table. Per-experiment values of the loop weights W^{EI}, W^{IE} and the input rate, and ranges swept by experiment (e.g. τ_{GABA}, W^{IE}), are stated in the corresponding figure captions.

5.7 Datasets and evaluation

The classification task is MNIST [**58**], rate-encoded as in §5.4 (200 ms presentation, 25 Hz peak Poisson rate per active pixel). MNIST is used under its standard train/test split, and test-set accuracy is reported. For the streaming protocol used in §2.7, the trained network is presented with a continuously concatenated input stream. The output-LIF membrane potentials are averaged over a trailing window and their argmax gives the streaming class label. For every segment, the readout-window duration is matched exactly to the presentation duration, $T_{\mathrm{readout}} = T_{\mathrm{presentation}} = \tau$; it is not varied independently. This explicit averaging window is distinct from the output neuron's fixed membrane time constant. The duration–rate grid uses $\tau \in \{10, 15, 25, 40, 50, 75, 100, 200\}$ ms and rates in $\{5, 10, 25, 50, 100, 200\}$ Hz. Additional evaluations hold both durations fixed at 200 ms while sweeping below 5 Hz to locate the encoder's chance floor, using the same three trained seeds. Reported metrics are test accuracy, mean E and I firing rates, and f_γ.

5.8 Reproducibility

Each reported result is produced by a standalone experiment script in the project repository (linked under §6). Each experiment hardcodes its own run scale (the sample count, seed set, and any parameter sweeps) and records those settings, together with the git commit and a run identifier, in the run's provenance file; every run number quoted in this manuscript and its captions is interpolated from those files rather than typed by hand, so a figure and the text that describes it cannot drift apart. The figure-render pipeline regenerates every print-quality figure from the experiments, so a clean re-run of the repository reproduces the figures and numbers reported here.

5.9 Software and implementation

The model, training, and analysis are implemented in Python (≥ 3.10). Spiking dynamics and surrogate-gradient training use PyTorch (2.11; with snnTorch 0.9 for baseline spiking primitives). Numerical analysis uses NumPy (2.2) and SciPy (1.15). All figures are produced with Matplotlib (3.10).

6. Code and data availability

Source code, per-notebook reproduction scripts, trained-weight artefacts, and the figure-render pipeline are available at https://github.com/eoinmurray/pinglab. MNIST is obtained from its standard public distributor. Library versions are listed in §5.9.

7. Declaration of generative-AI use

Claude Code (Anthropic; model Opus 4.8), an agentic large-language-model coding tool, was used extensively and interactively in the development of this work. Its use covers writing the simulation, training, analysis, and figure-rendering code in the project repository; debugging, refactoring, and code review; drafting, revising, and copy-editing the manuscript prose; and iteration on experimental design and presentation. No figure, illustration, table, or visualisation in this manuscript is produced by a generative AI model; all figures are rendered by Matplotlib from Python code in the project repository. The authors set the research

questions, designed the experiments and analyses, ran the simulations, reviewed model-generated code prior to commit, and verified the reported results and the manuscript text. The authors are responsible for the content, accuracy, and claims of this work.

References

1. Buzsáki & Wang — *Mechanisms of Gamma Oscillations*. 2012. doi:10.1146/annurev-neuro-062111-150444
2. Fries — *Rhythms for Cognition: Communication through Coherence*. 2015. doi:10.1016/j.neuron.2015.09.034
3. Fries, Reynolds, Rorie & Desimone — *Modulation of Oscillatory Neuronal Synchronization by Selective Visual Attention*. 2001. doi:10.1126/science.1055465
4. Gray, König, Engel & Singer — *Oscillatory Responses in Cat Visual Cortex Exhibit Inter-Columnar Synchronization Which Reflects Global Stimulus Properties*. 1989. doi:10.1038/338334a0
5. Whittington, Traub, Kopell, Ermentrout & Buhl — *Inhibition-Based Rhythms: Experimental and Mathematical Observations on Network Dynamics*. 2000. doi:10.1016/S0167-8760(00)00173-2
6. Williams et al. — *Fast Spiking Interneurons Autonomously Generate Fast Gamma Oscillations in the Medial Entorhinal Cortex with Excitation Strength Tuning ING-PING Transitions*. 2026. doi:10.1523/ENEURO.0452-25.2026
7. Börgers — *The PING Model of Gamma Rhythms*. 2017. doi:10.1007/978-3-319-51171-9_30
8. Cardin, Carlén, Meletis, Knoblich, Zhang, Deisseroth, Tsai & Moore — *Driving Fast-Spiking Cells Induces Gamma Rhythm and Controls Sensory Responses*. 2009. doi:10.1038/nature08002
9. Sohal, Zhang, Yizhar & Deisseroth — *Parvalbumin Neurons and Gamma Rhythms Enhance Cortical Circuit Performance*. 2009. doi:10.1038/nature07991
10. Phensy et al. — *Prefrontal Gamma Oscillations Engage Dynamic Cell Type-Specific Configurations to Support Flexible Behavior*. 2026. doi:10.1016/j.neuron.2026.05.002
11. Offermanns, Pöpplau & Hanganu-Opatz — *Developmental Embedding of Parvalbumin Interneurons Drives Local and Crosshemispheric Prefrontal Gamma Synchrony*. 2026.
12. Whittington, Traub & Jefferys — *Synchronized Oscillations in Interneuron Networks Driven by Metabotropic Glutamate Receptor Activation*. 1995. doi:10.1038/373612a0
13. Wang & Buzsáki — *Gamma Oscillation by Synaptic Inhibition in a Hippocampal Interneuronal Network Model*. 1996. doi:10.1523/JNEUROSCI.16-20-06402.1996
14. Bartos, Vida & Jonas — *Synaptic Mechanisms of Synchronized Gamma Oscillations in Inhibitory Interneuron Networks*. 2007. doi:10.1038/nrn2044
15. Kopell, Börgers, Pervouchine, Malerba & Tort — *Gamma and Theta Rhythms in Biophysical Models of Hippocampal Circuits*. 2010. doi:10.1007/978-1-4419-0996-1_15
16. Viriyopase, Memmesheimer & Gielen — *Cooperation and Competition of Gamma Oscillation Mechanisms*. 2016. doi:10.1152/jn.00493.2015
17. Brunel & Wang — *What Determines the Frequency of Fast Network Oscillations with Irregular Neural Discharges? I. Synaptic Dynamics and Excitation-Inhibition Balance*. 2003. doi:10.1152/jn.01095.2002

18. Wilson & Cowan — *Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons*. 1972. doi:10.1016/S0006-3495(72)86068-5
19. Segneri, Bi, Olmi & Torcini — *Theta-Nested Gamma Oscillations in Next Generation Neural Mass Models*. 2020. doi:10.3389/fncom.2020.00047
20. Nandi, Valla & di Volo — *Bursting Gamma Oscillations in Neural Mass Models*. 2024. doi:10.3389/fncom.2024.1422159
21. Tahvili, Vinck & di Volo — *A Mean-Field Model of Neural Networks with PV and SOM Interneurons Reveals Connectivity-Based Mechanisms of Gamma Oscillations*. 2026. doi:10.1371/journal.pcbi.1014378
22. Eshraghian, Ward, Neftci, Wang, Lenz, Dwivedi, Bennamoun, Jeong & Lu — *Training Spiking Neural Networks Using Lessons From Deep Learning*. 2023. doi:10.1109/JPROC.2023.3308088
23. Neftci, Mostafa & Zenke — *Surrogate Gradient Learning in Spiking Neural Networks*. 2019. doi:10.1109/MSP.2019.2931595
24. Deckers et al. — *Advancing Spatio-Temporal Processing Through Adaptation in Spiking Neural Networks*. 2025. doi:10.1038/s41467-025-60878-z
25. Yan, Yang, Wu, Liu, Zhang, Li, Tan & Wu — *Efficient and Robust Temporal Processing with Neural Oscillations Modulated Spiking Neural Networks*. 2025. doi:10.1038/s41467-025-63771-x
26. Bittar & Garner — *Exploring Neural Oscillations During Speech Perception via Surrogate-Gradient Spiking Neural Networks*. 2024. doi:10.3389/fnins.2024.1449181
27. Barth & Poulet — *Experimental Evidence for Sparse Firing in the Neocortex*. 2012. doi:10.1016/j.tins.2012.03.008
28. Attwell & Laughlin — *An Energy Budget for Signaling in the Grey Matter of the Brain*. 2001. doi:10.1097/00004647-200110000-00001
29. Howarth, Gleeson & Attwell — *Updated Energy Budgets for Neural Computation in the Neocortex and Cerebellum*. 2012. doi:10.1038/jcbfm.2012.35
30. Ainsworth, Lee, Cunningham, Traub, Kopell & Whittington — *Rates and Rhythms: A Synergistic View of Frequency and Temporal Coding in Neuronal Networks*. 2012. doi:10.1016/j.neuron.2012.06.027
31. Schaefer, Angelo, Spors & Margrie — *Neuronal Oscillations Enhance Stimulus Discrimination by Ensuring Action Potential Precision*. 2006. doi:10.1371/journal.pbio.0040163
32. Nguyen & Rubchinsky — *Temporal Patterns of Synchrony in a Pyramidal-Interneuron Gamma (PING) Network*. 2021. doi:10.1063/5.0042451
33. Shadlen & Movshon — *Synchrony Unbound: A Critical Evaluation of the Temporal Binding Hypothesis*. 1999. doi:10.1016/S0896-6273(00)80822-3
34. London, Roth, Beeren, Häusser & Latham — *Sensitivity to Perturbations in vivo Implies High Noise and Suggests Rate Coding in Cortex*. 2010. doi:10.1038/nature09086
35. Akam & Kullmann — *Efficient "Communication through Coherence" Requires Oscillations Structured to Minimize Interference between Signals*. 2012. doi:10.1371/journal.pcbi.1002760
36. Renart, de la Rocha, Bartho, Hollender, Parga, Reyes & Harris — *The Asynchronous State in Cortical Circuits*. 2010. doi:10.1126/science.1179850

37. van Vreeswijk & Sompolinsky — *Chaos in Neuronal Networks with Balanced Excitatory and Inhibitory Activity*. 1996. doi:10.1126/science.274.5293.1724
38. Vogels, Sprekeler, Zenke, Clopath & Gerstner — *Inhibitory Plasticity Balances Excitation and Inhibition in Sensory Pathways and Memory Networks*. 2011. doi:10.1126/science.1211095
39. Hennequin, Agnes & Vogels — *Inhibitory Plasticity: Balance, Control, and Codependence*. 2017. doi:10.1146/annurev-neuro-072116-031005
40. Wu, Miehl & Gjorgjieva — *Regulation of Circuit Organization and Function Through Inhibitory Synaptic Plasticity*. 2022. doi:10.1016/j.tins.2022.10.006
41. Páscoa dos Santos & Verschure — *Excitatory-Inhibitory Homeostasis and Bifurcation Control in the Wilson-Cowan Model of Cortical Dynamics*. 2025. doi:10.1371/journal.pcbi.1012723
42. Kann — *The Interneuron Energy Hypothesis: Implications for Brain Disease*. 2016. doi:10.1177/0271678X16638956
43. Börgers, Talei Franzesi, LeBeau, Boyden & Kopell — *Minimal Size of Cell Assemblies Coordinated by Gamma Oscillations*. 2012. doi:10.1371/journal.pcbi.1002362
44. Cramer, Stradmann, Schemmel & Zenke — *The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks*. 2022. doi:10.1109/TNNLS.2020.3044364
45. Tiesinga & Sejnowski — *Cortical Enlightenment: Are Attentional Gamma Oscillations Driven by ING or PING?*. 2009. doi:10.1016/j.neuron.2009.09.009
46. Brunel — *Dynamics of Sparsely Connected Networks of Excitatory and Inhibitory Spiking Neurons*. 2000. doi:10.1023/A:1008925309027
47. Gerstner — *Population Dynamics of Spiking Neurons: Fast Transients, Asynchronous States, and Locking*. 2000. doi:10.1162/089976600300015899
48. Montbrió, Pazó & Roxin — *Macroscopic Description for Networks of Spiking Neurons*. 2015. doi:10.1103/PhysRevX.5.021028
49. Zenke & Ganguli — *SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks*. 2018. doi:10.1162/neco_a_01086
50. Kingma & Ba — *Adam: A Method for Stochastic Optimization*. 2015.
51. Pascanu, Mikolov & Bengio — *On the Difficulty of Training Recurrent Neural Networks*. 2013.
52. Cornford, Kalajdzievski, Leite, Lamarquette, Kullmann & Richards — *Learning to Live with Dale's Principle: ANNs with Separate Excitatory and Inhibitory Units*. 2021.
53. Zhu et al. — *Task Success in Trained Spiking Neural Network Models Coincides with Emergence of Cross-Stimulus-Modulated Inhibition*. 2026. doi:10.1007/s00422-025-01030-4
54. Welch — *The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms*. 1967. doi:10.1109/TAU.1967.1161901
55. Atallah & Scanziani — *Instantaneous Modulation of Gamma Oscillation Frequency by Balancing Excitation with Inhibition*. 2009. doi:10.1016/j.neuron.2009.04.027
56. Xing, Shen, Burns, Yeh, Shapley & Li — *Stochastic Generation of Gamma-Band Activity in Primary Visual Cortex of Awake and Anesthetized Monkeys*. 2012. doi:10.1523/JNEUROSCI.5644-11.2012

57. Rotter & Diesmann — *Exact Digital Simulation of Time-Invariant Linear Systems with Applications to Neuronal Modeling*. 1999. doi:10.1007/s004220050570
58. LeCun, Bottou, Bengio & Haffner — *Gradient-Based Learning Applied to Document Recognition*. 1998. doi:10.1109/5.726791

The SNN tool

6 July 2026

One command-line tool, *tools/snn/tool.py*, drives every simulation, training run, and measurement in this project. Experiment runners never import the model code; they shell out to this tool, which writes data files (metrics, rasters, population traces, weight dumps) into an output directory, and then draw their own figures from those files. The tool is the engine; the plotting lives in the runners under *experiments/*.

This page is the reference for that engine. It is not a tutorial. It is the dictionary you reach for when an experiment mentions *–ei-strength 0.5* and you want to know exactly what that did.

At a glance

- Run it with *uv run python tools/snn/tool.py* (never a bare *python*; the repo convention is *uv*).
- Three subcommands: *sim* (forward pass plus metrics), *train* (surrogate-gradient BPTT), *dump-weights* (emit weight matrices).
- Every parameter is a flag. There is no global config file; a saved run's *config.json* can be inherited with *–load-config*.
- Every run writes *config.json*, *run.sh*, *output.log*, and *run.jsonl* for reproducibility.
- The tool emits data, not figures. The runner draws the figure.

The tool and the experiment

demolab draws a hard line between a **tool** and an **experiment**. A tool holds the reusable science and speaks only through files; an experiment (a runner in *experiments/expNNN.py*) chooses which tool commands to run, then reads their data files back and renders the figures. The runner reaches the tool by running its CLI as a subprocess, never by importing it. That firewall is what keeps *tools/snn/* generic and lets the same engine serve every writeup in the lab.

So a typical experiment does three things: it invokes *tool.py* (often many times, sweeping a parameter), it aggregates each run's config plus headline metrics into a single *numbers.json*, and it draws PNG or SVG figures from the run's data. The committed record lands in *artifacts/data/expNNN/*; the tool's own scratch output is disposable (see Artifacts).

Quick start

Run a metrics-only forward pass of the canonical PING network:

```
uv run python tools/snn/tool.py sim --model ping --ei-strength 0.5
```

Train on MNIST for 100 epochs:

```
uv run python tools/snn/tool.py train --dataset mnist --epochs 100 --lr
0.0001
```

Evaluate a trained run on the test set, replaying it at a coarser timestep:

```
uv run python tools/snn/tool.py sim --infer \
  --load-config runs/foo/config.json \
  --load-weights runs/foo/weights.pth \
  --dt 0.5
```

Each subcommand has its own complete *–help*, and that per-subcommand help is authoritative. The top-level dispatcher's group summary is hand-maintained and can lag the parser. When a flag here disagrees with reality, run *sim –help* and trust that.

Commands

sim

Run one forward pass and report firing-rate metrics. This is the cheapest mode (no training, no plots), used by the test suite, the dt-stability checks in ar003, and every experiment that needs a single inference pass over a trained or fresh network.

```
uv run python tools/snn/tool.py sim --model ping --dataset mnist --digit 3
```

On its own it prints metrics. The flags below make it load trained weights, evaluate a test set, emit extra data artifacts, or inject perturbations. There is no *–image* or *–video*: where those retired flags once produced panels and sweep MP4s, *sim* now emits raw data (via *–outputs*) that the calling runner plots, and sweep videos are assembled runner-side from many *sim* calls.

flag	default	description
–infer	off	Load trained weights and evaluate test-set accuracy; writes *results.json* (and *metrics.json*).
–load-config PATH	—	Load a saved *config.json* and inherit its model, dataset, and parameters. Explicit CLI flags override loaded values.
–load-weights PATH	—	Path to a *weights.pth* file for inference.
–max-samples INT	all	[*–infer*] Cap the evaluation set to N samples.
–outputs OUTPUT [dots.c]	—	[*–infer*] Extra artifacts from the single forward pass (*metrics.json*

		always written): *per_cell_rates* (per-cell E/I Hz to *per_cell_rates.npz*), *pop_traces* (per-trial population activity to *pop_traces.npz*, base signal for PSD / f_γ), *rasters* (sparse spike indices to *rasters.npz*, for cycle-level analysis).
–tau-gaba FLOAT	inherited / 9.0	[*–infer*] Override τ_GABA (ms) to replay a trained cell under specified inhibitory dynamics. Normally unset: *–load-config* inherits the trained value.
–skip-load PREFIX [dots.c]	—	[*–infer*] Drop *state_dict* keys with these prefixes before loading (e.g. *W_ei. W_ie.*) so a fresh sub-block survives. Transfer-load probes (exp038).
–perturb-mode {drop, add}	—	[*–infer*] Hidden-spike perturbation inside the forward loop: *drop* (Bernoulli mask), *add* (Poisson Hz). The exp037 drop/add asymmetry.
–perturb-level LEVEL [dots.c]	—	[*–perturb-mode*] One value: probability for *drop*, Hz for *add*.
–i-override-file PATH	—	[*–infer*] NPZ with a sparse per-trial I-spike stream to substitute for the inhibitory spikes each timestep. Injection dual of *–outputs rasters* (exp042).

–input-file PATH	—	NPZ with *input_spikes* (T, B, N_IN) to forward instead of Poisson input. Arbitrary stimulus (exp048 digit streams).
–scale-w-in / –scale-w-ei / –scale-w-ie FLOAT	1.0	[*–infer*] Multiply loaded input / E→I / I→E weights before the forward pass. Inference-time coupling sweeps without retraining (exp038).
–sample-index INT	—	Raw test-set index for a snapshot, overriding *–digit / –sample*.
–n-in / –n-inh / –n-batch INT	784 / — / 64	[synthetic-spikes] Input channels, inhibitory pool size, Poisson trials averaged.
–w-ei-mean / –w-ie-mean FLOAT	from *–ei-strength*	[synthetic-spikes] Explicit W_EI / W_IE mean (std = 0.1 · mean).
–private-w-in	off	[synthetic-spikes] Identity W_in: one input channel per E cell.

The block from *–skip-load* down is the **generic-primitive family**: small, experiment-agnostic hooks (perturb hidden spikes, inject an inhibitory stream, forward an arbitrary input file, scale a weight block at inference). Runners compose these instead of importing model code, and that is what keeps the tool/experiment boundary clean.

train

Surrogate-gradient BPTT training loop. Writes *weights.pth*, *metrics.json*, a per-step *metrics.jsonl*, and *test_predictions.json*.

```
uv run python tools/snn/tool.py train --model ping --dataset mnist \
  --epochs 100 --lr 0.0001 --v-grad-dampen 1000
```

–epochs 0 runs the init snapshot only, useful as a probe.

flag	**default**	**description**

–lr FLOAT	0.01	Adam learning rate. Biophysical models (ping) typically need 0.0001; current-based models 0.01.
–epochs INT	0	Number of epochs. 0 = init-snapshot probe only.
–batch-size INT	64	DataLoader batch size.
–max-samples INT	all	Cap dataset to N samples for smoke tests.
–v-grad-dampen FLOAT	80.0	Dampening factor d on the COBA membrane-voltage gradient. PING needs a much larger value (d = *1000* in the paper) to stabilise BPTT through the E↔I loop; COBA trains at the default. Theory in ar006.
–fr-reg-upper-theta FLOAT	0 (off)	Upper-bound target spike count per neuron per trial (θ_u). Adds $s\_u \cdot \Sigma\ relu(\langle z\_i \rangle - \theta\_u)^2$ to the loss. Cramer et al. SHD RSNN: 100.
–fr-reg-upper-strength FLOAT	0	Coefficient s_u on the upper regulariser. Cramer et al.: 0.06.
–tau-gaba FLOAT	9.0 ms	Override τ_GABA. Default = *models.py*'s value (Börgers / Buzsáki-Wang range). exp041 sweeps this across {4.5 … 27} ms while training PING from scratch; the realised gamma frequency f_γ tracks 1/τ_GABA.

dump-weights

Build the network from a config and emit its weight matrices to *weights_dump.npz*: the init state, plus (with *–load-weights*) the trained state. It runs no forward pass.

```
uv run python tools/snn/tool.py dump-weights \
  --load-config runs/foo/config.json \
  --load-weights runs/foo/weights.pth \
  --out-dir runs/foo/dump
```

Keys follow *W_ff_N_init* / *W_ff_N_trained* (feedforward, per layer N) plus the E-I blocks *W_ei* / *W_ie*. This is how a runner recovers the trained readout matrix (W_out = the last *W_ff*) or compares init-vs-trained loop weights (the exp049 pruning analysis) without loading the model in-process. It takes the shared option groups plus *–load-config* / *–load-weights*.

Shared options

These groups are attached to every subcommand. The grouping matches what each subcommand's *–help* prints.

Network

flag	default	description
–model {ping}	*ping*	Architecture. *ping* is the COBANet with E↔I coupling; with *–ei-strength 0* the

		inhibitory loop is silenced for a no-rhythm control.
–n-hidden INT [INT dots.c]	dataset-dependent	Hidden layer sizes. One integer = single layer; multiple stacks layers. Default for mnist: 1024.
–readout {rate, mem-mean}	*rate*	Output stage. *rate* sums last-hidden spikes and projects linearly at the final timestep. *mem-mean* averages a per-class output-LIF membrane over time (the trained classification readout).
–dales-law / –no-dales-law	on	Enforce Dale's law (non-negative weights) or allow signed weights. *–no-dales-law* is used for balanced-network experiments.
–ei-strength FLOAT	0.5	E-I coupling strength s. Sets W_EI = s and W_IE = s · ratio.
–ei-ratio FLOAT	2.0	W_IE / W_EI.
–w-in-sparsity FLOAT	0.95	Fraction of input weights zeroed at init.
–ei-sparsity FLOAT	0.0	Sparsity of the recurrent E↔I matrices: fraction of entries zeroed, survivors rescaled by 1/(1−s) to preserve expected drive. Use ≈ 1 − K/N for Brunel/Vreeswijk sparse random connectivity.
–exact-k	off	Fixed-fan-in (exact-K) recurrent connectivity: every post cell draws exactly K = round((1−ei_sparsity) · N_pre) inputs. No effect unless *–ei-sparsity* > 0.
–dt FLOAT	0.25	Integration timestep (ms).
–t-ms FLOAT	200	Total trial duration (ms). Metrics are measured over the full trace; runners strip any startup transient in post.
–readout-w-out-scale FLOAT	1.0	Scalar applied to the readout matrix after *build_net*, compensating for low hidden firing rate under *mem-mean*. Train-mode only.
–surrogate-slope FLOAT	1.0	Fast-sigmoid surrogate-gradient slope β. Larger = narrower active window. Cramer et al. use 40 for SHD RSNNs.

The drive family below also lives in the Network group. It exists for the balanced-network (Brunel / van Vreeswijk-Sompolinsky) experiments and the Lyapunov chaos probe; canonical PING runs leave all of it off.

flag	default	description
–independent-drive RATE G_PER_SPIKE	off	Per-E-cell independent Poisson drive (bypasses W_in): N_E uncorrelated streams at RATE Hz, each spike adding G_PER_SPIKE μS of g_E. Zero cross-cell correlation.
–independent-drive-i RATE G_PER_SPIKE	off	As above, targeting the I population directly. Needed for the full V&S asynchronous-irregular state.
–quenched-drive MEAN STD	off	Per-E-cell DC conductance drawn once from N(MEAN, STD) μS and frozen for the trial. V&S quenched input: no fluctuation, so it cannot pin spike times; the Lyapunov probe then measures autonomous chaos.
–quenched-drive-i MEAN STD	off	Per-I-cell frozen DC conductance.
–lyapunov-eps FLOAT	0 (off)	If > 0 (synthetic-spikes mode), rerun with all membranes ε-perturbed at t=0 and save the divergence $\|\Delta V(t)\|$ to *snapshot.npz*. Its growth rate is the max Lyapunov exponent: positive for the chaotic V&S state, ≈ 0 for cycle-locked PING.

Input

flag	default	description
–input {synthetic-spikes, dataset}	*synthetic-spikes*	Stimulus regime. *synthetic-spikes* is Poisson at *–input-rate*. *dataset* draws from *–dataset*.
–input-rate FLOAT	25	Baseline input rate (Hz).
–digit INT	0	Dataset class (0–9).
–sample INT	0	Sample index within the class.
–sample-index INT	—	Raw test-set index, overriding *–digit* / *–sample*.

–dataset {mnist}	*mnist*	Dataset. *mnist* is the full 28×28 image encoded to spikes.

Weights (advanced)

flag	default	description
–w-in MEAN [STD]	*0.3 0.06*	Input fan-in init. Single value sets STD = MEAN × 0.1.
–w-ei MEAN STD	from *–ei-strength*	Override the W_EI init.
–w-ie MEAN STD	from *–ei-strength / –ei-ratio*	Override the W_IE init.
–w-ii MEAN STD	*0 0*	W_II (I→I) init. Off by default (canonical PING has no I→I). Enable for balanced-network experiments.
–w-ee MEAN STD	*0 0*	W_EE (E→E) init. Off by default. Enable for the full four-coupling balanced network, where recurrent excitation pins the E rate.
–trainable-w-ei	frozen	Promote E→I to gradient-carrying. Asks whether the optimiser will discover the PING-loop weights from scratch.
–trainable-w-ie	frozen	Promote I→E. The exp049 result *gradient descent dismantles PING* comes from flipping *–trainable-w-ei* and *–trainable-w-ie* on simultaneously.

Output

flag	default	description
–out-dir DIR	*temp/pinglab-cli/*	Output directory. The default is scratch (gitignored); runners always pass an explicit path.
–wipe-dir	off	Clear the output directory before the run.

Execution

flag	default	description
–seed INT	—	RNG seed. Seeds Python, NumPy, torch (CPU + CUDA + MPS) before dataset load and model init. Persisted to *config.json*.
–modal	off	Re-dispatch to Modal.com. Artifacts sync back to *–out-dir* after completion.

–modal-gpu {none, T4, L4, A10G, A100, H100}	*T4*	GPU type for Modal runs. *none* runs CPU-only.

–modal costs money. The project default is local; only pass it when explicitly instructed.

Config inheritance

The trick that makes the experiment chain work is *–load-config*. Every *train* run writes a *config.json* alongside its *weights.pth*; a later *sim* or *dump-weights* run inherits from it:

```
uv run python tools/snn/tool.py sim --infer \
  --load-config runs/foo/config.json \
  --load-weights runs/foo/weights.pth \
  --dt 0.5
```

This inherits the model, hidden sizes, dataset, E-I parameters, input rate, τ_GABA, and seed, while the explicitly-passed *–dt 0.5* overrides the trained value, replaying the network at a new timestep. Precedence is: explicit CLI flag, then loaded config, then default. The parser builds the set of CLI-explicit flags from *sys.argv* before applying inheritance.

Backwards compatibility: old configs that stored *n_hidden* as a scalar are remapped to the *hidden_sizes* list, legacy model names are aliased with a one-line stderr note, and configs missing *dales_law* trigger a warning to pass it explicitly or retrain.

Artifacts

Every subcommand calls *save_run_artifacts* on entry, writing four provenance files into *–out-dir*:

file	contents
config.json	The parsed argparse namespace plus a provenance block (*git_sha* with a *dirty* suffix, *torch_version*, *device*, *python_env_hash*, *run_id*, *started_at*) and the *mode*. Consumed by *–load-config* and the runner's metadata extractors.
run.sh	The literal *sys.argv* joined with spaces and prefixed with a shebang. Re-running reproduces the run (modulo seeded RNG state, also in *config.json*).
output.log	The human run log. ANSI escapes are stripped from the file but preserved on stdout, so terminals see colour while the log stays grep-friendly.
run.jsonl	The machine-readable event spine: one typed JSON object per event (epoch rows, warnings, summary). This is what a runner parses when it wants structured progress, not scraped log text.

Each command adds its own outputs: *train* writes *weights.pth*, *metrics.json*, *metrics.jsonl*, *test_predictions.json*; *sim –infer* writes *metrics.json* (and *results.json*) plus whatever *–outputs* requested; *dump-weights* writes *weights_dump.npz*.

These files are scratch. By default they land under *temp/pinglab-cli/*, which is gitignored and overwritten every run. The committed record is produced by the runner: it aggregates each command's config plus headline metrics into one *artifacts/data/expNNN/numbers.json* and

renders its figures alongside. That folder, not the ephemeral tool output, is what the publisher reads and what reaches the site.

Recipes

Train, then measure the trained network. Train writes a run directory; *sim –infer* reads it back and emits the population traces a runner needs for a PSD:

```
uv run python tools/snn/tool.py train --dataset mnist --epochs 100 \
  --lr 0.0001 --v-grad-dampen 1000 --out-dir runs/ping

uv run python tools/snn/tool.py sim --infer \
  --load-config runs/ping/config.json \
  --load-weights runs/ping/weights.pth \
  --outputs pop_traces per_cell_rates
```

Loop-transfer at inference (exp038). Take a network trained as COBA and scale its E→I coupling up at inference, with no retraining:

```
uv run python tools/snn/tool.py sim --infer \
  --load-config runs/coba/config.json \
  --load-weights runs/coba/weights.pth \
  --scale-w-ei 1.0 --scale-w-ie 1.0 --outputs rasters
```

Perturbation sweep (exp037). Drop a fraction of emitted spikes, or add off-phase Poisson noise, inside the forward loop:

```
uv run python tools/snn/tool.py sim --infer \
  --load-config runs/ping/config.json --load-weights runs/ping/weights.pth \
  --perturb-mode drop --perturb-level 0.8
```

Recover the trained readout matrix. Dump weights and read *W_ff_N_trained* (the last layer is W_out):

```
uv run python tools/snn/tool.py dump-weights \
  --load-config runs/ping/config.json \
  --load-weights runs/ping/weights.pth --out-dir runs/ping/dump
```

Gradient Stabilisation

12 June 2026

Conductance-based spiking networks (COBA, PING) need one extra ingredient to train at all: a single flag, `--v-grad-dampen`. This article is the deep dive on why it is needed and what it does, derived from the discrete equations the code runs. For the surrounding recipe — BPTT, the spike surrogate, the loss, optimiser, readout, and regulariser — see the Training article; this one assumes that background.

In plain English. The E and I cells form a loop that makes the network oscillate at gamma (≈25 Hz). Training sends the gradient backward around that same loop, and each time a neuron crosses its spike threshold the surrogate hands the gradient a large multiplier. The loop is traversed *once per gamma cycle* — only about five times in a 200 ms trial — but each pass multiplies by a factor well above one (the surrogate slope enters squared), so a handful of passes is enough to overflow to NaN. The forward simulation stays bounded because the spike reset clamps each cell every cycle; the gradient sidesteps that reset, so forward stability buys nothing backward. The fix, `--v-grad-dampen`, divides the voltage gradient by a constant γ at every step, shrinking the loop's multiplier below one — and thanks to a straight-through trick it changes only the gradient, never the simulation. The rest of this article proves each of those claims from the discrete equations.

Naive BPTT through a 2000-step COBA/PING trial reaches NaN within a few batches. This article establishes the failure and its remedy from the discrete equations the code runs — no step omitted. The plan is: write the one-step map, differentiate it exactly (Lemma 1), propagate the gradient backward (the recursion), show the recurrent loop forces that gradient to diverge geometrically (Proposition 1), and show that scaling the per-step voltage gradient by $1/\gamma$ makes the loop a contraction while leaving the forward trajectory untouched (Proposition 2).

Setup: the discrete update

Index the timesteps $t = 0, \ldots, T$ at $\Delta t = 0.1$ ms. Each cell holds a state (V^t, g_e^t, g_i^t). With reversal potentials E_L, E_e, E_i, capacitance C_m, leak g_L, synaptic decays $\beta_e = e^{-\Delta t/\tau_e}$ and $\beta_i = e^{-\Delta t/\tau_i}$, threshold ϑ and reset V_r, the step is exactly (`lif_step_expeuler`, `exp_synapse`):

$$s^t = \Theta(V^t - \vartheta)\chi^t, \quad \text{(S)}$$

$$g_e^{t+1} = \beta_e\big(g_e^t + W_e s_{\text{pre}}^t\big), \quad g_i^{t+1} = \beta_i\big(g_i^t + W_i s_{\text{pre}}^t\big), \quad \text{(C)}$$

$$\tilde{V}^{t+1} = V_\infty^{t+1} + (V^t - V_\infty^{t+1})\alpha^{t+1}, \quad V^{t+1} = \begin{cases} V_r & \text{if } \ s^t = 1 \ \text{ or refractory} \\ \tilde{V}^{t+1} & \text{otherwise,} \end{cases} \quad \text{(V)}$$

where $\chi^t \in \{0, 1\}$ is the not-refractory gate, and the membrane decay and rest point are built from the *post*-update conductances g^{t+1}:

$$g_{\text{tot}} = g_L + g_e^{t+1} + g_i^{t+1}, \quad \alpha^{t+1} = e^{-\Delta t g_{\text{tot}}/C_m}, \quad V_\infty^{t+1} = \frac{g_L E_L + g_e^{t+1} E_e + g_i^{t+1} E_i}{g_{\text{tot}}}. \quad \text{(P)}$$

The timing matters and is taken from the code: the spike s^t in (S) is read from the *current* voltage V^t, drives the conductance one step later in (C), and that conductance sets the

voltage in (V). So one synaptic edge — presynaptic voltage to postsynaptic voltage — spans one timestep.

Lemma 1 — the one-step Jacobian

Differentiate (S), (C), (V) entry by entry. The spike surrogate gives the spike derivative

$$\frac{\partial s^t}{\partial V^t} = \sigma'(V^t - \vartheta)\chi^t, \quad \sigma'(u) = \frac{k}{(1 + k\ |u|)^2}, \quad \sigma'(0) = k. \quad (1)$$

From (C), the conductance partials are a self-decay and a *presynaptic-voltage* coupling obtained by chaining (1):

$$\frac{\partial g_e^{t+1}}{\partial g_e^t} = \beta_e, \quad \frac{\partial g_e^{t+1}}{\partial V_{\text{pre}}^t} = \beta_e W_e \sigma'\big(V_{\text{pre}}^t - \vartheta\big)\chi_{\text{pre}}^t. \quad (2)$$

From (V), with g held, the membrane self-term is the decay; and differentiating (P) gives the conductance-to-voltage term ($\partial g_{\text{tot}}/\partial g_e = 1$, $\partial V_\infty/\partial g_e = (E_e - V_\infty)/g_{\text{tot}}$, $\partial \alpha/\partial g_e = -(\Delta t/C_m)\alpha$):

$$\frac{\partial \tilde{V}^{t+1}}{\partial V^t} = \alpha^{t+1}, \quad \frac{\partial \tilde{V}^{t+1}}{\partial g_e^{t+1}} = \underbrace{(1-\alpha)\frac{E_e - V_\infty}{g_{\text{tot}}}}_{\text{rest shifts}} - \underbrace{\frac{\Delta t}{C_m}\alpha(V^t - V_\infty)}_{\text{decay shifts}} \equiv \kappa_e \approx \frac{\Delta t}{C_m}(E_e - V), \quad (3)$$

the last step being the leading order in Δt (using $1 - \alpha \approx \Delta t g_{\text{tot}}/C_m$). Define κ_i identically with E_i. Collecting (1)–(3), the one-step Jacobian on (V, g_e, g_i), with the cross-cell coupling in the lower-left block, is

$$J^t = \frac{\partial (V, g_e, g_i)^{t+1}}{\partial (V, g_e, g_i)^t} = \begin{pmatrix} \rho\alpha & \kappa_e & \kappa_i \\ \beta_e W_e \sigma'\big(V_{\text{pre}} - \vartheta\big)\chi_{\text{pre}} & \beta_e & 0 \\ \beta_i W_i \sigma'\big(V_{\text{pre}} - \vartheta\big)\chi_{\text{pre}} & 0 & \beta_i \end{pmatrix}. \quad (\text{J})$$

The single subtlety, and it is decisive below: the reset in (V) is a `torch.where`, so on any cell that spikes or is refractory the output is the constant V_r and the membrane self-term is gated to zero. We write this as the factor $\rho \in \{0, 1\}$ on the $\partial V^{t+1}/\partial V^t$ entry: $\rho = 0$ at a spike (gradient through the cell's own membrane is cut), $\rho = 1$ otherwise. Crucially ρ multiplies only the membrane self-term — it does **not** touch the spike output s^t in (1)–(2), which is evaluated at V^t *before* the reset.

Backpropagation: the gradient recursion

Let $\lambda^t \equiv \partial \mathcal{L}/\partial (V, g_e, g_i)^t$. Reverse-mode autodiff is exactly the linear recursion

$$\lambda^t = \big(J^t\big)^\top \lambda^{t+1}, \quad \lambda^T = \nabla_{x^T}\mathcal{L} \quad \Longrightarrow \quad \lambda^t = \left(\prod_{s=t}^{T-1} J^s\right)^\top \lambda^T. \quad (4)$$

So the gradient that reaches step t is governed by the product of one-step Jacobians, and $\|\lambda^t\| \le \big(\prod_s \|J^s\|\big)\|\lambda^T\|$. Whether this is benign or catastrophic is decided by the voltage component of (J) chained through the recurrent wiring.

Proposition 1 — the backpropagated gradient diverges (the problem)

Claim. In a network with a recurrent E→I→E loop, the voltage gradient grows geometrically in the number of gamma *cycles* traversed — not in the number of timesteps. A 200 ms trial holds only $N \approx 5$ cycles against $T = 2000$ steps; the danger is that each cycle multiplies by a large factor, not that there are many cycles.

Proof. Compose (2) and (3): the gradient carried from a postsynaptic voltage at $t\{+\}1$ to a presynaptic voltage at t across one synapse is the product of the conductance-coupling and the membrane term,

$$a \equiv \frac{\partial V_{\text{post}}^{t+1}}{\partial V_{\text{pre}}^{t}} = \underbrace{\kappa}_{\approx \frac{\Delta t}{C_m}(E_{\text{syn}} - V)} \cdot \underbrace{\beta W}_{\text{synapse}} \cdot \underbrace{\sigma'\big(V_{\text{pre}} - \vartheta\big)}_{\text{spike}}. \quad (5)$$

Traverse the loop once: $V_E \to V_I$ across W_{ei} (edge gain a_{ei}), then $V_I \to V_E$ across W_{ie} (edge gain a_{ie}). Over one round trip the E-voltage gradient maps to itself with the **loop gain**

$$\varrho = a_{ei} a_{ie} = \underbrace{k^2}_{\substack{\text{two } \sigma' \\ \text{at volley}}} \cdot \beta_e \beta_i W_{ei} W_{ie} \kappa_I \kappa_E. \quad (6)$$

The factor k^2 appears **once per gamma cycle**. Each population fires a single synchronous volley per cycle, so the two large kicks $\sigma' \to \sigma'(0) = k$ (one for E, one for I) occur together once per loop traversal and nowhere else; between volleys the per-step Jacobians are the mild sub-unit decays $\alpha, \beta < 1$, which merely carry the gradient along without amplifying it. So (4) collapses, across cycles, to the scalar recursion $\lambda_{V,E}^{(n)} \approx \varrho \lambda_{V,E}^{(n+1)}$, giving after N cycles

$$|\lambda_{V,E}| \geqslant |\varrho|^N \, |\lambda_{V,E}^{T}|. \quad (7)$$

Worked example (default PING init). Take the released constants: $\Delta t = 0.1$ ms, $k = 5$, $C_m^E = 1.0$ nF, $C_m^I = 0.5$ nF, $E_e = 0$, $E_i = -80$ mV, $V \approx \vartheta = -50$ mV at the crossing, $\tau_{\text{AMPA}} = 2$ ms ($\beta_e = e^{-0.05} \approx 0.95$), $\tau_{\text{GABA}} = 9$ ms ($\beta_i \approx 0.99$), and order-µS coupling $W_{ei} \approx 1$, $W_{ie} \approx 2$ µS. The two edge gains (5) are

$$a_{ei} = \underbrace{\frac{0.1}{0.5}(0 - (-50))}_{\kappa_{\to I} = 10} \cdot \underbrace{0.95 \cdot 1}_{\beta_e W_{ei}} \cdot \underbrace{5}_{k} \approx 48,$$

$$a_{ie} = \underbrace{\frac{0.1}{1.0}(-80 - (-50))}_{\kappa_{\to E} = -3} \cdot \underbrace{0.99 \cdot 2}_{\beta_i W_{ie}} \cdot \underbrace{5}_{k} \approx -30,$$

so the loop gain is $\varrho = a_{ei} a_{ie} \approx -1.4 \times 10^3$. Over the $N \approx 5$ cycles of a 200 ms trial — not the $T = 2000$ steps — the voltage gradient is amplified by

$$|\varrho|^N \approx \left(1.4 \times 10^3\right)^5 \approx 6 \times 10^{15}.$$

A unit gradient seeded at the readout thus returns to $t = 0$ scaled by $\sim 10^{16}$; summed over the batch and compounded across successive optimiser steps it crosses the fp32 ceiling ($\approx 3.4 \times 10^{38}$) and the loss becomes NaN within a few batches. The blow-up is robust: even if threshold spread cuts the effective σ' tenfold (each edge $\div 10$, so $\varrho \div 100$), $|\varrho| \approx 14$ and $|\varrho|^5 \approx 6 \times 10^5$ — still divergent. The compounding is per cycle, not per step. ■

Why the forward pass does not blow up the same way. The forward orbit is bounded because the reset (V) slams each spiking cell to V_r every cycle — that is the $\rho = 0$ gate in (J), a strong per-cycle contraction. But ϱ in (6) is built only from the spike-output edges (5), i.e. from σ' evaluated *before* the reset; the gate ρ sits on the membrane self-term and never enters the loop product. So the backward loop bypasses precisely the contraction that bounds the forward orbit. Forward stability and backward divergence are not in contradiction: they travel different paths through (J), and the straight-through reset is what separates them.

Proposition 2 — per-step voltage-gradient damping bounds it (the solution)

The flag `--v-grad-dampen` inserts, before the reset, the operation `dv = _scale_grad(dv, 1/γ)` where $dv = (V_\infty - V)(1-\alpha)$ is the membrane increment in (V). The primitive

$$\text{\_scale\_grad}(x, c) = cx + \text{detach}(x)(1-c) \quad (8)$$

is the identity in the forward pass ($cx + (1\{-\}c)x = x$) but multiplies the backward gradient through x by $c = 1/\gamma$. Re-differentiating (V) with dv so scaled changes two partials of (J):

$$\frac{\partial V^{t+1}}{\partial V^t} = 1 - \frac{1-\alpha}{\gamma} \in [\alpha, 1], \quad \frac{\partial V^{t+1}}{\partial g^{t+1}} = \frac{\kappa}{\gamma}. \quad (9)$$

The membrane self-term stays bounded by 1 (for $\gamma \to \infty$ it tends to 1 — the slow integration pathway is *preserved*, not crushed), while every voltage←conductance edge — and therefore every loop edge (5) — is divided by γ. The loop gain (6) becomes

$$\varrho_\gamma = \frac{a_{ei}}{\gamma}\frac{a_{ie}}{\gamma} = \frac{\varrho}{\gamma^2}, \quad (10)$$

so the backward loop is a contraction, $|\varrho_\gamma| \le 1$, as soon as

$$\gamma \ge \sqrt{|\varrho|}. \quad (11)$$

By (8) the forward trajectory x^t is bitwise identical with or without the flag, so this is a pure modification of the gradient, not of the dynamics. ■

In code this is the single line `_scale_grad(dv, 1.0 / v_grad_dampen)` in the LIF step. Since $|\varrho| \sim k^2 c$ for an order-unity loop constant c, the threshold (11) scales like $\sqrt{|\varrho|} \sim k\sqrt{c}$, consistent with the recipes: $\gamma \approx 80$ for the unitless standard SNN and $\gamma \approx 1000$ for COBA/PING (used by exp025 and downstream), whose larger conductance-scale loop constant demands the larger γ.

Corollary — the cost, and choosing γ

The $1/\gamma$ in (9) lands on *every* voltage←conductance gradient, not only the recurrent loop. The feedforward input also enters through g_e (the term $W_{\text{in}} s_{\text{in}}$ in (C)), so the input-weight gradient is suppressed by the same factor: damping trades a slice of the legitimate learning signal for stability. Hence the operating rule — take the smallest γ satisfying (11), i.e. the smallest value that prevents overflow. Too large a γ can therefore quietly cap achievable accuracy by starving the input layer of gradient. Distinct from gradient clipping, which rescales the assembled parameter gradient after the fact: damping reshapes the recursion (4)

term by term, before any parameter gradient is formed. Tightening (11) per-layer on long-trial tasks is open work.

Prediction. The mechanism implies the flag is load-bearing only when the loop exists: the same network should train with damping fully off when run as COBA ($W_{ei} = 0$, loop open), but fail as PING ($W_{ei} > 0$) — every optimiser step's gradient going non-finite and being skipped, leaving the network frozen at chance.

The Functional Role of PING: mapping the space

12 July 2026

Abstract

This article is a *corpus*, not a review. It is an attempt at an exhaustive, auditable paper list for one narrow question, leaving the synthesis, the open debates, and the gaps to a later pass. The question:

> In which papers is the PING mechanism explicitly the vehicle for a functional claim?

PING here means gamma generated by the reciprocal pyramidal↔interneuron loop with its characteristic excitatory–inhibitory timing. A paper is in the corpus only if it both invokes that mechanism as the generator of the rhythm *and* argues that a cognitive or computational function follows from it: attention, stimulus selection, communication-through-coherence, gain control and coincidence windows, phase coding, or binding. Papers where gamma merely *correlates* with a function, or where the mechanism is studied with no functional claim, are out; so are critiques that attack a function without naming the mechanism, though the major ones are collected in the appendix so the eventual review can engage them. The scope was frozen and git tagged (`freeze-ar016-v1`) before any measured run, so the recall figure below is valid: the completeness estimate compares samples of one fixed population, which a scope that drifted mid-run would destroy.

Papers in the corpus	160, every DOI verified
Estimated true size	≈ 180–250 (two-source Chapman 246, 95% CI 175–316)
Estimated recall	≈ 65–90 %
Channels	memory · keyword-search · citation-graph · embedding
Membership authority	one two-stage judge (wide-net → confirm), consistent across channels
Coverage caveats	keyless (no Semantic Scholar / PubMed); single judge per paper; OpenAlex abstract gaps on seminal papers

The estimate is a range, not a point. Completeness is measured by *capture–recapture* (the Lincoln–Petersen / Chapman estimator): when two channels independently sample one fixed population, the size of their overlap implies how many papers both missed, and so the true size $\hat{N}$. Computed over the three channel-pairs that overlap, it yields three estimates, and they disagree: 246 (memory × search), 219 (search × citation), and 142 (memory × citation). The disagreement is expected rather than anomalous: the channels have heterogeneous catchability (memory reaches the old and seminal, search the recent and well-indexed), which violates the independence the estimator assumes. No single $\hat{N}$ is reliable, so we report the band 180–250 and a recall of 65–90 % rather than a point value.

Methods

The model can estimate the size of a literature (it can tell hundreds from thousands) and judge whether a specific paper is in scope. Its memory is also a recall channel: unreliable in a single readout, where it is lossy and confabulates at the margin, but usable once multiplexed

across many decorrelated passes and filtered through DOI verification. Recall is gathered from four channels (memory among them), membership is decided by judgment, and completeness is estimated statistically. The procedure, in order:

1. **Refine and scout the question, before anything is frozen.** The raw interest ("the big questions about PING") is too broad to search, so it is narrowed conversationally: the model reflects the corpus size back from memory (gamma broadly is thousands of papers; the mechanism-carries-function slice is hundreds), weighs narrowing and widening axes by their size, and converges on 4 anchors with explicit in/out boundaries, targeting 100–400. A cheap scout pass then probes live database counts to confirm the slice is exhaustible and expose traps: the mechanism core returns 100–510 hits against the "communication through coherence" balloon (5878 hits) the scope must exclude, and the coincidence/gain axis is nearly invisible to keywords (6 hits), needing concept-level queries later. It also flags collisions like "PING" naming a network tool. The result is the scope proposal, estimated at 100–250, which is then frozen.
2. **Freeze the scope.** The inclusion rubric is fixed and git tagged (`freeze-ar016-v1`) before any measured run, so both capture samples target one fixed population and the recall figure is meaningful.
3. **Gather recall from four orthogonal channels,** chosen to fail on different papers so their union covers what any one misses:
   - **Memory / weights:** 20 decorrelated elicitation passes (per-lab, per-function-axis, per-decade), each in its own context, unioned. This channel is the knowledge baked into the model's trained weights. Covers the old, seminal, abstract-less stratum.
   - **Keyword search:** a deep OpenAlex full-text sweep across the frozen synonym ring (the fixed set of the concept's synonyms and near-terms, agreed at scope-freeze), deliberately high-recall. Covers the recent, well-indexed stratum.
   - **Citation graph:** the backward references and forward citers of the found set, scored by how many of the corpus each candidate connects to. Covers the well-connected stratum, independent of words and of memory.
   - **Embedding:** nearest neighbours of the strongest seeds in SPECTER (a document-embedding model, served by Semantic Scholar, that places papers by learned content similarity), independent of shared words, citations, and authorship.
4. **Verify every candidate DOI against Crossref.** An unresolvable DOI is treated as a hallucination and discarded, so nothing enters the corpus that cannot be resolved to a real paper.
5. **Judge membership with one consistent two-stage judge:** a wide-net first pass, then a strict confirming pass against the frozen rubric, so the in/out boundary is one authority applied uniformly. Membership is *not* decided by any keyword filter. An earlier attempt to do so failed, because a filter strict enough to exclude the broad "gamma and attention" literature also excludes the function-forward canonical papers whose abstracts never say "interneuron".

6. **Estimate completeness by capture–recapture.** Lincoln–Petersen / Chapman over the single judged population, treating each channel as an independent capture, gives the true-size band and the recall figure.

Hallucination and verification

Because a language model generates the memory channel, fabricated references are the obvious failure mode. Every model-proposed reference passes a two-layer DOI check: each generating pass self-verifies against Crossref and drops what will not resolve, then an independent re-check resolves the deduplicated union again. In a controlled pilot where the model gave author, year, and title (not DOIs) and the resolver did the lookup, all 44 of 44 recalled papers were real, resolvable works. In the full harvest the 20 memory passes returned 420 candidates, deduplicated to 180 references, of which 0 failed to resolve at the re-check.

The caveat is that DOI-resolution catches an *invented identifier* but not a *valid DOI attached to the wrong paper*. That residual is real: a later audit removed 1 reference whose DOI resolved only to a journal, and corrected 4 records with valid DOIs but corrupt third-party metadata (a prostate-cancer note, a Japanese architecture paper). Resolution is necessary but not sufficient; a Crossref-versus-OpenAlex cross-check catches the rest. One such entry reached the list and was removed.

Results

Scope

Scouting produced the frozen scope (tag `freeze-ar016-v1`), estimated at 100–250 papers. A paper is in only if it names the PING / E–I loop as the gamma generator *and* draws a functional claim from it; correlation-only phenomenology, mechanism with no functional claim, non-gamma work, and mechanism-agnostic critiques are out. Four boundary questions were resolved during scouting: communication-through-coherence counts only when the E–I/PING generator is named (not for generic gamma); the coincidence-detection and gain axis is included, reached by concept-level rather than keyword queries; reviews that argue mechanism→function are kept as members; and there is no era floor. The scope was seeded by 4 anchor papers:

1. **Borgers, Kopell (2008)**. Gamma Oscillations and Stimulus Selection. doi:10.1162/neco.2007.07-06-289
2. **Borgers, Epstein, Kopell (2008)**. Gamma oscillations mediate stimulus competition and attentional selection in a cortical network model. doi:10.1073/pnas.0809511105
3. **Tiesinga, Sejnowski (2009)**. Cortical Enlightenment: Are Attentional Gamma Oscillations Driven by ING or PING?. doi:10.1016/j.neuron.2009.09.009
4. **Fries (2015)**. Rhythms for Cognition: Communication through Coherence. doi:10.1016/j.neuron.2015.09.034

Selection

The four channels surfaced 3111 distinct candidate papers, every one put through the same membership judge. 160 were accepted and 2951 rejected, an acceptance rate of 5.1 %. Most rejected candidates are gamma-and-cognition papers that do not make the mechanism-*carries*-function argument.

Candidates judged (search + memory union)	1694
Candidates judged (citation graph)	1154
Candidates judged (embedding)	263
Total distinct papers judged	3111
Accepted into the corpus	160 (5.1 %)
Rejected	2951

Provenance
The corpus is the union of complementary channels; no single one would have produced it:

Found by memory only	59
Found by search only	45
Found by memory and search	21
Found by citation graph only	35
Total	160

Only 21 of the 160 papers were caught by both memory and keyword search. The low overlap reflects the channels' complementarity, and is why four were needed rather than one.

Saturation
The stopping point rests on the marginal-yield curve rather than a target count. Each successive channel returned fewer new in-scope papers:

Channel	New in-scope	Candidates screened
memory	80	n/a
keyword search	+45	1552
citation graph	+36	1154
embedding (SPECTER)	+0	263

The yield fell from 80 to +45 to +36 to +0. The final channel (orthogonal, abstract-rich, centred on the hundred strongest seeds) screened 263 semantic neighbours and added nothing. This is a saturation signal rather than a proof: falling returns together with a null from an orthogonal channel. It should be discounted somewhat, since SPECTER is recency-biased and its null is strongest for recent work; but the citation channel already covered the older literature, so the remaining un-caught set is probably small.

Distribution by function axis
Every paper is tagged with the one function its mechanism-to-function argument chiefly serves, from a fixed set of eight. The distribution is the first-order map of the field:

Communication-through-coherence, routing & gating	39
Attention & stimulus selection	17
Stimulus & biased competition	15
Coincidence detection, gain & the temporal window	16
Phase coding & spike timing	26

Binding by synchrony	12
Causal / optogenetic PV–gamma & circuit performance	17
Reviews & theory (mechanism → function)	18
Total	160

Communication-through-coherence and routing is the largest axis (39): the recurring claim is that the E–I gamma rhythm gates which signals pass between areas. The causal / optogenetic cluster (17) is a sizeable group of its own, reflecting a shift from correlating gamma with function to manipulating PV interneurons and measuring the effect. The full annotated list, grouped by axis with a one-line rationale per paper, is the bibliography below.

Discussion

This is a DOI-verified corpus of 160 papers, with a *measured* recall of roughly 65–90 %, and a recorded reason for each paper's inclusion. It is not complete. The tail of a semantic corpus is expensive (papers in no index, mis-tagged, or obscure cost more per paper than the bulk did), and the capture–recapture band (180–250) is itself uncertain because the channels are heterogeneous. The most-cited references of the corpus are foundational papers that are purely mechanism (how gamma is generated) or purely function (attention modulates gamma), each satisfying only half the "mechanism carries function" test; this is why the citation graph returned many candidates but few members. The scope is restrictive, and the set it defines is small.

The strict scope excludes one class of paper by construction: the *critique*. Papers that question a functional claim (does synchrony bind? does gamma carry a usable code?) argue at the level of spike timing or coding in general, without naming the pyramidal↔interneuron loop as the generator, so they fail the mechanism-as-vehicle test. The judge rejected the canonical skeptics it saw (Shadlen & Movshon's *Synchrony Unbound* and both Ray & Maunsell papers), and the recall channels did not reach the others (Thiele & Stoner, Palanca & DeAngelis, Merker), since the scope does not target them. This is consistent for a corpus but a limitation for a review, which should not omit the critical literature. The major critiques are listed in the appendix, outside the corpus, for that later pass.

The run did not support the assumption that a language model is a poor recall channel. Multiplexed across 20 decorrelated passes and filtered through DOI verification, memory was the largest single channel: 80 of the 160 in-scope papers, 59 of them found by memory alone, with low fabrication (44 of 44 real in the controlled pilot, 0 of 180 unresolvable in the full harvest). What made memory usable was decorrelation and verification, not restricting its use.

Next steps

The corpus is the input to the review, not the review. The obvious continuations: (1) a second, independent judge pass to turn the single-judge boundary into a voted one and tighten precision; (2) one more orthogonal recall channel where a key exists (PubMed for the biomedical silo, or author-complete feeds for the core labs), expected, from the yield curve, to add only single digits; (3) the synthesis itself, reading down each axis below to map which

mechanistic claims are contested and where the gaps are. Scope reopens only under a new freeze tag; questions that surfaced during the build are logged, not silently acted on.

The corpus: annotated bibliography by function axis

All 160 papers, grouped by primary function axis and sorted by year, each with a one-sentence statement of how the E–I/PING gamma mechanism carries its functional claim and a tag for the channel that found it (*m* memory · *s* search · *m+s* both · *c* citation).

Communication-through-coherence, routing & gating (39)

1. **Kopell et al. (2000)**. Gamma rhythms and beta rhythms have different synchronization properties. *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.97.4.1867
   E-I gamma synchronizes only over short delays while beta tolerates long ones, assigning gamma the role of local computation and beta of long-range interaction — the mechanism sets the spatial scale of communication. [m]
2. **Tiesinga et al. (2001)**. Optimal information transfer in synchronized neocortical neurons. *Neurocomputing*. doi:10.1016/s0925-2312(01)00464-7
   Argues synchronization among neocortical neurons optimizes information transfer, with inhibition-set synchronous windows serving as the vehicle for efficient signal transmission between populations. [m]
3. **Salinas & Sejnowski (2001)**. Correlated neuronal activity and the flow of neural information. *Nature Reviews Neuroscience*. doi:10.1038/35086012
   Argues correlated/synchronous activity gates the flow of neural information, so inhibition-generated synchrony acts as the vehicle controlling which signals are transmitted downstream. [m]
4. **Fries (2005)**. A mechanism for cognitive dynamics: neuronal communication through neuronal coherence. *Trends in Cognitive Sciences*. doi:10.1016/j.tics.2005.08.011
   The founding communication-through-coherence hypothesis: rhythmic (gamma) inhibitory windows must be phase-aligned between groups for effective interaction, making E-I gamma coherence the vehicle for flexible inter-areal communication. [m]
5. **Middleton et al. (2008)**. NMDA receptor-dependent switching between different gamma rhythm-generating microcircuits in entorhinal cortex. *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.0809302105
   NMDA-dependent switching of interneuron-mediated gamma frequencies in entorhinal cortex matches distinct hippocampal subfield frequencies, so the E-I gamma mechanism sets the temporal channel for information transfer between regions. [c]
6. **Gielen, Krupa & Zeitler (2010)**. Gamma oscillations as a mechanism for selective information transmission. *Biological Cybernetics*. doi:10.1007/s00422-010-0390-x
   Shows PING-type gamma-modulated input makes a downstream neuron phase-lock to and selectively transmit the larger of competing gamma inputs, implementing selective information routing. [m+s]
7. **Tiesinga & Sejnowski (2010)**. Mechanisms for Phase Shifting in Cortical Networks and their Role in Communication through Coherence. *Frontiers in Human Neuroscience*. doi:10.3389/fnhum.2010.00196
   Reciprocal E-I network models phase-shift their gamma output in response to

depolarization or I-cell pulses, providing the phase-adjustment mechanism that implements communication-through-coherence and stimulus selection. [m+s]

8. **Ainsworth et al. (2011)**. Dual Gamma Rhythm Generators Control Interlaminar Synchrony in Auditory Cortex. *The Journal of Neuroscience*. doi:10.1523/jneurosci.2209-11.2011
   Distinct laminar gamma generators (including a granular-layer PING) must frequency-match to provide temporal coupling of co-active regions, making E-I gamma the vehicle for interlaminar/inter-regional synchronization and signal control. [m+s]
9. **Bastos et al. (2012)**. Canonical Microcircuits for Predictive Coding. *Neuron*. doi:10.1016/j.neuron.2012.10.038
   Maps predictive-coding computations onto the canonical cortical microcircuit, assigning gamma from superficial E-I populations to feedforward message passing between hierarchical areas. [m]
10. **Hahn et al. (2013)**. Synfire chains and gamma oscillations: two complementary modes of information transmission in cortical networks. *BMC Neuroscience*. doi:10.1186/1471-2202-14-s1-p226
    Flexible routing of activity across specialized networks is achieved by consistent phase relations between E-I-generated population gamma oscillations, so PING-type coherence is the vehicle for high-fidelity inter-network communication. [s]
11. **Srinivasan, Thorpe & Nunez (2013)**. Top-Down Influences on Local Networks: Basic Theory with Experimental Implications. *Frontiers in Computational Neuroscience*. doi:10.3389/fncom.2013.00029
    A Wilson-Cowan E-I gamma local network is modulated by slow (alpha/theta) rhythms carrying attention/arousal biases, making cross-frequency control of E-I gamma the vehicle for top-down/bottom-up interaction between spatial scales. [s]
12. **Roberts et al. (2013)**. Robust Gamma Coherence between Macaque V1 and V2 by Dynamic Frequency Matching. *Neuron*. doi:10.1016/j.neuron.2013.03.003
    Despite the garbled title, the abstract shows V1-V2 gamma coherence is maintained across stimulus-induced frequency changes, supporting communication-through-coherence via correlated E-I gamma between areas. [m+s]
13. **Pietersen et al. (2014)**. Transition between fast and slow gamma modes in rat hippocampus area CA1 in vitro is modulated by slow CA3 gamma oscillations. *The Journal of Physiology*. doi:10.1113/jphysiol.2013.263889
    Feedback-inhibition-generated fast CA1 gamma can be forced to the slower CA3 gamma via feed-forward inhibition, letting CA1 switch its effective communication between entorhinal cortex and CA3, with gamma synchrony determining the effectiveness of neuronal communication for memory encoding vs recall. [s]
14. **Akam & Kullmann (2014)**. Oscillatory multiplexing of population codes for selective communication in the mammalian brain. *Nature Reviews Neuroscience*. doi:10.1038/nrn3668
    Review proposes oscillatory (gamma) multiplexing of firing-rate population codes enables selective inter-areal communication by reconfiguring effective connectivity, with periodic inhibitory modulation as the routing substrate. [m]

15. **Bastos et al. (2015)**. Visual Areas Exert Feedforward and Feedback Influences through Distinct Frequency Channels. *Neuron*. doi:10.1016/j.neuron.2014.12.018
Shows visual areas route feedforward influences through gamma and feedback through lower frequencies, so E-I gamma coherence implements directional inter-areal communication. [m]
16. **Fries (2015)**. Rhythms for Cognition: Communication through Coherence. *Neuron*. doi:10.1016/j.neuron.2015.09.034
Fries's canonical statement that flexible neuronal communication is mechanistically implemented by coherence of E-I gamma rhythms establishing open windows between sending and receiving groups. [m]
17. **Lee, Whittington & Kopell (2015)**. Potential Mechanisms Underlying Intercortical Signal Regulation via Cholinergic Neuromodulators. *The Journal of Neuroscience*. doi:10.1523/jneurosci.0629-15.2015
Cholinergic modulation lets top-down beta/gamma interact with a bottom-up E-I gamma rhythm to gate signal flow between primary sensory and association cortex, making the PING gamma the vehicle for flexible inter-areal routing. [s]
18. **Harnack, Ernst & Pawelzik (2015)**. A model for attentional information routing through coherence predicts biased competition and multistable perception. *Journal of Neurophysiology*. doi:10.1152/jn.01038.2014
A two-layer spiking network with lateral inhibition uses phase shifts between sending and receiving layers to route the attended stimulus, reproducing both biased competition and selective information routing via gamma coherence. [c]
19. **McLelland & VanRullen (2016)**. Theta-Gamma Coding Meets Communication-through-Coherence: Neuronal Oscillatory Multiplexing Theories Reconciled. *PLOS Computational Biology*. doi:10.1371/journal.pcbi.1005162
Feedback-inhibition (PING) gamma phase-matching selectively routes single items via communication-through-coherence while theta-gamma multiplexing supports a phase code, with the E-I gamma rhythm as the vehicle for both. [m+s]
20. **Veit et al. (2017)**. Cortical gamma band synchronization through somatostatin interneurons. *Nature Neuroscience*. doi:10.1038/nn.4562
Optogenetics shows dendrite-targeting SOM interneurons are required for visually induced gamma and long-distance coherence across visual cortex, casting the inhibitory gamma mechanism as the vehicle for synchronizing distributed assemblies for information transfer. [m]
21. **Palmigiano et al. (2017)**. Flexible information routing by transient synchrony. *Nature Neuroscience*. doi:10.1038/nn.4569
Modeling shows transient gamma synchrony among E-I networks flexibly routes information by dynamically reconfiguring effective connectivity, the core communication-through-coherence claim. [m]
22. **Wal & Tiesinga (2017)**. Phase Difference between Model Cortical Areas Determines Level of Information Transfer. *Frontiers in Computational Neuroscience*. doi:10.3389/fncom.2017.00006
Two PING circuits transfer information maximally only when their frequency-set phase

difference aligns their susceptibility windows, so the E-I gamma loop's phase relationship is the vehicle for flexible inter-areal routing (communication-through-coherence). [m]

23. **Strüber et al. (2017)**. Distance-dependent inhibition facilitates focality of gamma oscillations in the dentate gyrus. *Nature Communications*. doi:10.1038/s41467-017-00936-3
    Distance-dependent inhibition among perisomatic interneurons fragments dentate gamma into multiple focal synchronous centers, letting the interneuron-based PING mechanism route and process complex inputs in parallel across flexibly organized assemblies. [c]
24. **Sherfey et al. (2018)**. Flexible resonance in prefrontal networks with strong feedback inhibition. *PLOS Computational Biology*. doi:10.1371/journal.pcbi.1006357
    Strong feedback inhibition makes PFC networks resonant relays that preferentially transmit inputs near their gamma frequency, so the PING E-I loop is the vehicle for frequency-selective routing/gain of oscillatory signals. [m]
25. **Adesnik (2018)**. Layer-specific excitation/inhibition balances during neuronal synchronization in the visual cortex. *The Journal of Physiology*. doi:10.1113/jp274986
    Layer-specific optogenetic drive shows excitatory neurons in any layer entrain gamma while balanced inhibition promotes propagation of activity to downstream layers, making layer-specific E-I gamma the vehicle for inter-laminar information flow. [c]
26. **Sherfey et al. (2019)**. Prefrontal oscillations modulate the propagation of neuronal activity required for working memory. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/531574
    Biophysical PFC modeling shows local inhibition-based gamma/beta oscillations set the inter-burst period that selectively gates which working-memory item propagates to downstream effectors, making the E-I rhythm the vehicle for flexible signal routing. [s]
27. **Dumont & Gutkin (2019)**. Macroscopic phase resetting-curves determine oscillatory coherence and signal transfer in inter-coupled neural circuits. *PLOS Computational Biology*. doi:10.1371/journal.pcbi.1007019
    Macroscopic phase-resetting curves of PING/ING gamma circuits determine the phase-locking states that govern signal transfer between coupled areas, making the E-I gamma rhythm the vehicle for inter-areal communication. [s]
28. **Sherfey et al. (2020)**. Prefrontal oscillations modulate the propagation of neuronal activity required for working memory. *Neurobiology of Learning and Memory*. doi:10.1016/j.nlm.2020.107228
    Biophysical PFC model argues local interneuron-generated oscillations selectively gate which working-memory item propagates to downstream effectors, making the inhibition-based rhythm the vehicle for flexible signal routing. [s]
29. **Lewis et al. (2020)**. Cortical resonance selects coherent input. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/2020.12.09.417782
    Optogenetically injecting white-noise excitation into pyramidal cells reveals that feedback-inhibition-generated cortical gamma resonance selectively transmits coherent input components, giving causal support for communication-through-coherence. [c]
30. **Reyner-Parra & Huguet (2021)**. Phase-locking patterns underlying effective communication in exact firing rate models of neural networks. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/2021.08.13.456218

Exact mean-field PING models show excitatory-inhibitory populations must phase-lock so input volleys arrive at excitability peaks, implementing communication-through-coherence between gamma-oscillating circuits. [s]

31. **Lucas, Klaus & Christoph (2021)**. Synchronization through uncorrelated noise in excitatory-inhibitory networks. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/2021.10.29.466430
    Uncorrelated synaptic noise induces synchronization between PING excitatory-inhibitory networks and thereby facilitates inter-regional gamma communication via a mechanism distinct from strong coupling. [s]
32. **Lewis et al. (2021)**. Cortical gamma-band resonance preferentially transmits coherent input. *Cell Reports*. doi:10.1016/j.celrep.2021.109083
    Optogenetically probing pyramidal cells shows the recurrent E-I gamma resonance preferentially transmits coherent input, providing causal evidence for communication-through-coherence as the function of cortical gamma. [c]
33. **Reyner-Parra & Huguet (2022)**. Phase-locking patterns underlying effective communication in exact firing rate models of neural networks. *PLOS Computational Biology*. doi:10.1371/journal.pcbi.1009342
    Exact mean-field PING networks are analyzed to find the phase-locked states that make input volleys arrive at excitability peaks, so the E-I gamma rhythm is the vehicle for effective communication-through-coherence. [s]
34. **Rebscher, Obermayer & Metzner (2022)**. Synchronization Through Uncorrelated Noise in Excitatory-Inhibitory Networks. *Frontiers in Computational Neuroscience*. doi:10.3389/fncom.2022.825865
    Uncorrelated synaptic noise enhances between-region synchronization of PING networks, so the E-I gamma mechanism supports inter-regional communication even under noisy conditions. [s]
35. **Veit et al. (2023)**. Cortical VIP neurons locally control the gain but globally control the coherence of gamma band rhythms. *Neuron*. doi:10.1016/j.neuron.2022.10.036
    Shows VIP disinhibition (via VIP->SST) globally tunes long-range gamma coherence for stimulus-matched routing while locally scaling gamma gain, making the E-I gamma loop the vehicle for coherence-based inter-areal communication. [m+s]
36. **Katsanevaki et al. (2023)**. Attentional effects on local V1 microcircuits explain selective V1-V4 communication. *NeuroImage*. doi:10.1016/j.neuroimage.2023.120375
    Dynamic causal modeling attributes selective V1-V4 routing to attentional changes in intrinsic V1 inhibition that let the attended stimulus's gamma entrain V4, making the local E-I gamma circuit the vehicle for selective communication. [c]
37. **Gonzalez-Burgos et al. (2023)**. Mechanisms regulating the properties of inhibition-based gamma oscillations in primate prefrontal and parietal cortices. *Cerebral Cortex*. doi:10.1093/cercor/bhad077
    Recurrent-excitation and GABAAR-mediated synchrony set distinct inhibition-based gamma frequencies in primate DLPFC versus PPC, and these matched frequencies are argued to enable information transfer between the two areas of the working-memory network. [c]

38. **Phensy et al. (2024)**. Prefrontal gamma oscillations engage dynamic cell type-specific configurations to support flexible behavior. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/2024.03.08.584173
    PV-interneuron-entrained ~40Hz gamma is shown (via optogenetics/voltage imaging) to organize distinct circuit-specific synchronization motifs that route PFC->mediodorsal-thalamus communication in a behaviorally selective way to support flexible behavior. [s]
39. **Phensy et al. (2026)**. Prefrontal gamma oscillations engage dynamic cell-type-specific configurations to support flexible behavior. *Neuron*. doi:10.1016/j.neuron.2026.05.002
    Uses voltage imaging and optogenetics to show PV-interneuron gamma synchronization with MD-projecting PFC neurons forms circuit-specific motifs that route PFC->MD communication in support of flexible behavior. [s]

Attention & stimulus selection (17)

1. **Niebur, Koch & Rosin (1993)**. An oscillation-based model for the neuronal basis of attention. *Vision Research*. doi:10.1016/0042-6989(93)90236-p
    Proposes an oscillation-based model where interneuron-network gamma synchronization among attended neurons implements the neuronal basis of selective attention. [m]
2. **Niebur & Koch (1994)**. A model for the neuronal implementation of selective visual attention based on temporal correlation among neurons. *Journal of Computational Neuroscience*. doi:10.1007/bf00962722
    Models selective visual attention via temporal correlation among neurons, using inhibitory-network-generated synchrony to tag and select the attended stimulus's assembly. [m]
3. **Börgers, Epstein & Kopell (2005)**. Background gamma rhythmicity and attention in cortical local circuits: A computational study. *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.0502366102
    Computational model links cholinergic neuromodulation to a background PING gamma rhythm that amplifies stimulus-specific responses and enhances competition, proposing gamma as the substrate of preparatory attention. [m+s]
4. **Tiesinga et al. (2005)**. Inhibitory synchrony as a mechanism for attentional gain modulation. *arXiv (Cornell University)*. doi:10.48550/arxiv.q-bio/0503019
    Top-down control of local interneuron-network synchrony modulates a target neuron's firing rate and spike-LFP gamma coherence, so attentional gain in V4 is the vehicle-function of synchronized inhibitory (E-I) gamma. [s]
5. **Buia & Tiesinga (2006)**. Attentional modulation of firing rate and synchrony in a model cortical network. *Journal of Computational Neuroscience*. doi:10.1007/s10827-006-6358-0
    Models attention as reduced drive to interneurons in a cortical E-I network, so that gamma synchrony set by the interneuron network shifts the contrast-response (gain) curve as seen in V4. [m]
6. **Mishra, Fellous & Sejnowski (2006)**. Selective attention through phase relationship of excitatory and inhibitory input synchrony in a model cortical neuron. *Neural Networks*. doi:10.1016/j.neunet.2006.08.005
    Models selective attention as controlled by the phase relationship between synchronous

excitatory and inhibitory (interneuron) inputs, gating a cortical neuron's response by gamma-phase alignment. [m]

7. **Tiesinga & Buia (2006)**. Attentional modulation in layer 4 of the visual cortex could be mediated by interneurons with complex receptive field characteristics. *arXiv (Cornell University)*. doi:10.48550/arxiv.q-bio/0611030
   Increasing interneuron excitability in a layer-4 orientation-tuned circuit raises spike-LFP gamma coherence without changing rate, reproducing attentional modulation through control of the E-I gamma network. [s]
8. **Buia & Tiesinga (2008)**. Role of Interneuron Diversity in the Cortical Microcircuit for Attention. *Journal of Neurophysiology*. doi:10.1152/jn.01004.2007
   A V4 microcircuit model with PV-like feedforward interneurons shows feature-based attention shifts assembly synchrony into the gamma band and resolves stimulus competition, arguing interneuron-diversity-based gamma mediates attentional selection among competing stimuli. [m+s]
9. **Deco & Thiele (2009)**. Attention – oscillations and neuropharmacology. *European Journal of Neuroscience*. doi:10.1111/j.1460-9568.2009.06833.x
   Review of attention arguing that NMDA/AMPA- (and ACh-) tuned gamma-range rhythmic synchronization in local circuits enables enhanced processing of attended stimuli and efficient communication between ensembles. [m+s]
10. **Ardid et al. (2010)**. Reconciling Coherent Oscillation with Modulationof Irregular Spiking Activity in Selective Attention:Gamma-Range Synchronization between Sensoryand Executive Cortical Areas. *The Journal of Neuroscience*. doi:10.1523/jneurosci.4222-09.2010
    A reciprocal MT<->PFC spiking loop reproduces attention-driven interareal gamma coherence that is feature-selective and enhances sensory gain, making sparsely-synchronized E-I gamma the vehicle for selective attention. [m+s]
11. **Vinck et al. (2013)**. Attentional Modulation of Cell-Class-Specific Gamma-Band Synchronization in Awake Monkey Area V4. *Neuron*. doi:10.1016/j.neuron.2013.08.019
    Dissects attention's effect on gamma synchronization of narrow- vs broad-spiking (interneuron vs pyramidal) cells in V4, showing attention reshapes the E-I gamma cycle that selects attended stimuli. [m+s]
12. **Blaes & Burwick (2015)**. Attentional Bias Through Oscillatory Coherence Between Excitatory Activity and Inhibitory Minima. *Neural Computation*. doi:10.1162/neco_a_00742
    A coupled excitatory-inhibitory gamma network implements attentional bias as oscillatory (spike-field) coherence between target-encoding excitatory units and the inhibitory pool, selecting the attended target while suppressing distracters. [s]
13. **Boroujeni, Tiesinga & Womelsdorf (2020)**. Interneuron Specific Gamma Synchronization Indexes Cue Uncertainty and Prediction Errors in Lateral Prefrontal and Anterior Cingulate Cortex. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/2020.07.24.220319
    A specific interneuron subclass's 35-45 Hz gamma-synchronous spiking indexes cue uncertainty and prediction errors in PFC/ACC and is modeled as a soft winner-take-all gate — interneuron gamma as the selection/gating vehicle. [s]

14. **Lu et al. (2020)**. Phasic cholinergic signaling promotes emergence of local gamma rhythms in excitatory–inhibitory networks. *European Journal of Neuroscience*. doi:10.1111/ejn.14744
    Computational E-I modeling shows brief acetylcholine pulses (via M-current suppression) trigger transient PING gamma, arguing the E-I loop's cholinergically driven gamma underlies cue detection in behavioral attention tasks. [s]
15. **Wagatsuma, Nobukawa & Fukai (2023)**. A microcircuit model involving parvalbumin, somatostatin, and vasoactive intestinal polypeptide inhibitory interneurons for the modulation of neuronal oscillation during visual processing. *Cerebral Cortex*. doi:10.1093/cercor/bhac355
    A PV/SOM/VIP microcircuit model in which PV-generated gamma (vs SOM beta) regulates the integration of feedforward sensory and feedback attentional signals during visual processing. [s]
16. **Zheng et al. (2024)**. Analyzing top-down visual attention in the context of gamma oscillations: a layer- dependent network-of- networks approach. *Frontiers in Computational Neuroscience*. doi:10.3389/fncom.2024.1439632
    A layer-dependent network-of-PING-networks reproduces top-down attentional enhancement of neuronal responses, using the E-I gamma loop across cortical layers as the vehicle for selective attention. [s]
17. **Sajedin, FallahTaherpazir & Menhaj (2025)**. Cholinergic modulation mediates attentional mechanism to enhance coherence between cortical layers in macaque V1 and V4. *Scientific Reports*. doi:10.1038/s41598-025-24109-1
    Biophysical V1-V4 model shows cholinergic drive enhances gamma synchrony/coherence via increased inhibitory drive to reproduce attentional modulation of inter-areal communication in visual cortex. [s]

Stimulus & biased competition (15)

1. **Olufsen et al. (2003)**. New Roles for the Gamma Rhythm: Population Tuning and Preprocessing for the Beta Rhythm. *Journal of Computational Neuroscience*. doi:10.1023/a:1021124317706
    Proposes a new role for gamma in population tuning, where the E-I gamma cycle sharpens which neurons fire (winner selection) and preprocesses activity for the beta rhythm. [m]
2. **Doiron et al. (2003)**. Inhibitory feedback required for network oscillatory responses to communication but not prey stimuli. *Nature*. doi:10.1038/nature01360
    Shows inhibitory feedback is required to generate oscillatory network responses selectively to global communication stimuli but not prey stimuli, making the E-I feedback loop the vehicle for stimulus-specific selection. [m]
3. **Tiesinga & Sejnowski (2004)**. Rapid Temporal Modulation of Synchrony by Competition in Cortical Interneuron Networks. *Neural Computation*. doi:10.1162/089976604322742029
    Hodgkin-Huxley interneuron-network models show selectively activating a fraction of GABAergic interneurons rapidly synchronizes them and suppresses competitors, so 'synchrony by competition' modulates gamma synchrony (with minimal rate change) to enhance attended stimulus impact on downstream cortex. [m]

4. **Tiesinga (2004)**. Stimulus competition by inhibitory interference. *arXiv (Cornell University)*. doi:10.48550/arxiv.q-bio/0410019
   A V4 model where two interneuron networks deliver synchronous inhibitory volleys shows stimulus competition set by inter-volley delay, with attention biasing the competition, making interference between inhibitory gamma volleys the vehicle for biased competition. [c]
5. **Tiesinga (2005)**. Stimulus Competition by Inhibitory Interference. *Neural Computation*. doi:10.1162/0899766054796905
   A V4 model driven by synchronous inhibition from competing local interneuron networks reproduces biased-competition firing-rate and spike-field-coherence effects, arguing attention resolves stimulus competition by modulating gamma-generating interneuron synchrony. [m]
6. **Tiesinga (2006)**. Stimulus competition by inhibitory interference. *Neurocomputing*. doi:10.1016/j.neucom.2005.12.089
   Proposes that interference between competing inhibitory (interneuron) gamma rhythms lets one stimulus assembly suppress another, implementing stimulus competition through inhibition. [m]
7. **Börgers & Kopell (2008)**. Gamma Oscillations and Stimulus Selection. *Neural Computation*. doi:10.1162/neco.2007.07-06-289
   Borgers and Kopell show a coherent gamma-frequency input gains a large competitive advantage over less coherent inputs when the target circuit includes GABA-A interneurons, providing the PING-based mechanism for attentional biasing of stimulus competition. [m+s]
8. **Börgers, Epstein & Kopell (2008)**. Gamma oscillations mediate stimulus competition and attentional selection in a cortical network model. *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.0809511105
   Cortical network model shows PING gamma oscillations mediate stimulus competition and attentional selection via oscillatory selection, where interneuron coherence regulates competing assemblies' firing rates. [m+s]
9. **Almeida, Idiart & Lisman (2009)**. A Second Function of Gamma Frequency Oscillations: An E%-Max Winner-Take-All Mechanism Selects Which Cells Fire. *Journal of Neuroscience*. doi:10.1523/jneurosci.6044-08.2009
   Gamma-frequency feedback inhibition implements an E%-max winner-take-all that selects which principal cells fire and sharpens V1 orientation tuning, making the E-I gamma loop the vehicle for competitive stimulus selection. [m]
10. **Wildie (2012)**. Establishing communication between neuronal populations through competitive entrainment. *Frontiers in Computational Neuroscience*. doi:10.3389/fncom.2011.00062
    Multiple stimuli compete to entrain a target population by gamma coherence and only the winner is transmitted via communication-through-coherence, making the PING E-I loop the vehicle for competitive stimulus selection and routing. [m+s]
11. **Börgers & Walker (2013)**. Toggling between gamma-frequency activity and suppression of cell assemblies. *Frontiers in Computational Neuroscience*. doi:10.3389/fncom.2013.00033

The PING suppression boundary lets strongly-driven I-cells suppress subsets of E-cells so that strong assembly volleys are followed by weaker ones, making the reciprocal E-I loop a mechanism for competitive gating/winner-selection among cell assemblies. [m]

12. **Mostafa, Müller & Indiveri (2015)**. Rhythmic Inhibition Allows Neural Networks to Search for Maximally Consistent States. *Neural Computation*. doi:10.1162/neco_a_00785
    Gamma-band rhythmic inhibition drives coupled cortical motifs to search for maximally consistent states, implementing constraint-satisfaction and perceptual multistability so the E-I gamma rhythm mediates competition among candidate network configurations. [c]
13. **Rennó-Costa, Teixeira & Soltesz (2019)**. Regulation of gamma-frequency oscillation by feedforward inhibition: A computational modeling study. *Hippocampus*. doi:10.1002/hipo.23093
    Argues the reciprocal E-I (PING) gamma rhythm selects which subset of cells fires each cycle (level of selection/winner-take-all), and that feedforward inhibition stabilizes this selection against input magnitude. [s]
14. **Onorato et al. (2020)**. A Distinct Class of Bursting Neurons with Strong Gamma Synchronization and Stimulus Selectivity in Monkey V1. *Neuron*. doi:10.1016/j.neuron.2019.09.039
    Argues a distinct bursting cell class in V1 that strongly gamma-synchronizes carries the strongest stimulus selectivity, tying E-I/PING gamma synchronization to which stimulus-tuned population dominates the local representation. [m]
15. **Boroujeni, Tiesinga & Womelsdorf (2021)**. Interneuron-specific gamma synchronization indexes cue uncertainty and prediction errors in lateral prefrontal and anterior cingulate cortex. *eLife*. doi:10.7554/elife.69111
    An identified interneuron subclass shows uncertainty-linked gamma-synchronous spiking that circuit modeling interprets as soft winner-take-all gating, making interneuron gamma the vehicle for competitive selection of high-uncertainty information. [m+s]

Coincidence detection, gain & the temporal window (16)

1. **König, Engel & Singer (1996)**. Integrator or coincidence detector? The role of the cortical neuron revisited. *Trends in Neurosciences*. doi:10.1016/s0166-2236(96)80019-1
   Argues cortical neurons operate as coincidence detectors rather than integrators, so inhibition-set narrow integration windows enable synchrony-based signaling relevant to temporal binding. [m]
2. **Burchell, Faulkner & Whittington (1998)**. Gamma frequency oscillations gate temporally coded afferent inputs in the rat hippocampal slice. *Neuroscience Letters*. doi:10.1016/s0304-3940(98)00676-4
   Hippocampal gamma oscillations gate temporally coded afferent inputs, so the inhibition-defined gamma window of the E-I network determines which timed inputs are admitted (title-inferred, empty abstract). [c]
3. **Palva et al. (2000)**. Fast Network Oscillations in the Newborn Rat HippocampusIn Vitro. *The Journal of Neuroscience*. doi:10.1523/jneurosci.20-03-01170.2000
   GABAergic gamma-frequency modulation synchronizes newborn hippocampal pyramidal firing to millisecond precision, proposed as a substrate for selective coincidence detection, making the E-I gamma the vehicle for a precise integration window. [s]

4. **Tiesinga et al. (2002)**. Information transfer in entrained cortical neurons. *Network: Computation in Neural Systems*. doi:10.1080/net.13.1.41.66
   Synchronized gap-junction-coupled interneuron inhibition entrains pyramidal cells and, counterintuitively, increases the mutual information their spike phase carries about input — inhibitory gamma entrainment as the vehicle for information transfer. [m]
5. **Jonas et al. (2004)**. Interneuron Diversity series: Fast in, fast out – temporal and spatial signal processing in hippocampal interneurons. *Trends in Neurosciences*. doi:10.1016/j.tins.2003.10.010
   Reviews how fast-spiking hippocampal interneurons' rapid, precise signaling ('fast in, fast out') imposes narrow temporal integration windows, positioning interneuron speed as the basis for coincidence detection and gamma timing. [m]
6. **Tiesinga et al. (2004)**. Inhibitory synchrony as a mechanism for attentional gain modulation. *Journal of Physiology-Paris*. doi:10.1016/j.jphysparis.2005.09.002
   Shows that increasing synchrony of local interneuron (inhibitory) inputs modulates a neuron's f-I gain and spike-LFP gamma coherence, offering inhibitory synchrony as the mechanism of attentional gain modulation. [m]
7. **Tiesinga et al. (2004)**. Synchronization as a mechanism for attentional gain modulation. *Neurocomputing*. doi:10.1016/j.neucom.2004.01.108
   Companion modeling work arguing synchrony of inhibitory interneuron input controls neuronal gain and spike-input coherence, the substrate of attentional gain modulation in the gamma band. [m]
8. **Paik, Kumar & Glaser (2009)**. Spontaneous Local Gamma Oscillation Selectively Enhances Neural Network Responsiveness. *PLoS Computational Biology*. doi:10.1371/journal.pcbi.1000342
   In an E-I Hodgkin-Huxley model, reciprocal E-I/I-E gamma injects a periodic current that amplifies weak feedforward inputs, so the PING rhythm acts as an input-strength-dependent gain mechanism selectively boosting network responsiveness. [c]
9. **Knoblich et al. (2010)**. What do We Gain from Gamma? Local Dynamic Gain Modulation Drives Enhanced Efficacy and Efficiency of Signal Transmission. *Frontiers in Human Neuroscience*. doi:10.3389/fnhum.2010.00185
   Optogenetic 40 Hz FS-interneuron drive sharpens a post-stimulus temporal window that boosts spike precision and downstream gain, making inhibition-set coincidence windows of the E-I loop the vehicle for efficient signal transmission. [m+s]
10. **Otte, Hasenstaub & Callaway (2010)**. Cell Type-Specific Control of Neuronal Responsiveness by Gamma-Band Oscillatory Inhibition. *The Journal of Neuroscience*. doi:10.1523/jneurosci.4818-09.2010
    Using dynamic-clamp gamma-frequency perisomatic inhibition, the paper shows cell-type-specific control of neuronal responsiveness, so synchronized interneuron gamma sets the gain and integration window with which each cell type transforms input to output. [c]
11. **Li et al. (2011)**. Impact of gamma-oscillatory inhibition on the signal transmission of a cortical pyramidal neuron. *Cognitive Neurodynamics*. doi:10.1007/s11571-011-9169-6
    Modeling a pyramidal neuron under gamma-oscillatory inhibition, the paper shows this rhythmic perisomatic inhibition shapes signal transmission, so the E-I gamma cycle gates the neuron's gain and effective integration window. [c]

12. **Lozano-Soldevilla et al. (2014)**. GABAergic Modulation of Visual Gamma and Alpha Oscillations and Its Consequences for Working Memory Performance. *Current Biology*. doi:10.1016/j.cub.2014.10.017
    Pharmacological GABAergic modulation shifts visual gamma (and alpha), and these changes in inhibition-generated gamma track working-memory performance, casting the E-I gamma cycle as the inhibitory processing window supporting memory. [c]
13. **Cardin (2018)**. Inhibitory Interneurons Regulate Temporal Precision and Correlations in Cortical Circuits. *Trends in Neurosciences*. doi:10.1016/j.tins.2018.07.015
    Reviews how GABAergic interneurons in the E-I circuit regulate spike-timing precision and suppress slow correlations, making inhibition the vehicle for temporal windowing and coordinated cortical rhythms. [m]
14. **Orekhova et al. (2018)**. Input-dependent modulation of MEG gamma oscillations reflects gain control in the visual cortex. *Scientific Reports*. doi:10.1038/s41598-018-26779-6
    MEG visual gamma power follows a bell-shaped input-output curve interpreted as inhibitory gain control, using the E-I balance underlying gamma as a non-invasive index of the cortex's capacity to counterbalance excitation with inhibition. [c]
15. **Han et al. (2020)**. High-Frequency Synchronization Improves Firing Rate Contrast and Information Transmission Efficiency in E/I Neuronal Networks. *Neural Plasticity*. doi:10.1155/2020/8823111
    High-frequency synchronization in E/I network models enhances firing-rate contrast and information-encoding efficiency, so the E-I gamma rhythm sharpens gain/contrast to improve information transmission. [c]
16. **Susin & Destexhe (2021)**. Integration, coincidence detection and resonance in networks of spiking neurons expressing Gamma oscillations and asynchronous states. *PLOS Computational Biology*. doi:10.1371/journal.pcbi.1009416
    PING/ING gamma states modulate network responsiveness and resonance to afferent stimuli relative to the asynchronous mode, making the E-I gamma cycle the vehicle for gating stimulus integration and gain. [m]

Phase coding & spike timing (26)

1. **Traub, Whittington & Jefferys (1997)**. Gamma Oscillation Model Predicts Intensity Coding by Phase Rather than Frequency. *Neural Computation*. doi:10.1162/neco.1997.9.6.1251
    A distributed pyramidal-interneuron network where interneuron spike doublets set a zero-phase temporal framework predicts stimulus intensity is coded by gamma phase rather than frequency, making the E-I gamma cycle the reference for a phase code that also supports feature binding. [s]
2. **Tiesinga et al. (2002)**. Information transfer in entrained cortical neurons. *Network: Computation in Neural Systems*. doi:10.1088/0954-898x/13/1/302
    When pyramidal cells are entrained by synchronized inhibitory gamma input, mutual information between the number of inputs and the output-spike phase lag rises, so the E-I gamma cycle acts as a phase code carrying afferent information. [c]
3. **Hasenstaub et al. (2005)**. Inhibitory Postsynaptic Potentials Carry Synchronized Frequency Information in Active Cortical Networks. *Neuron*. doi:10.1016/

j.neuron.2005.06.016
Shows rhythmic IPSPs from the interneuron network carry synchronized gamma-frequency timing information to pyramidal cells, providing the inhibitory clock that times cortical spiking. [m]

4. **Fries, Nikolić & Singer (2007)**. The gamma cycle. *Trends in Neurosciences*. doi:10.1016/j.tins.2007.05.005
Proposes the gamma cycle produced by the E-I loop as a temporal frame that orders pyramidal spikes by excitation strength, supplying a phase-of-firing code and reference for downstream reading. [m]
5. **Tiesinga, Fellous & Sejnowski (2008)**. Regulation of spike timing in visual cortical circuits. *Nature Reviews Neuroscience*. doi:10.1038/nrn2315
Review argues interneuron-generated gamma regulates the precise spike timing of pyramidal cells in visual cortical circuits, framing rhythmic inhibition as the mechanism that sets a spike-timing code. [m]
6. **Morita et al. (2008)**. Recurrent Synaptic Input and the Timing of Gamma-Frequency-Modulated Firing of Pyramidal Cells during Neocortical "UP" States. *The Journal of Neuroscience*. doi:10.1523/jneurosci.3948-07.2008
Dynamic-clamp shows pyramidal gamma-phase firing during UP states requires low-latency FS->RS inhibition rather than recurrent excitation, making the PING inhibitory drive the vehicle for gamma-timed pyramidal spiking that maximizes representational capacity. [s]
7. **Mazzoni et al. (2008)**. Encoding of Naturalistic Stimuli by Local Field Potential Spectra in Networks of Excitatory and Inhibitory Neurons. *PLoS Computational Biology*. doi:10.1371/journal.pcbi.1000239
A sparse E-I network encodes static input rates into gamma-range oscillations generated by excitatory-inhibitory interaction, so gamma from the E-I loop serves as the code carrying sensory information in the LFP. [c]
8. **Edward (2010)**. Neuronal biophysics modulate the ability of gamma oscillations to control response timing. *Frontiers in Neuroscience*. doi:10.3389/conf.fnins.2010.03.00004
Neuronal biophysics determine how synchronized inhibitory gamma enforces temporal precision and controls the timing of postsynaptic responses, making the interneuron-gamma inhibition the vehicle for spike-timing control. [s]
9. **Miconi & VanRullen (2010)**. The Gamma Slideshow: Object-Based Perceptual Cycles in a Model of the Visual Cortex. *Frontiers in Human Neuroscience*. doi:10.3389/fnhum.2010.00205
In a V1 model, feedback interneuron inhibition generating gamma combines with lateral Gestalt connections so different objects fire on successive gamma cycles, using the PING cycle as a phase reference that temporally segments the scene into perceptual cycles. [c]
10. **Womelsdorf et al. (2012)**. Orientation selectivity and noise correlation in awake monkey area V1 are modulated by the gamma cycle. *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.1114223109
Orientation selectivity and noise correlations vary across the gamma cycle, so the E-I-driven gamma phase acts as a temporal reference that maximizes stimulus-selective, low-noise spiking for downstream readout. [c]

11. **Nikolić, Fries & Singer (2013)**. Gamma oscillations: precise temporal coordination without a metronome. *Trends in Cognitive Sciences*. doi:10.1016/j.tics.2012.12.003
Argues gamma provides precise, self-organized temporal coordination of spikes without an external clock, casting the E-I gamma cycle as a flexible timing reference for neural coordination and communication. [m]
12. **Sancristóbal et al. (2013)**. Emergent bimodal firing patterns implement different encoding strategies during gamma-band oscillations. *Frontiers in Computational Neuroscience*. doi:10.3389/fncom.2013.00018
In a balanced E-I network expressing gamma, emergent bimodal firing lets neurons co-encode input rate and LFP gamma phase, making the E-I gamma cycle the vehicle for a phase-based encoding strategy. [m+s]
13. **Jadi & Sejnowski (2014)**. Cortical oscillations arise from contextual interactions that regulate sparse coding. *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.1405300111
Model cortical circuit shows the balance of mono- vs disynaptic drive to inhibitory neurons regulates gamma power/frequency and hence pyramidal spike timing, linking interneuron-driven oscillation to a stimulus-modulated spike-timing code. [m]
14. **Lowet et al. (2015)**. Input-Dependent Frequency Modulation of Cortical Gamma Oscillations Shapes Spatial Synchronization and Enables Phase Coding. *PLOS Computational Biology*. doi:10.1371/journal.pcbi.1004072
Input-dependent gamma-frequency modulation in an E-I network sets phase relations that carry stimulus information (phase coding) and organize spatial synchronization, making the PING gamma cycle the vehicle for temporal coding. [m]
15. **Li & Cleland (2017)**. A coupled-oscillator model of olfactory bulb gamma oscillations. *PLOS Computational Biology*. doi:10.1371/journal.pcbi.1005760
In the olfactory bulb, a pyramidal-resonance interneuron-network gamma (PRING) provides a zero-phase common clock that phase-restricts informative principal-cell spikes, making the E-I gamma cycle the vehicle for a phase-of-firing code. [s]
16. **Shin & Moore (2018)**. Persistent Gamma Spiking in Non-Sensory Fast-Spiking Cells Predicts Perceptual Success. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/374314
A non-sensory fast-spiking interneuron subtype whose regular gamma-periodic firing (the FS-driven gamma template) persists across stimulus onset predicts perceptual detection, arguing the E-I gamma clock provides the temporal reference that enables perception. [s]
17. **Onorato et al. (2019)**. A distinct class of bursting neurons with strong gamma synchronization and stimulus selectivity in monkey V1. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/583955
A bursting excitatory 'pacemaker' cell class in monkey V1 whose gamma phase-locking (via rhythmic inhibition from interneurons, i.e. the E-I loop) is highly predictive of its orientation tuning, arguing gamma-phase relative to the E-I rhythm encodes and transmits stimulus information. [s]
18. **Feng et al. (2019)**. Gamma Oscillations in the Basolateral Amygdala: Biophysical Mechanisms and Computational Consequences. *eneuro*. doi:10.1523/eneuro.0388-18.2018
A biophysical BLA model shows gamma arising from reciprocal principal-cell/fast-

spiking-interneuron interaction entrains individual neurons' spike timing, making the PING cycle a temporal reference that structures amygdalar spiking. [c]

19. **Meneghetti et al. (2020)**. Thalamic inputs determine functionally distinct gamma bands in mouse primary visual cortex. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/2020.07.09.194811
    A recurrent E-I network reproduces a broadband gamma channel encoding high-contrast information distinct from the thalamic narrowband, so the E-I gamma band functions as a code for complementary visual features (preprint of the eNeuro study). [c]
20. **Meneghetti et al. (2021)**. Narrow and Broad γ Bands Process Complementary Visual Information in Mouse Primary Visual Cortex. *eneuro*. doi:10.1523/eneuro.0106-21.2021
    Broadband gamma arising from cortical excitatory-inhibitory interplay carries high-contrast information (complementary to thalamic narrowband), so the E-I gamma band is the vehicle for encoding distinct visual features. [c]
21. **Quast et al. (2023)**. Rapid synaptic and gamma rhythm signature of mouse critical period plasticity. *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.2123182120
    Computational TC model shows an interneuronal (PV) gamma rhythm dissociates thalamic input from cortical spiking, driving spike-timing-dependent plasticity that opens critical-period plasticity, using ING timing as the plasticity gate. [s]
22. **Cattani et al. (2023)**. Basolateral amygdala oscillations enable fear learning in a biophysical model. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/2023.04.28.538604
    Preprint of the BLA fear-learning model showing interneuron-generated theta-gamma rhythms shape spike-timing-dependent plasticity, so the E-I rhythms enable associative learning by organizing precise spike timing. [c]
23. **Cattani et al. (2024)**. Basolateral amygdala oscillations enable fear learning in a biophysical model. *eLife*. doi:10.7554/elife.89519
    A biophysical BLA model shows PV/SOM/VIP interneurons generate theta-gamma rhythms that shape spike-timing-dependent plasticity to build a fear circuit, so interneuron-based rhythms enable learning by structuring precise spike timing. [c]
24. **Chalkiadakis et al. (2025)**. The role of feedforward and feedback inhibition in modulating theta-gamma cross-frequency interactions in neural circuits. *PLOS Computational Biology*. doi:10.1371/journal.pcbi.1013363
    Feedback-inhibition PING versus feedforward ING gamma set the directionality of theta-gamma cross-frequency coupling that organizes memory firing sequences, making the E-I gamma mechanism the vehicle for phase-organized timing. [s]
25. **Chalkiadakis et al. (2025)**. The Role of Feedforward and Feedback Inhibition in Modulating Theta-Gamma Cross-Frequency Interactions. *Qeios*. doi:10.32388/rih5uu
    This preprint shows PING feedback versus ING feedforward inhibition set the directionality of theta-gamma cross-frequency coupling that organizes neuronal firing sequences, making the E-I gamma motif the vehicle for phase-organized timing. [s]
26. **Tucker & Luu (2026)**. Excitatory-inhibitory resonance in cognition stabilizes synaptic traces in memory. *Cerebral Cortex*. doi:10.1093/cercor/bhag044
    Representation is carried by phase alignment of pyramidal-interneuron network gamma

(PING), and excitatory-inhibitory resonance across laminae stabilizes synaptic memory traces. [s]

Binding by synchrony (12)

1. **Singer (1993)**. Synchronization of Cortical Activity and Its Putative Role in Information Processing and Learning. *Annual Review of Physiology*. doi:10.1146/annurev.physiol.55.1.349
   Singer's foundational review arguing that synchronization of cortical activity (temporal correlation among assemblies) serves feature binding and information processing/learning, the origin of the binding-by-synchrony hypothesis. [m]
2. **Gray (1994)**. Synchronous oscillations in neuronal systems: Mechanisms and functions. *Journal of Computational Neuroscience*. doi:10.1007/bf00962716
   Reviews synchronous oscillations as the mechanism for perceptual feature binding, with the gamma rhythm arising from local excitatory-inhibitory network interactions grouping co-active neurons. [m]
3. **Traub et al. (1996)**. A mechanism for generation of long-range synchronous fast oscillations in the cortex. *Nature*. doi:10.1038/383621a0
   Describes an E-I network mechanism producing long-range zero-lag synchronous fast oscillations, supplying the synchronization substrate for binding features across distant cortical sites. [m]
4. **Eckhorn et al. (2004)**. Dynamic cortical cooperation related to visual perception. *Proceedings of the International Joint Conference on Neural Networks, 2003.*. doi:10.1109/ijcnn.2003.1223931
   Perceptually modulated gamma synchrony (generated by inhibitory feedback loops) among neural groups binds figure features while decoupling figure from background, supporting the binding-by-synchronization account of visual perception. [s]
5. **Tort et al. (2007)**. On the formation of gamma-coherent cell assemblies by oriens lacunosum-moleculare interneurons in the hippocampus. *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.0705708104
   Hippocampal network model shows O-LM interneurons produce gamma coherence that couples anatomically distinct pyramidal modules into gamma-coherent cell assemblies, an interneuron-driven synchrony mechanism for binding distant cells. [m]
6. **Arthur & Boahen (2007)**. Synchrony in Silicon: The Gamma Rhythm. *IEEE Transactions on Neural Networks*. doi:10.1109/tnn.2007.900238
   A silicon interneuron network synchronizing in the gamma band via shunting inhibition entrains model excitatory principal neurons to implement object binding, instantiating the interneuron-network gamma mechanism as the vehicle for feature binding. [s]
7. **Thomas & Rufin (2010)**. Gamma oscillations decompose the visual scene into object-based perceptual cycles: a computational model. *Journal of Vision*. doi:10.1167/10.7.1270
   A V1 model with feedback inhibition, refractory periods and gestalt lateral connections generates gamma cycles that both bind an object's features by co-firing and act as a winner-take-all gate, decomposing the scene into successive object-based perceptual cycles. [s]
8. **Folias et al. (2013)**. Synchronisation hubs in the visual cortex may arise from strong rhythmic inhibition during gamma oscillations. *European Journal of Neuroscience*.

doi:10.1111/ejn.12287
Strong rhythmic inhibition during gamma makes certain visual-cortex neurons synchronization hubs that entrain promiscuously with others, so the E-I gamma mechanism sets which neurons synchronize while preserving orientation selectivity. [c]

9. **Cornford et al. (2018)**. Dendritic NMDA receptors in parvalbumin neurons enable strong and stable neuronal assemblies. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/279505
   Dendritic NMDARs give PV cells supralinear (coincidence-sensitive) integration of feedback excitation so that reciprocal E-I recruitment strengthens and stabilizes principal-cell assemblies, the gamma-generating loop thereby binding neurons into robust assemblies and explaining sensory-gating deficits when it fails. [s]
10. **Barbosa et al. (2021)**. Across-area synchronization supports feature integration in working memory. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/2021.06.09.447667
    PING gamma (fast excitation, slow feedback inhibition) holds memory items at distinct phases, and weak cross-area coupling synchronizes the two PING networks to bind color and location features in working memory. [s]
11. **Barbosa et al. (2021)**. Across-Area Synchronization Supports Feature Integration in a Biophysical Network Model of Working Memory. *Frontiers in Neural Circuits*. doi:10.3389/fncir.2021.716965
    Reciprocally coupled color and location attractor networks, each generating gamma via recurrent excitation and feedback inhibition, bind features through cross-area gamma synchrony, making PING gamma the vehicle for feature binding in working memory. [s]
12. **Ursino & Pirazzini (2024)**. Construction of a Hierarchical Organization in Semantic Memory: A Model Based on Neural Masses and Gamma-Band Synchronization. *Cognitive Computation*. doi:10.1007/s12559-023-10202-y
    In a neural-mass attractor model of semantic memory, GABAergic-interneuron/E-I balance produces gamma synchronization that binds shared features into hierarchical concept representations, making the PING rhythm the vehicle for feature binding. [c]

Causal / optogenetic PV–gamma & circuit performance (17)

1. **Fuchs et al. (2007)**. Recruitment of Parvalbumin-Positive Interneurons Determines Hippocampal Function and Associated Behavior. *Neuron*. doi:10.1016/j.neuron.2007.01.031
   Genetically altering fast AMPA drive to parvalbumin interneurons disrupts hippocampal gamma and associated behavior, causally linking PV-interneuron recruitment (the PING driver) to network function. [m]
2. **Sohal et al. (2009)**. Parvalbumin neurons and gamma rhythms enhance cortical circuit performance. *Nature*. doi:10.1038/nature07991
   Optogenetically drives PV interneurons to generate gamma and shows this enhances cortical signal transmission by reducing noise, causally tying PV-gamma to improved circuit performance. [m]
3. **Cardin et al. (2009)**. Driving fast-spiking cells induces gamma rhythm and controls sensory responses. *Nature*. doi:10.1038/nature08002
   Optogenetic driving of fast-spiking PV interneurons selectively amplifies gamma and the

timing of sensory input relative to the gamma cycle controls the amplitude/precision of evoked responses, the founding causal FS-gamma demonstration. [m]

4. **Korotkova et al. (2010)**. NMDA Receptor Ablation on Parvalbumin-Positive Interneurons Impairs Hippocampal Synchrony, Spatial Representations, and Working Memory. *Neuron*. doi:10.1016/j.neuron.2010.09.017
   Cell-type-specific NMDA-receptor ablation on parvalbumin interneurons degrades hippocampal gamma synchrony, spatial coding, and working memory, giving causal evidence that PV-interneuron gamma from the E-I loop supports cognitive performance (title-inferred, empty abstract). [c]
5. **Yizhar et al. (2011)**. Neocortical excitation/inhibition balance in information processing and social dysfunction. *Nature*. doi:10.1038/nature10360
   Optogenetically elevating cellular E/I balance in prefrontal cortex raises 30-80 Hz gamma power and causally impairs information processing and social behavior, tying inhibition-generated gamma to circuit and behavioral performance. [c]
6. **Carlén et al. (2012)**. A critical role for NMDA receptors in parvalbumin interneurons for gamma rhythm induction and behavior. *Molecular Psychiatry*. doi:10.1038/mp.2011.31
   Deletes NMDA receptors selectively in PV interneurons, causally disrupting PV-driven gamma induction and producing specific cognitive deficits, directly linking the PV-gamma E-I mechanism to behavioural performance. [m]
7. **Siegle, Pritchett & Moore (2014)**. Gamma-range synchronization of fast-spiking interneurons can enhance detection of tactile stimuli. *Nature Neuroscience*. doi:10.1038/nn.3797
   Optogenetic FS-gamma enhances detection of less-salient tactile stimuli specifically when excitation arrives in the 20-25 ms window after FS synchronization, causally tying the interneuron-set inhibitory window to perceptual performance. [m]
8. **Cho et al. (2015)**. Gamma Rhythms Link Prefrontal Interneuron Dysfunction with Cognitive Inflexibility in Dlx5/6+/− Mice. *Neuron*. doi:10.1016/j.neuron.2015.02.019
   Optogenetically driving prefrontal fast-spiking interneurons at gamma restores cognitive flexibility in Dlx5/6 mutant mice, causally linking FSIN-generated gamma to cognition. [m]
9. **Kim et al. (2016)**. Prefrontal Parvalbumin Neurons in Control of Attention. *Cell*. doi:10.1016/j.cell.2015.11.038
   Optogenetically silencing vs gamma-synchronizing prefrontal FS-PV interneurons causally impairs vs improves attentional behavior, tying PV-driven gamma and pyramidal phase-locking to attention. [m+s]
10. **Shin & Moore (2019)**. Persistent Gamma Spiking in SI Nonsensory Fast Spiking Cells Predicts Perceptual Success. *Neuron*. doi:10.1016/j.neuron.2019.06.014
    Claims perceptual detection success is predicted by sustained gamma-regular spiking of a nonsensory FS interneuron subtype, casting FS-mediated E-I gamma as the temporal reference whose regularity gates perceptual performance. [m]
11. **Tan et al. (2019)**. Gamma oscillations in somatosensory cortex recruit prefrontal and descending serotonergic pathways in aversion and nociception. *Nature Communications*. doi:10.1038/s41467-019-08873-z

Optogenetic activation of S1 PV interneurons induces gamma that causally enhances nociceptive sensitivity and aversive behavior, tying PV-driven gamma to a behavioural pain-processing outcome. [s]

12. **Cho et al. (2019)**. Interhemispheric gamma synchrony between parvalbumin interneurons supports behavioral adaptation. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/784330
    Out-of-phase optogenetic disruption of parvalbumin-interneuron ~40 Hz interhemispheric synchrony causally impairs behavioral adaptation, showing PV-driven gamma synchrony is required for prefrontal communication supporting learning. [c]
13. **Cho et al. (2020)**. Cross-hemispheric gamma synchrony between prefrontal parvalbumin interneurons supports behavioral adaptation during rule shift learning. *Nature Neuroscience*. doi:10.1038/s41593-020-0647-1
    Cross-hemispheric gamma synchrony between prefrontal PV interneurons is causally required (out-of-phase optogenetic disruption causes perseveration) for behavioral adaptation, casting PV-gamma coherence as the vehicle for flexible rule-shift learning. [m]
14. **Patrono et al. (2022)**. The role of optogenetic stimulations of parvalbumin-positive interneurons in the prefrontal cortex and the ventral hippocampus on an acute MK801 model of schizophrenia-like cognitive inflexibility. *bioRxiv (Cold Spring Harbor Laboratory)*. doi:10.1101/2022.03.25.485752
    Optogenetically driving PFC and ventral-hippocampal PV interneurons is used to rescue MK801-induced cognitive inflexibility, arguing that PV-generated prefrontal gamma (and its theta-coupled cross-regional synchronization) causally supports attentional set-shifting behavior. [s]
15. **Dalal & Haddad (2022)**. Upstream γ-synchronization enhances odor processing in downstream neurons. *Cell Reports*. doi:10.1016/j.celrep.2022.110693
    Optogenetically increasing mitral-cell gamma-synchrony (via GABAergic granule cells) causally enhances downstream piriform odor coding despite lower firing rates, giving direct causal evidence that inhibition-based gamma-synchrony improves inter-areal sensory transmission. [c]
16. **Ferguson, Glick & Huguenard (2023)**. Prefrontal PV interneurons facilitate attention and are linked to attentional dysfunction in a mouse model of absence epilepsy. *eLife*. doi:10.7554/elife.78349
    Gamma-frequency optogenetic stimulation of prefrontal PV interneurons rescues attention deficits in Scn8a+/- mice, giving causal evidence that PV-driven gamma from the E-I loop underlies attentional performance. [s]
17. **Knowles (2024)**. Attentional Deficits and Absence Epilepsy: A Tale of 2 Interneuronopathies. *Epilepsy Currents*. doi:10.1177/15357597241251709
    Attention deficits track PV-interneuron hypoactivity and reduced cue-evoked gamma, and gamma-frequency optogenetic PVIN stimulation rescues attentional performance, making the PING-type PV->pyramidal gamma loop the causal vehicle for attention. [s]

Reviews & theory (mechanism → function) (18)

1. **Buzsáki & Chrobak (1995)**. Temporal structure in spatially organized neuronal ensembles: a role for interneuronal networks. *Current Opinion in Neurobiology*.

doi:10.1016/0959-4388(95)80012-3
Reviews how interneuronal networks impose temporal structure on spatially organized ensembles, arguing inhibition-based oscillations provide the timing reference for ensemble coding. [m]

2. **Jefferys, Traub & Whittington (1996)**. Neuronal networks for induced '40 Hz' rhythms. *Trends in Neurosciences*. doi:10.1016/s0166-2236(96)10023-0
Reviews simulations showing interneuron and E-I networks generate induced 40 Hz rhythms, linking that mechanism to higher functions such as feature binding and lower-level phase coding. [m+s]
3. **Whittington et al. (2000)**. Inhibition-based rhythms: experimental and mathematical observations on network dynamics. *International Journal of Psychophysiology*. doi:10.1016/s0167-8760(00)00173-2
Reviews experimental and mathematical work on inhibition-based (interneuron-network) rhythm generation, establishing the E-I/inhibitory mechanism that underlies gamma's temporal-coordination functions. [m]
4. **Whittington & Traub (2003)**. Interneuron Diversity series: Inhibitory interneurons and network oscillations in vitro. *Trends in Neurosciences*. doi:10.1016/j.tins.2003.09.016
Reviews how mutually connected inhibitory interneuron networks generate gamma/beta rhythms in vitro, framing the interneuron E-I mechanism as the substrate for temporal coordination of cortical activity. [m]
5. **Tiesinga & Sejnowski (2009)**. Cortical Enlightenment: Are Attentional Gamma Oscillations Driven by ING or PING?. *Neuron*. doi:10.1016/j.neuron.2009.09.009
Perspective dissecting whether attentional gamma is generated by ING vs PING interneuron mechanisms, i.e. which E-I circuit motif underlies the attention-related gamma rhythm. [m+s]
6. **Wang (2010)**. Neurophysiological and Computational Principles of Cortical Rhythms in Cognition. *Physiological Reviews*. doi:10.1152/physrev.00035.2008
Wang's review synthesizing how gamma and theta rhythms arise from interneuronal networks and reciprocal excitatory-inhibitory loops and how these synchronous rhythms sculpt temporal coordination across cognitive functions. [m+s]
7. **Whittington et al. (2011)**. Multiple origins of the cortical gamma rhythm. *Developmental Neurobiology*. doi:10.1002/dneu.20814
Reviews how inhibition-based (interneuron/E-I) gamma provides shared temporal structure and argues different interneuron mechanisms of gamma generation carry distinct functional correlates including assembly formation and inter-regional communication. [m]
8. **Uhlhaas et al. (2011)**. A new look at gamma? High- (>60 Hz) γ-band activity in cortical networks: Function, mechanisms and impairment. *Progress in Biophysics and Molecular Biology*. doi:10.1016/j.pbiomolbio.2010.10.004
Review arguing that high-frequency (>60 Hz) gamma generated by the E-I interneuron loop supports cortical computation and is disrupted in disease, surveying mechanism-to-function links across several cognitive domains. [c]
9. **Ainsworth et al. (2012)**. Rates and Rhythms: A Synergistic View of Frequency and Temporal Coding in Neuronal Networks. *Neuron*. doi:10.1016/j.neuron.2012.08.004

Reviews how inhibition-based rhythms including gamma synergistically combine rate and temporal (phase) coding, arguing the E-I oscillation structures information across frequency and timing. [m]

10. **Buzsáki & Wang (2012)**. Mechanisms of Gamma Oscillations. *Annual Review of Neuroscience*. doi:10.1146/annurev-neuro-062111-150444
    Canonical review establishing that gamma rhythmogenesis is inextricably tied to perisomatic inhibition emerging from coordinated excitation-inhibition, framing the mechanism that later functional claims depend on. [m]
11. **Bosman, Lansink & Pennartz (2014)**. Functions of gamma-band synchronization in cognition: from single circuits to functional diversity across cortical and subcortical systems. *European Journal of Neuroscience*. doi:10.1111/ejn.12606
    Review arguing that inhibition-excitation gamma generation arises from a limited set of circuit motifs that support many cognitive functions (perception, attention, memory, motivation, behavioral control) rather than a single universal one. [s]
12. **Kann, Papageorgiou & Draguhn (2014)**. Highly Energized Inhibitory Interneurons are a Central Element for Information Processing in Cortical Networks. *Journal of Cerebral Blood Flow & Metabolism*. doi:10.1038/jcbfm.2014.104
    Reviews how energetically specialized fast-spiking PV interneurons provide the 'clockwork' rhythmic inhibition of the E-I loop that generates gamma as a central element for cortical information processing. [s]
13. **Womelsdorf et al. (2014)**. Dynamic circuit motifs underlying rhythmic gain control, gating and integration. *Nature Neuroscience*. doi:10.1038/nn.3764
    Review casting recurrent E-I interneuron circuit motifs as the substrate for rhythmic gain control, gating and temporal integration, linking the PING mechanism to multiple routing/gating functions. [m]
14. **Ray & Maunsell (2015)**. Do gamma oscillations play a role in cerebral cortex?. *Trends in Cognitive Sciences*. doi:10.1016/j.tics.2014.12.002
    Critically reviews whether E-I-generated gamma actually supports functions such as inter-areal communication and phase coding, weighing the mechanism-to-function link across attention, memory and routing accounts. [m]
15. **Pittman-Polletta et al. (2015)**. Brain Rhythms Connect Impaired Inhibition to Altered Cognition in Schizophrenia. *Biological Psychiatry*. doi:10.1016/j.biopsych.2015.02.005
    Review theorizing that fast-spiking-interneuron dysfunction degrades PING-generated gamma, and that this altered temporal structure links impaired inhibition to the cognitive/perceptual deficits of schizophrenia across multiple functions. [c]
16. **Sohal (2016)**. How Close Are We to Understanding What (if Anything) γ Oscillations Do in Cortical Circuits?. *The Journal of Neuroscience*. doi:10.1523/jneurosci.0990-16.2016
    Sohal reviews the many candidate mechanisms by which E-I gamma oscillations could modulate input responses, activity patterns and output efficacy, arguing about how the gamma mechanism links to cortical function across several proposed roles. [m]
17. **Han, Shapley & Xing (2022)**. Gamma rhythms in the visual cortex: functions and mechanisms. *Cognitive Neurodynamics*. doi:10.1007/s11571-021-09767-x

A review of gamma in visual cortex weighing candidate functions against E-I and thalamic generating mechanisms, spanning attention, encoding, and communication accounts of the rhythm. [c]

18. **Sohal (2022)**. Transforming Discoveries About Cortical Microcircuits and Gamma Oscillations Into New Treatments for Cognitive Deficits in Schizophrenia. *American Journal of Psychiatry*. doi:10.1176/appi.ajp.20220147
    Sohal's overview argues that PV-interneuron-generated cortical gamma, deficient in schizophrenia, causally supports cognition, citing mouse optogenetic studies where disrupting or enhancing synchronized gamma reproduces or ameliorates cognitive deficits. [m]

Appendix: adjacent critiques (not corpus members)

These are the major skeptical papers on gamma/synchrony function. They are *not* in the frozen corpus and are excluded from every count and recall figure above: each questions a functional claim without naming the E–I/PING loop as the generator, so it fails the mechanism-as-vehicle test. They are recorded here for the later review. The disposition tag marks whether the build surfaced and rejected the paper, or never reached it.

1. **Shadlen & Movshon (1999)**. Synchrony Unbound: A Critical Evaluation of the Temporal Binding Hypothesis. *Neuron*. doi:10.1016/s0896-6273(00)80822-3 (surfaced, judged out)
2. **Thiele & Stoner (2003)**. Neuronal synchrony does not correlate with motion coherence in cortical area MT. *Nature*. doi:10.1038/nature01285 (not surfaced)
3. **Palanca & DeAngelis (2005)**. Does Neuronal Synchrony Underlie Visual Feature Grouping?. *Neuron*. doi:10.1016/j.neuron.2005.03.002 (not surfaced)
4. **Ray & Maunsell (2010)**. Differences in Gamma Frequencies across Visual Cortex Restrict Their Possible Use in Computation. *Neuron*. doi:10.1016/j.neuron.2010.08.004 (surfaced, judged out)
5. **Merker (2013)**. Cortical gamma oscillations: the functional key is activation, not cognition. *Neuroscience & Biobehavioral Reviews*. doi:10.1016/j.neubiorev.2013.01.013 (not surfaced)
6. **Ray & Maunsell (2015)**. Do gamma oscillations play a role in cerebral cortex?. *Trends in Cognitive Sciences*. doi:10.1016/j.tics.2014.12.002 (surfaced, judged out)

Introduction

11 August 2026

This collection asks whether a pyramidal–interneuron network gamma (PING) loop can act as a structural sparsity constraint in a task-trained spiking network. The experiments separate three questions that are easy to muddle: what generates the rhythm, what sets the firing-rate floor, and whether the resulting sparse activity remains useful for classification. They also distinguish a genuine experimental dependency—one entry consuming another entry's checkpoints or measurements—from a looser conceptual dependency.

exp022 — Training-run guide
Link to exp022. The collection's operational hub owns the canonical training registry and produces the shared COBA and PING checkpoint banks.

Child experiments

1. **exp024 — Accuracy converges, firing rate does not**
   Link to exp024. Audits exp022′s per-epoch baseline histories, comparing convergence of classification accuracy and firing rate. **Manuscript:** supports the rate-attractor interpretation of Figure 3; it is not the plotted image source.
2. **exp025 — PING locks E rate ≈10× below COBA**
   Link to exp025. Uses exp022′s trained cells for the central comparison between loop-free COBA and gamma-gated PING. **Manuscript:** Figure 3.
3. **exp037 — PING tolerates 80% dropped spikes but collapses on added noise**
   Link to exp037. Applies spike-deletion and spike-addition perturbations to exp022′s trained cells. **Manuscript:** Figure 8.
4. **exp038 — Switching the loop on at inference cuts E rate ≈15×**
   Link to exp038. Transfers a trained loop-free baseline into a fresh inhibitory architecture, separating the loop's inference-time effect from training history. **Manuscript:** Figure 4.
5. **exp041 — E rate is affine in gamma frequency**
   Link to exp041. Uses exp022′s τ_{GABA} family to measure how inhibitory decay changes gamma frequency and excitatory rate. **Manuscript:** supplies the spiking comparison in Figure 2 and the complete Figure 6.
   1. **exp033 — Mean-field PING**
      Link to exp033. Compares the measured sweep with the mean-field model. **Manuscript:** mean-field panels in Figure 2.
   2. **exp042 — Timing perturbations**
      Link to exp042. Uses exp022 baseline checkpoints and exp041 timing measurements to separate inhibitory timing from inhibitory level. **Manuscript:** Figure 9.
   3. **exp046 — Per-cycle E-spike count**
      Link to exp046. Counts excitatory spikes within cycles measured by exp041. **Manuscript:** Figure 7.
6. **exp044 — Rate floor stable across a 20× Δt sweep**
   Link to exp044. Reads exp022′s timestep-sweep cells to test whether the rate floor is physical rather than a step-count artefact. **Manuscript:** Figure 10.

7. **exp048 — Temporal and spatial evidence limits of trained PING**
   Link to exp048. Loads the canonical trained PING checkpoint and varies presentation time, input rate, and evidence distribution. **Manuscript:** current source for Figures 11 and 12, pending replacement by exp082.
8. **exp049 — Gradient descent does not preserve a trainable PING loop**
   Link to exp049. Uses exp022′s initialisation-family cells to test whether training retains or dismantles the recurrent loop. **Manuscript:** Figure 5.
9. **exp082 — Variable-rate streaming with a spike-rate readout**
   Link to exp082. Consumes exp022′s variable-rate checkpoint bank. Exp080 and exp081 support its design conceptually but are not execution dependencies. **Manuscript:** planned replacement source for Figures 11 and 12 after results are complete.

exp023 — PING fundamentals
Link to exp023. Establishes the free-running excitatory–inhibitory gamma mechanism without training. **Manuscript:** Figure 1.

exp047 — Pool-size invariance requires inverse synaptic scaling
Link to exp047. Separates nominal total coupling from realised per-synapse strength while varying inhibitory-pool size.

exp054 — A PING rhythmicity metric
Link to exp054. Calibrates a rhythmicity statistic and identifies its low-rate and shared-input failure modes using independent simulations. **Manuscript:** coupling-plane and spiking panels in the Figure 2 composite.

exp080 — Empirical input-rate calibration for variable-rate PING training
Link to exp080. Calibrates transformations of sparse pixel intensities into variable Poisson input rates. It conceptually informs exp082 but supplies no runtime artifact.

exp081 — Linear-filter analysis of sparse conductance-driven pixel features
Link to exp081. Analyses how sparse conductance-driven pixel streams are filtered before and within the PING network. It conceptually informs exp082 but supplies no runtime artifact.

Bayesian Fundamentals

28 June 2026

A short primer for the vocabulary the neural-Bayesian papers in the Bayesian Literature Review lean on without redefining. The aim is not to teach Bayesian statistics from scratch — it is to fix notation, name the moves, and flag where the hard parts are, so a reader recognises each term on contact when they hit it in Ma, Fiser, Lengyel, or Echeveste.

The core equation

Given a generative model — a forward story for how unobserved causes θ produce observations x — Bayes' rule inverts the story:

$$p(\theta \mid x) = \frac{p(x \mid \theta)p(\theta)}{p(x)} \propto p(x \mid \theta)p(\theta) \quad (1)$$

- θ — the latent (unobserved) cause to be inferred (a tilt angle, a phoneme, a depth);
- x — the observation (the retinal image, the cochlear input);
- $p(\theta)$ — the **prior**, what was believed about θ before seeing x;
- $p(x \mid \theta)$ — the **likelihood**, how probable this x would be if θ were the cause;
- $p(\theta \mid x)$ — the **posterior**, the updated belief about θ after seeing x;
- $p(x) = \int p(x \mid \theta)p(\theta)\,\mathrm{d}\theta$ — the **evidence** (or marginal likelihood), the normaliser.

The proportionality at the right is doing most of the work in practice: every algorithm in the rest of this article exists to avoid computing $p(x)$ exactly.

Generative model vs recognition

A **generative model** is the brain's (or the modeller's) story about how the world produces the data: causes → observations. **Recognition** (or **inference**) is the reverse — recover $p(\theta \mid x)$ from x. These are different computations and the brain need not implement them the same way; a sensory area can run a fast amortised recognition map even if the underlying generative model is much richer. When neural-Bayesian papers say "the cortex represents the posterior", they mean the recognition output.

Marginalisation, and why it's the hard part

Anything you don't care about you must integrate out:

$$p(\theta_1 \mid x) = \int p(\theta_1, \theta_2 \mid x)\,\mathrm{d}\theta_2 \quad (2)$$

- θ_1 — the quantity of interest (the depth of a surface);
- θ_2 — the nuisance variables (lighting, surface reflectance) you marginalise over.

For high-dimensional θ this integral is intractable. Every approximate-inference algorithm below is, at heart, a different way to dodge it.

Conjugacy and the exponential family

A likelihood–prior pair is **conjugate** if the posterior has the same parametric form as the prior. Conjugate pairs (Gaussian–Gaussian, beta–Bernoulli, Dirichlet–multinomial) give closed-form posterior updates and are the only place Bayes is genuinely cheap. They live inside the **exponential family**:

$$p(x \mid \eta) = h(x) \exp(\eta^\top T(x) - A(\eta)) \quad (3)$$

- η — the **natural parameters**;
- $T(x)$ — the **sufficient statistics** (the only function of x the posterior cares about);
- $A(\eta)$ — the **log-partition** (normaliser), whose derivatives give the moments of T;
- $h(x)$ — a base measure.

This matters for cortex because the Probabilistic Population Code framework (Ma et al. 2006, in ar007) writes posteriors as exponential families whose natural parameters are population firing rates: $\eta \equiv \boldsymbol{r}$. Multiplying two PPCs (combining cues) becomes adding their firing rates — the framework's central, very pretty claim.

Three flavours of approximate inference

When the posterior is not conjugate, you approximate. Three families dominate; the neural-Bayesian papers each pick one:

- **Laplace approximation.** Find the posterior mode $\hat{\theta}$, fit a Gaussian whose covariance is $\left(-\nabla^2 \log p(\theta \mid x)\right)^{-1}$ at the mode. Cheap, local, breaks down when the posterior is multimodal.
- **Variational inference (VI).** Pick a tractable family $q_\varphi(\theta)$ and minimise $\mathrm{KL}\big(q_\varphi(\theta) \parallel p(\theta \mid x)\big)$ in φ. Turns inference into optimisation. Fast, scalable, but biased — the bias is whatever p has that q cannot represent.
- **Markov chain Monte Carlo (MCMC).** Construct a Markov chain whose stationary distribution *is* the posterior, then run it. Unbiased in the limit, slow to converge. Variants: Metropolis–Hastings, Gibbs sampling, Langevin dynamics (gradient + noise), Hamiltonian Monte Carlo (gradient + momentum). The Lengyel-group "sampling cortex" papers in ar007 argue that cortical dynamics are a biological MCMC chain — Aitchison & Lengyel cast E/I circuits as a Hamiltonian sampler, with the inhibitory population playing the momentum role.

KL divergence

The asymmetric "distance" between two distributions:

$$\mathrm{KL}(q \parallel p) = \int q(\theta) \log \frac{q(\theta)}{p(\theta)} \, \mathrm{d}\theta \geq 0, \quad \text{with equality iff } \; q = p. \quad (4)$$

Asymmetric because $\mathrm{KL}(q \parallel p) \neq \mathrm{KL}(p \parallel q)$ in general. VI minimises $\mathrm{KL}(q \parallel p)$ (forward KL) which makes q *mode-seeking* (it concentrates on a single mode rather than averaging across modes); some methods minimise $\mathrm{KL}(p \parallel q)$ (reverse KL) which makes q *mass-covering*. The choice shows up in what the approximate posterior looks like.

Posterior summaries (the things a brain might actually read out)

A posterior is a distribution; downstream computation usually wants a number or two. The standard summaries:

- **MAP estimate.** The mode: $\theta_{\mathrm{MAP}} = \arg\max_\theta p(\theta \mid x)$. A single best guess.
- **Posterior mean.** $\mathbb{E}[\theta \mid x]$. The Bayes-optimal point estimate under squared-error loss.

- **Posterior variance / credible interval.** The width — how *uncertain* the inference is. The whole point of being Bayesian rather than maximum-likelihood; if downstream computation needs to know "should I bet on this?" it needs the spread, not just the mean.

The neural-Bayesian split is partly about how the cortex represents this last one — as a population gain (PPC) or as trial-to-trial variability (sampling).

How this lands in the cortex

Three claims recur in the reading list, in roughly this order of strength:

1. **Cortical responses look Bayesian behaviourally.** Cue integration, prior-weighted perception, multisensory fusion — all match Bayes-optimal predictions in many tasks. The papers in ar007 take this as given.
2. **The cortical *representation* of probability is one of two flavours.** Either parametric (firing rates encode posterior parameters — PPC) or sample-based (instantaneous activity is itself a sample). These are different empirical predictions about neural variability, and the camps disagree.
3. **The cortical *dynamics* implement the inference algorithm.** The sampling camp claims recurrent E/I circuits run Langevin or HMC; the PPC camp claims linear combinations of population activity perform the algebra. Echeveste et al. 2020 is the cleanest version of the dynamical claim.

This article stops short of those claims — see ar007 for the reading list that develops them.

A Python graph language for the SNN tool

30 July 2026

Abstract

Translating a circuit idea into *tools/snn* is cumbersome. A paper may describe a small motif in a paragraph, while its implementation requires manual work across CLI flags, model construction, tensor dimensions, training choices, recordings, and artifact handling. This proposal introduces *snnlang*: a typed Python library for constructing a spiking computation graph and, when needed, a narrow training specification. One compile operation validates them and writes a portable bundle containing *graph.json* and optional *training.json*. *tools/snn* executes that bundle through a small Python API wrapped by its CLI.

The proposal is deliberately smaller than a language for whole experiments, but it is not restricted to the present `COBANet` architecture. Experiment runners continue to own hypotheses, condition and seed grids, custom interventions, derived analysis, figures, and publication. An audit of the current code identifies three workloads that should test the design: trained MNIST and SHD classifiers, including deeper layers; coupled untrained E/I circuits with population-level recordings; and online confidence feedback that modulates g_L to trade decision speed for additional PING cycles. The bundle is the canonical reproducibility boundary, while a small importable API preserves exploratory PyTorch flexibility. *tools/snn* lowers graphs into vectorized, compiled PyTorch rather than interpreting them. The implementation order is: draft the graph language, upgrade *tools/snn* to execute it, run one real experiment through both, and then prove that the combined system is flexible enough for the confidence experiment.

1. The existing separation of concerns

The repository intends to contain two different layers. *tools/snn*, documented in ar011, is the reusable simulation and training engine. Files under *experiments/* are scientific runners. The desired boundary is subprocesses and files, but the present code does not consistently maintain it.

```
experiment runner
      |
      | CLI invocation
      v
tools/snn/tool.py
      |
      | config, metrics, traces, weights
      v
experiment runner
      |
      | analysis, figures, numbers.json
      v
writings/expNNN.typ
```

This firewall is useful when it exists. It keeps reusable model science out of paper-specific code and prevents the simulator from accumulating every protocol ever used in the lab. The

leaks are diagnostic: they identify missing reusable execution contracts more clearly than an abstract architecture exercise can.

1.1 What tools/snn owns

tools/snn owns operations that remain meaningful across experiments:

- neuron and synapse dynamics;
- graph execution and numerical integration;
- the standard training loop and surrogate-gradient implementation;
- checkpoint loading and saving;
- reusable input, recording, and perturbation primitives;
- generic rates, accuracy, and run metrics;
- deterministic seeding and run provenance;
- stable machine-readable outputs.

Its current use of *torch.compile* remains an internal execution detail. *snnlang* does not manage compilation caches or accelerator-specific artifacts.

1.2 What experiment runners own

A runner owns decisions whose meaning comes from one scientific question:

- the hypothesis and condition labels;
- parameter and seed grids;
- job ordering, caching, resumption, and dispatch;
- paper-specific data preparation;
- custom intervention streams built from earlier outputs;
- derived statistics and success criteria;
- figure design and *numbers.json*;
- staging and atomic publication.

Substantial runner code is not automatically a failure of abstraction. Code should move into a reusable tool only after a second use demonstrates a stable operation.

1.3 What the runners reveal

Existing runners make the boundary concrete. *exp022* defines a canonical family of trained MNIST cells. *exp041* computes spectra, gamma peaks, fits, and figures from tool outputs. *exp042* constructs experiment-specific inhibitory override streams from a reusable simulator primitive. *exp054* and *exp058* use untrained networks to study connectivity, synchrony, rhythmicity, and perturbation growth.

The newer SHD runners also expose failures in the current boundary. They import the CLI module in-process, replace the dataset directory, and construct networks directly for evaluation. *exp069* temporarily replaces `MetricsJsonl.write`, inspects its caller's local variables to obtain the live network, and saves a validation-selected checkpoint. This is ingenious in the same sense that opening a tin with a screwdriver is ingenious: it works, but it is evidence that the correct tool is missing.

The audit therefore identifies four engine seams needed before a general graph executor:

- an explicit data binding for dense samples and event streams, instead of fixed dataset paths;

- a standard checkpoint-selection policy or stable epoch event contract;
- evaluation of supplied event datasets without direct model construction;
- named recordings from every layer and population, rather than a privileged "primary" hidden layer.

These cases support one rule:

> The tool should understand one graph execution. The runner should understand why several executions constitute an experiment.

1.4 What the current model actually is

The present `COBANet` is a useful but narrow execution model:

- it builds a feedforward list of hidden E/I layers;
- E spikes alone feed the next hidden layer;
- each layer owns W_{EE}, W_{EI}, W_{IE}, and W_{II} recurrence;
- rate, mean-membrane, and cumulative-potential readouts are hard-coded modes;
- two hidden layers are already exercised by *exp071*;
- `train_leak` learns a static bounded leak per neuron, not an online feedback law;
- recording can capture all layers, although several consumers select only the deepest layer.

This means "support deeper networks" is not a distant grammar problem. The first graph schema should represent the existing sequential stack, but its ontology should be named populations and projections rather than `COBANet` constructor arguments. Coupling two independently named circuits is a genuinely new backend capability, even when neither circuit is trained. It is acceptable for the first compiler to emit graphs that the current backend reports as unsupported; the language and engine capability levels must be explicit rather than artificially identical.

1.5 Data is an execution binding, not graph structure

The standard trainer currently knows MNIST and SHD. MNIST is dense and Poisson encoded; SHD is an event stream binned lazily. The generic inference path still assumes 784 dense inputs, so SHD experiments work around it.

graph.json should declare typed input ports and output dimensions. *TrainSpec* should declare the target interface and encoder class. A runner should supply a small resolved data binding that maps actual train, validation, or evaluation sources onto those ports. Dataset splitting, sealing, and scientific cohort selection remain runner responsibilities. This avoids both extremes: hard-coding MNIST and SHD into the graph, or inventing a universal dataset language.

2. The proposed boundary

The proposal introduces one Python package with two related authoring objects:

```
snnlang.Network
snnlang.TrainSpec
```

Network describes forward computation. *TrainSpec* is an optional, narrow configuration for the standard trainer. It refers to parameters and outputs in a particular *Network*, but it is not part of that network and is not a second independent language.

One compile operation produces a bundle:

```
Python authoring
  Network
  optional TrainSpec
        |
        | snnlang.compile(...)
        v
ping.bundle/
  graph.json
  training.json
  manifest.json
        |
        | tools/snn API or CLI + data binding
        v
run artifacts
```

graph.json is the resolved executable graph. *training.json* records standard training policy and the digest of the exact graph against which it was validated. *manifest.json* binds the files into one convenient invocation unit.

A separate, resolved *data.json* is supplied when training or evaluating. It contains paths and representation metadata, not dataset contents. It belongs to an execution and may be replaced while the graph remains unchanged.

tools/snn depends on the versioned graph and training schemas, not on the *snnlang* Python package. Given an archived bundle, it must be possible to replay the run without installing the compiler that originally authored it.

2.1 Same repository, separate libraries

snnlang should begin in the same repository and Python environment as *tools/snn*, but as a sibling library:

```
tools/
    snn/
    snnlang/
schemas/
    snn-graph-v1.schema.json
    snn-training-v1.schema.json
```

Keeping both sides together makes early schema changes cheap. Keeping their package boundaries separate prevents the authoring library from becoming simulator internals. A separate repository or published project is justified only after another repository, simulator, or release cadence actually needs it.

3. The smallest useful authoring model

The first authoring surface is Python, not a custom textual grammar. Python already supplies composition, functions, loops, autocomplete, testing, and debugging.

```
import snnlang as snn
from snnlang import training

net = snn.Network("ping_classifier")

E = net.population(
    "E",
    size=1024,
    neuron=snn.COBA_LIF(...),
)
I = net.population(
    "I",
    size=256,
    neuron=snn.COBA_LIF(...),
)

ei = net.connect(
    E.spikes,
    I.excitatory,
    name="E_to_I",
    synapse=snn.AMPA(tau=...),
    weight=snn.Normal(0.5, 0.05),
    constraint=snn.NonNegative(),
)
ie = net.connect(
    I.spikes,
    E.inhibitory,
    name="I_to_E",
    synapse=snn.GABA(tau=9 * snn.ms),
    weight=snn.Normal(1.0, 0.1),
    constraint=snn.NonNegative(),
)

logits = snn.readouts.MeanVoltage(
    source=E.spikes,
    classes=10,
    tau=20 * snn.ms,
    name="classifier",
)

net.output("class_logits", logits)
net.expose(E.spikes, I.spikes)
```

The graph contains populations, projections, ordinary transformations, parameters, outputs, and observables. It does not contain experiment conditions, sweeps, losses, optimizers, or figures.

3.1 Components and layers are authoring conveniences

A reusable component may shorten construction:

```
cell = snn.components.ping(
    net,
    name="cell",
    n_e=1024,
    n_i=256,
    tau_gaba=9 * snn.ms,
)
```

Components and layers expand into the same small graph vocabulary. The serialized graph may retain group metadata for reports, but it need not introduce a special executable `layer` primitive.

The same rule applies to readouts. A helper such as:

```
logits = snn.readouts.SpikeRate(
    source=cell.E.spikes,
    classes=10,
    name="class_logits",
)
```

expands into ordinary population, projection, reduction, and output operations. Authoring functions execute while constructing the graph; the compiler serializes their result rather than the Python callable. The first graph schema therefore has no special `head` ontology. Named outputs provide the interface needed by inference and training.

3.2 Readouts are concise components with precise semantics

Readout switching is routine experimental work and must be a local edit. The initial library should include:

```
snn.readouts.MeanVoltage(...)
snn.readouts.FinalVoltage(...)
snn.readouts.SpikeCount(...)
snn.readouts.SpikeRate(...)
snn.readouts.CumulativePotential(...)
```

`MeanVoltage` expands into a trainable projection, a non-spiking stateful layer, and a temporal mean of its membrane voltage. The non-spiking layer remains explicit in the compiled graph and checkpoint even when hidden by the concise authoring helper.

`SpikeCount` means

$$z_c = \sum_t s_{c(t)}.$$

`SpikeRate` is distinct:

$$z_c = \frac{\sum_t s_{c(t)}}{T\Delta t}.$$

For padded or variable-duration samples it must use the valid-time mask independently for each sample. “Rate” without a declared duration and unit is rejected as ambiguous. Common window specifications should cover the full trial, a post-transient interval, or the final duration without requiring a Python callback.

Users may define their own authoring helpers from standard operations:

```
def my_readout(source, classes):
    x = snn.ops.sum(source, over="time")
    x = snn.ops.normalise(x)
    return snn.ops.linear(x, size=classes)
```

This remains portable because the function expands at compile time. Arbitrary PyTorch executed during the forward pass is a different extension level and is not embedded as a callable in JSON.

3.3 Parameters are not intrinsically selected for this run

The graph distinguishes a *Parameter* from a *Constant* and records structural constraints such as non-negativity, sparsity masks, or tying. It does not permanently declare that every parameter must be updated.

Selection belongs to a particular training specification:

```
train = snn.TrainSpec(
    objectives=[
        training.CrossEntropy(
            prediction=net.outputs["class_logits"],
            target="digit",
        ),
    ],
    parameter_groups=[
        training.ParameterGroup(
            [ei.weight, ie.weight],
            lr=1e-4,
        ),
        training.ParameterGroup(
            logits.parameters,
            lr=1e-3,
        ),
    ],
    regularizers=[
        training.UpperRatePenalty(
            signal=E.spikes,
            threshold=1.0,
            strength=1e-4,
```

```
        ),
    ],
    optimizer=training.AdamW(),
    epochs=50,
)
```

The same graph can support readout-only training, recurrent fine-tuning, or inference without changing its structural identity.

3.4 Forward and backward concerns

Anything executed during inference belongs in the graph: populations, reductions, projections, classifier parameters, and named outputs. Anything used only to calculate or apply gradients belongs in *TrainSpec*: objectives, targets, parameter groups, regularizers, surrogate choice, clipping, and backward-only stop boundaries.

Checkpoints remain separate evolving state. They contain realised parameter values and, when resuming training, optimizer state. A checkpoint may initialise, resume, or partially map onto a graph, but its path is not part of the immutable graph.

3.5 Online feedback belongs to the combined execution system

Confidence-controlled leak is not a training recipe. At each timestep it computes evidence from the current readout, derives confidence, and modulates a population parameter before a later state update. The combined *snnlang* plus *tools/snn* system must support that causal loop, but version one need not give confidence or g_L modulation dedicated language constructs.

One eventual declarative spelling could be:

```
evidence = snn.readouts.cumulative(cell.E.spikes)
confidence = snn.signals.max_probability(evidence)

snn.controls.bounded_leak(
    source=confidence,
    target=cell.E,
    low_confidence_tau=30 * snn.ms,
    high_confidence_tau=10 * snn.ms,
    delay=1 * snn.step,
)
```

The spelling is provisional and is not a version-one requirement. Three implementation routes are legitimate:

- ordinary graph operations, if the initial vocabulary can express the loop cleanly;
- a generic stateful controller or modulatory-node extension implemented by *tools/snn*;
- custom experiment code using a small, stable runtime hook around a compiled graph.

Whichever route is chosen must state the source signal, transform, target, bounds, initial state, and whether the effect is immediate or delayed. A next-timestep default avoids an implicit algebraic loop. If controller parameters later become trainable, *TrainSpec* can select them like any other parameter.

The third route does not mean reaching into model locals or monkey-patching the training loop. It means an intentional engine extension point. The graph and checkpoint remain portable; the run manifest records the controller implementation and configuration. If the same controller pattern recurs, it can then be promoted into snnlang.

4. What remains outside snnlang

Neither *Network* nor *TrainSpec* should initially describe:

- experimental conditions or hypothesis labels;
- parameter and seed sweeps;
- matched-control searches;
- multi-stage curricula or alternating optimizers;
- arbitrary Python callbacks;
- experiment-specific adaptive interventions driven by intermediate results;
- paper-specific metrics;
- figures and publication;
- RunPod or Modal orchestration;
- automatic prose or claim generation.

A custom experiment may use *graph.json* with custom Python training code and omit *training.json*. A feature enters *TrainSpec* only when the standard *tools/snn* trainer supports it and repeated experiments share it.

A paper-specific rule that decides which condition to run next remains in the runner. A within-trial control loop must execute inside the simulator runtime, whether authored directly in the graph or attached through a stable extension point.

5. Compilation and static analysis

Compilation is pure and deterministic:

```
bundle = snn.compile(
    net,
    training=train,
    target="tools/snn",
)
bundle.write("ping.bundle")
```

It performs no simulation, accelerator allocation, realised random initialization, or mutation of *tools/snn* globals. Python object references become stable graph identifiers at serialization.

5.1 Hard validation

The early compiler can provide substantial value before execution:

- schema and version validation;
- unique names and resolved references;
- tensor and projection shapes;
- physical units;
- port compatibility;
- graph reachability from inputs to outputs;

- disconnected populations and unused projections;
- Dale and sign-constraint compatibility;
- parameter inventory and optimizer-group membership;
- objective and target compatibility;
- differentiable reachability from each objective;
- surrogate-gradient coverage on objective paths;
- feedback-cycle delays and a deterministic within-step update order;
- controller output units, target compatibility, and declared bounds;
- checkpoint names and shapes;
- backend capability checks;
- deterministic canonical serialization.

Example diagnostics should identify the object and failed relation:

```
error E203: projection "E_to_I" has shape [800, 200]
  expected [1024, 256] from E.spikes -> I.excitatory

error E501: selected parameter "I_to_E.weight"
  has no differentiable path to objective "classification"
```

5.2 Reports and estimates

The analyser can also report:

- population and projection tables;
- strongly connected components and recurrent motifs;
- feedforward, recurrent, and modulatory paths between named populations;
- parameter counts and selected versus frozen parameters;
- expected sparse edge counts and fan-in;
- approximate parameter, optimizer, state, recording, and BPTT memory;
- semantic differences between two bundles;
- whether two runs share a graph but differ in training policy;
- whether a bundle contains unresolved environment-dependent defaults.

Numerical risk checks should be warnings rather than proofs:

```
warning W311: tau_gaba=0.2 ms is only two timesteps
  at dt=0.1 ms; the decay may be poorly resolved
```

5.3 Visual graph reports

snnlang should optionally render a compiled bundle as a polished circuit diagram. This is a compiler report, not a simulator feature:

```
bundle.visualise(
    "network.svg",
    view="circuit",
)
bundle.visualise(
    "network.png",
    view="circuit",
    scale=2,
)
```

The implementation should generate styled DOT from *graph.json*, use Graphviz for layout, and treat SVG as the canonical visual output. PNG and PDF are derived publication formats. Graphviz is responsible for ranks, routing, and crossing reduction; *snnlang* remains responsible for visual meaning.

The visual grammar should be stable:

- rounded cards for populations, with size and neuron model;
- restrained, consistent colours for excitatory, inhibitory, input, output, and modulatory nodes;
- solid excitatory projections, inhibitory bar endings, and dashed modulatory paths;
- softly bounded clusters for layers, PING cells, and user-defined components;
- feedback routed separately from the primary feedforward direction;
- optional edge width or annotation for density, fan-in, delay, or weight summary;
- subtle marks for trainable parameters and stop-gradient boundaries.

One overloaded picture is not the goal. The renderer should offer at least:

- `circuit`, a paper-ready population and projection view;
- `training`, showing outputs, objectives, trainable groups, and gradient boundaries;
- `expanded`, showing lower-level operations and stateful nodes for debugging.

Components are collapsed by default and expandable on request. Parallel projections may be grouped, default parameters suppressed, and a selected path highlighted. If a graph is too dense for a legible static image, the renderer should warn and require filtering or collapsing rather than emit decorative spaghetti. Layout, colours, fonts, node ordering, and identifiers must be deterministic so an unchanged bundle produces an unchanged diagram.

The canonical SVG should retain stable object identifiers and classes. This permits tooltips or interactive inspection later without changing the graph schema. Visualisation failure must never make an otherwise valid bundle unexecutable.

5.4 What static analysis does not promise

The early analyser does not predict whether gamma emerges, its frequency, a Hopf threshold, firing rates, accuracy, or successful optimization. Those are dynamical claims. Later restricted analysers may attach approximate semantics to supported motifs, but every result must state its assumptions and unsupported components.

6. Runtime and CLI boundary

The bundle is the canonical reproducibility boundary, not the only execution route. *tools/snn* should expose one small request-based Python API:

```
model = tools_snn.build(graph)
result = tools_snn.train(
    graph,
    training,
    data,
    request,
)
result = tools_snn.simulate(
    graph,
    data,
    request,
)
```

The CLI is a thin wrapper over the same functions. Ordinary experiment runners should receive a typed subprocess helper when they want process isolation:

```
from experiments.helpers.snn import run_training

run_training(
    bundle=bundle_path,
    seed=seed,
    out_dir=cell_dir,
)
```

The helper invokes:

```
uv run python tools/snn/tool.py train \
  --bundle ping.bundle \
  --data data.json \
  --seed 42 \
  --out-dir cell/
```

Inference needs only the graph and optional checkpoint:

```
uv run python tools/snn/tool.py sim \
  --graph ping.bundle/graph.json \
  --weights weights.pth \
  --out-dir inference/
```

Scientific graph and standard training settings live in validated files rather than long argument lists. The data binding carries resolved sources. CLI arguments carry paths, seeds, output locations, and lifecycle controls.

The process boundary remains valuable because a fresh process contains failures, accelerator memory, compiler state, and legacy globals. The importable API remains available for

notebooks, debugging, composition into custom training code, and experiments needing deliberate runtime extensions. Runners must not obtain that flexibility by importing the CLI module, inspecting stack frames, or monkey-patching engine internals.

6.1 Three extension levels

The system supports increasing flexibility with decreasing portability:

1. **Authoring components.** Python functions compose standard snnlang operations and expand before serialization. This is the preferred route for readouts and circuit templates.
2. **Registered backend operations.** The graph records a versioned operation identifier and configuration; *tools/snn* supplies its PyTorch implementation. The manifest records the required extension.
3. **Direct Python execution.** Exploratory code uses the importable API with a custom `torch.nn.Module`, controller, or training loop. The run records its source identity, configuration, environment, and compiled graph.

Every standard experiment should replay from an archived bundle. Experimental Python extensions are allowed when recorded explicitly; they should be promoted into a portable operation only after their interface stabilizes.

6.2 PyTorch performance is non-negotiable

snnlang is an authoring language and intermediate representation. It is not a runtime interpreter. *tools/snn* lowers a validated graph into coarse, vectorized PyTorch modules before simulation:

```
graph.json
    -> tools_snn.build(...)
    -> torch.nn.Module
    -> torch.compile
    -> CUDA, MPS, or CPU execution
```

Populations are batched tensors; projections become dense, sparse, or structured tensor operations; recurrent state remains in tensors; and training uses PyTorch autograd with surrogate gradients. Compatible operations may be fused, and future Triton or custom CUDA kernels remain backend implementation choices.

The runtime must not execute a Python loop over graph nodes inside every timestep. Arbitrary Python callbacks may cause graph breaks and are an explicitly slower exploratory path. Stateful controllers intended for production execution should be PyTorch modules with fixed tensor interfaces.

Backend conformance includes performance. Each reference graph is benchmarked against its specialized legacy implementation after warm-up and compilation. The initial target is no more than approximately 5–10% steady-state overhead for an equivalent graph, with peak memory and compilation time reported separately. A pleasant API does not compensate for a materially slower simulator.

Existing flag-driven experiments remain supported. The manifest path is additive:

```
legacy flags -> compatibility adapter -\
                                       +-> ExecutionSpec
graph bundle -> bundle loader --------/
```

During migration, *ExecutionSpec* may select either the legacy `COBANet` builder or the graph-native executor. The target is one graph-native PyTorch execution core with the legacy CLI retained as a compatibility frontend. This permits gradual adoption without rewriting the historical experiment corpus.

7. Staged implementation

Implementation is divided into cumulative, goal-sized milestones. Each milestone states separate *snnlang* and *tools/snn* deliverables, the legacy behaviour that must remain unchanged, and executable acceptance tests. "Implement up to milestone N" therefore means: inspect the repository's declared milestone status, complete every unmet requirement through N, run each intervening gate, and stop. It does not authorize beginning milestone $N + 1$.

Milestone 0 records the bundle compiler, visualization, narrow legacy adapter, MNIST training smoke test, and lifecycle equivalence work already demonstrated by *exp074–exp076*. Milestones 1–3 create the execution boundary, prove one graph natively, and then unlock arbitrary coupled circuits. Later milestones add a real gamma-coupling experiment, graph-native training, SHD and deeper networks, and online control.

Appendix B is the normative milestone ledger. The sequence is additive and reversible: no milestone deletes the working legacy path, changes an old runner, or changes a default in order to prove the next one.

8. Migration policy

Completed experiments are scientific records, not merely old application code. They should remain executable through their existing runners.

- New experiments use *snnlang* once the needed feature set exists.
- A small MNIST, SHD, and untrained simulation suite becomes the conformance set.
- An old experiment migrates when it is materially extended.
- A migrated version is treated as a new implementation and compared explicitly.
- Custom training or intervention code remains acceptable when it is genuinely experiment-specific.

Mass retrofitting would risk changing defaults, seed handling, initialization, or simulator configuration while producing reassuringly similar figures. Architectural purity is not worth damaged provenance.

9. Success criterion

The first new scientific capability milestone is milestone 3:

> *snnlang* compiles two independently driven PING circuits with reciprocal, delayed inhibitory coupling; the graph-native *tools/snn* executor simulates them without

changing the legacy executor; and named population recordings make their phase relationship measurable by an experiment runner.

The confidence-to-g_L experiment is the next system-level acceptance test. It succeeds whether the feedback is expressed through ordinary graph operations, a generic controller extension, or a stable runtime hook. The important constraint is that it composes with a compiled graph, remains reproducible, and does not require surgery on simulator internals.

Appendix A. Bundle and artifact lifecycle

A.1 Construction and compilation

An experiment-specific definition may begin inside its runner:

```
def make_bundle(tau_gaba_ms: float):
    net = make_ping_classifier(
        tau_gaba_ms=tau_gaba_ms,
    )
    train = make_standard_training(net)
    return snnlang.compile(
        net,
        training=train,
        target="tools/snn",
    )
```

The first use remains local. A repeated component may later move into a reusable circuit module. Reuse must be observed rather than predicted.

Compilation writes:

```
ping-tg9.bundle/
    manifest.json
    graph.json
    training.json
    reports/
        circuit.svg
        circuit.png
```

The executable files contain data only. They contain no Python callables, live instances, pickled authoring objects, accelerator tensors, or *torch.compile* products. The optional *reports/* directory contains derived human-readable views and may be regenerated exactly from the bundle.

A.2 Bundle contract

graph.json contains:

- populations, operations, ports, and projections;
- dynamics and structural parameters;
- initializer specifications;
- constants and optimizable parameters;
- constraints, masks, and ties;

- named outputs and exposed observables;
- grouping metadata for source-level components.

training.json contains:

- the exact graph digest;
- the expected data interface and encoder supported by the standard trainer;
- objectives, targets, and regularizers;
- optimizer parameter groups;
- surrogate-gradient and clipping settings;
- epochs and an explicit standard checkpoint-selection policy.

A resolved *data.json* contains:

- the input representation, such as dense samples or timestamped events;
- train, validation, or evaluation source paths;
- the mapping from source fields to graph inputs and training targets;
- shape, class-vocabulary, split, and content digests needed for validation and provenance.

It does not decide how a scientific split was created. That remains runner code.

manifest.json binds the files:

```
{
  "schema": "snnlang.bundle/v1",
  "graph": {
    "path": "graph.json",
    "sha256": "..."
  },
  "training": {
    "path": "training.json",
    "sha256": "...",
    "graph_sha256": "..."
  }
}
```

A simulation-only bundle may omit *training.json*. *tools/snn* rejects missing files, unsupported schema versions, digest mismatches, or backend capabilities absent from the graph.

A.3 Run state

config.json, written by *tools/snn*, records the resolved execution:

- execution mode, seed, device, timestep, and duration;
- graph and training schema versions and digests;
- data-binding, input, and checkpoint paths and digests;
- requested outputs;
- code and environment provenance.

A checkpoint contains evolving parameter values and optional optimizer state. Initializing, resuming, fine-tuning, and inference are invocation choices rather than changes to *graph.json*.

Graph-runtime state is a separate artifact again. It contains membrane voltages, refractory counters, per-projection conductances, recurrent delay histories, delayed-input context,

completed steps, and a structural compatibility signature—but no weights. The initial portable representation is an authenticated `manifest.json` plus `tensors.npz`. This separation permits a mature zero-weight graph to branch into a state-compatible non-zero-weight graph without confusing dynamic continuation with parameter checkpointing. *snnlang* continues to author topology only; *tools/snn* owns state validation and I/O, while the experiment runner owns checkpoint selection and branching.

A.4 Scratch layout

Each experiment uses regenerable scratch separately from its committed publication record:

```
temp/experiments/expNNN/
    bundles/
        ping-tg4p5.bundle/
        ping-tg9.bundle/
        ping-tg18.bundle/
    reports/
        ping-tg4p5.svg
        ping-tg9.svg
        ping-tg18.svg
    cells/
        ping-tg4p5-seed42/
        ping-tg4p5-seed43/
        ping-tg9-seed42/

artifacts/data/expNNN/
```

A runner compiles once per unique graph and training policy, not once per seed. It may reuse that bundle with different validated data bindings.

A.5 Execution

The runner uses a typed subprocess helper:

```
run_training(
    bundle=bundles[condition],
    data=data_binding,
    seed=seed,
    out_dir=cell_dir(condition, seed),
)
```

At the shell boundary:

```
tools/snn train \
  --bundle ping-tg9.bundle \
  --data data.json \
  --seed 42 \
  --out-dir cells/ping-tg9-seed42
```

The tool copies the exact files it executed into the run directory:

```
cells/ping-tg9-seed42/
    graph.json
    training.json
    data.json
    config.json
    metrics.json
    weights.pth
    output.log
    run.jsonl
    run.sh
```

Copying prevents later edits to a source bundle from changing the apparent meaning of existing weights and metrics.

A.6 Publication

The committed experiment record deduplicates descriptions while retaining the mapping from every reported cell:

```
artifacts/data/expNNN/
    numbers.json
    bundle_manifest.json
    graphs/
        ping-tg4p5.json
        ping-tg9.json
        ping-tg18.json
    diagrams/
        ping-tg4p5.svg
        ping-tg9.svg
        ping-tg18.svg
    training/
        train-tg4p5.json
        train-tg9.json
        train-tg18.json
    figure.svg
    raster.png
```

The manifest maps each cell to graph and training digests:

```
{
  "cells": [
    {
      "condition": "tg9",
      "seed": 42,
      "graph": "ping-tg9",
      "training": "train-tg9"
    }
  ]
}
```

Compiler caches and accelerator-specific products are disposable implementation state and are not published scientific artifacts.

A.7 Atomic runner lifecycle

```
def main():
    run_id = next_run_id(SLUG)

    with published_run(
        SLUG,
        run_id,
        scale=SCALE,
    ) as (scratch, staging):

        bundles = {}

        for condition in CONDITIONS:
            bundle = make_bundle(condition)
            path = (
                scratch
                / "bundles"
                / f"{condition}.bundle"
            )
            bundle.write(path)
            bundles[condition] = path

        for condition in CONDITIONS:
            for seed in SEEDS:
                run_training(
                    bundle=bundles[condition],
                    data=data_binding(condition),
                    seed=seed,
                    out_dir=cell_dir(
                        condition,
                        seed,
                    ),
                )

        rows = analyse_cells(
            CONDITIONS,
            SEEDS,
        )
        render_figures(rows, staging)
        publish_descriptions(
            bundles,
            staging,
        )
        write_bundle_manifest(
            bundles,
            CONDITIONS,
```

```
            SEEDS,
            staging,
        )
        write_numbers(
            staging,
            run_id=run_id,
            duration_s=duration,
            payload=summarise(rows),
        )
```

The final artifact directory is replaced only after compilation, execution, analysis, plotting, and summary generation all succeed. A failed run cannot publish new graph descriptions beside figures from an older run.

Appendix B. Implementation milestone ledger

B.1 How milestone goals are interpreted

A goal phrased as `Implement ar063 up to milestone N` is cumulative. The implementation agent must:

1. read this ledger and inspect the current code rather than assuming its status;
2. complete unmet deliverables from milestone 0 through N;
3. run every acceptance gate through N;
4. update the status ledger and supporting documentation with evidence;
5. preserve all compatibility invariants; and
6. stop before work unique to milestone $N + 1$.

A milestone is complete only when its tests and experiment evidence are committed. Merely creating types, schemas, or command-line flags does not count. If a gate reveals a numerical or compatibility regression, the milestone remains incomplete.

The status labels are `demonstrated`, `partial`, and `not started`. They describe evidence in the repository at the date of this proposal and must be revised as the implementation advances.

B.2 Compatibility invariants

Every stage keeps the existing CLI and historical runners operational. New functionality is added beside the legacy path and becomes the default only after correctness, checkpoint, artifact, and performance gates pass.

The transition begins with two frontends and, temporarily, two construction paths:

```
old runner
    -> legacy CLI flags
    -> compatibility adapter --\
                                +-> ExecutionSpec
new runner                     /
    -> snnlang bundle --------/

ExecutionSpec
    -> legacy COBANet executor
    or
    -> graph-native executor
```

The target is:

```
legacy CLI flags
    -> compatibility adapter --\
                                +-> ExecutionSpec
snnlang bundle                 /
    -> bundle loader ---------/

ExecutionSpec
    -> graph-native PyTorch executor
```

The compatibility adapter survives after convergence. The legacy executor remains selected for legacy CLI invocations until a later milestone explicitly changes that routing. In particular:

- old commands, defaults, configuration files, checkpoints, parameter names, recordings, and artifact schemas remain valid;
- no existing experiment runner is edited merely to adopt *snnlang*;
- `--executor graph` or an equivalent explicit bundle field selects new execution during the compatibility period;
- graph checkpoints use their own manifest until an explicit, tested parameter map permits cross-loading; and
- low-level numerical kernels may be shared, but the legacy `COBANet` construction and update path are not refactored as a prerequisite for graph execution.

B.3 Milestone 0: established bundle baseline

Status at 2026-08-05: demonstrated for the narrow MNIST COBANet slice; broader legacy characterization remains partial.

snnlang deliverables: Python authoring objects; graph and training IRs; PING and readout helpers; deterministic bundles and manifests; basic validation; circuit, expanded, and training diagrams.

tools/snn deliverables: load a supported bundle without importing *snnlang*; translate the exact one-layer PING plus mean-voltage graph into the legacy executor; translate its narrow MNIST recipe into the legacy trainer.

Evidence: exp074 simulates supplied spikes and records rasters; *exp075* trains a small MNIST subset; *exp076* compares seeded initialization, forward values, loss, gradients, one optimizer step, checkpoint interchange, and replay.

Gate: those experiments and their focused unit tests pass through the unchanged CLI. This baseline must remain green at every later milestone.

B.4 Milestone 1: freeze the compatibility seam

Status at 2026-08-05: demonstrated. The typed request seam, legacy-default selector, versioned element-level capability vocabulary, and data-only bundle boundary are implemented in commits `5be1cdb` and `cf11906`. Focused tests lower legacy MNIST, SHD, checkpoint, recording, and bundle invocations into the same request type while retaining legacy routing. *exp074–exp076* reran successfully through their historical interfaces; the validation evidence and anomalies are recorded in *exp077*.

Purpose: create a safe place for a second executor without changing numerical behaviour. This milestone adds architecture and tests, not new circuit science.

snnlang deliverables: define a versioned backend capability vocabulary and make compilation report required capabilities such as neuron kinds, synapse kinds, delays, feedback, operations, training, and recording modes. Capability failure must identify the graph element and missing feature rather than merely saying that a bundle is unsupported.

tools/snn deliverables: introduce typed `ExecutionSpec` and `ExecutionResult` objects and a small internal request API for build, simulate, train, and infer. Add an explicit executor selector with `legacy` as the default. The existing CLI becomes a thin caller of this API, while bundle loading remains data-only and does not import *snnlang*. A `graph` executor entry may initially fail with a precise "not implemented" diagnostic.

Compatibility gate: run the milestone-0 suite plus representative existing MNIST, SHD, untrained, checkpoint-load, and recording CLI smoke tests. Commands, resolved defaults, parameter names and shapes, seeded outputs, artifacts, and exit behaviour must remain unchanged. No historical runner imports the new API.

Exit criterion: both legacy flags and bundles produce an `ExecutionSpec`; all existing invocations still select the untouched legacy executor; capability reports agree with what the legacy bundle adapter actually accepts.

B.5 Milestone 2: graph-native single-PING forward execution

Status at 2026-08-05: demonstrated. Commit `cf11906` makes the existing PING state, refractory counts, membrane constants, update order, silent recurrence, readout reset, delays, initialisers, constraints, outputs, and observables explicit. *exp077* records zero parameter error, zero E/I spike mismatches, zero named-output error, and zero checkpoint-replay error on an active matched CPU fixture. The graph path is faster than the legacy path on the matched steady-state workload, so the overhead gate passes without an exception. CPU Inductor compilation time, warm runtime, replay error, and traced peak memory are reported separately. A larger CPU compile attempt was killed after five minutes; accelerator compilation remains unmeasured.

Purpose: prove the new lowering and scheduling machinery on a topology whose legacy result is known.

snnlang deliverables: emit all explicit state, update-order, delay, initializer, constraint, output, and observable information required to execute the existing one-layer PING graph without backend guesses.

tools/snn deliverables: add a graph planner and vectorized PyTorch executor for COBA-LIF E/I populations, AMPA/GABA projections, dense weights, direct spike inputs, non-spiking leaky-integrator populations, mean-voltage reduction, and named recordings. Lower the complete graph before the timestep loop; do not interpret graph nodes dynamically at every step. Keep `torch.compile` behind the same internal boundary used by the legacy engine.

Compatibility gate: legacy remains the default. Run the same bundle once through its legacy translation and once through the graph executor. Compare parameter identities and shapes, seeded initialization, short CPU state trajectories, spikes, named outputs, recordings, and checkpoint round trips. Benchmark compiled steady-state runtime, compilation time, and peak memory separately.

Exit criterion: the graph executor matches the legacy single-PING forward path within declared tolerances and is no more than approximately 5–10% slower at steady state for the reference workload, unless a measured exception is recorded and accepted before proceeding.

B.6 Milestone 3: arbitrary coupled forward graphs

Status at 2026-08-05: demonstrated. Commit `cf11906` adds deterministic topological scheduling, arbitrary named population sizes, independent inputs, dense feedforward/recurrent/feedback projections, multiple incoming conductance streams, integral delay buffers, and all-population recordings. *exp077* (fixture commit `3b7a935`, evidence commit `fdfb19f`) archives uncoupled, unidirectional, reciprocal zero-additional-delay, and reciprocal explicitly delayed two-PING graphs. Each variant differs only in graph data and retains its authenticated graph, manifest, canonical diagram, named input/E/I recordings, delay evidence, and execution provenance. The fixture computes only compact recording diagnostics; the scientific coupling sweep is performed separately by Milestone 4 in *exp078*.

Purpose: deliver the first capability that the legacy `COBANet` architecture cannot express.

snnlang deliverables: compile multiple independently named components, feedforward/recurrent/feedback projections, explicit positive delays, arbitrary E/I sizes, independent inputs, and observables from every population. Tighten validation for temporal causality, projection dimensions, polarity, and delayed feedback.

tools/snn deliverables: execute arbitrary named populations and projections, multiple incoming conductance streams, delay buffers, deterministic recurrent and feedback scheduling, and population-level recordings. Initial support may remain dense; sparse and structured lowering are later optimizations.

Acceptance fixture: two independently driven PING circuits with reciprocal GABA projections from each inhibitory population to the other excitatory population. Include uncoupled, unidirectional, reciprocal, and delayed variants. Exact tests establish which

timestep receives each pulse; the experiment runner, not the engine, computes phase locking and synchrony.

Compatibility gate: milestone-0 and milestone-2 parity suites remain green and legacy CLI routing remains unchanged.

Exit criterion: all coupling variants require only graph changes, not simulator edits, and archive their graph, manifest, diagram, recordings, and execution provenance.

B.7 Milestone 4: first native gamma-coupling experiment

Status: executor capability complete; scientific experiment planned. Exact split-run tests cover refractory state, live conductances, recurrent delays, and delayed inputs. Runtime-state compatibility excludes parameter values but rejects changes to timestep, population size, projection identity, delay, synapse, and state layout. This is sufficient to burn in a structurally complete graph with reciprocal E-to-I weights at zero and continue the same causal trajectory with nonzero weights. General time-dependent weights are not required.

exp078 executes the registered graph-native Arnold-tongue test in two 80 E / 20 I circuits. It calibrates uncoupled gamma frequency, freezes locking tolerances and a bounded coupling pilot, then crosses measured natural-frequency detuning with reciprocal coupling. The 80/20 result recovers the widening tongue but narrowly fails the strict phase-lead clause. A reduced 800 E / 200 I confirmation near that phase-sign boundary is the appropriate finite-size follow-up; the full sweep is not silently redefined at tenfold scale.

B.8 Milestone 5: graph-native readouts and input bindings

snnlang deliverables: finish precise shape and unit inference for mean voltage, final voltage, spike count, duration-normalized spike rate, and cumulative potential. Define resolved dense and event-stream input bindings separately from the graph, including masks and durations; paths live in execution data, not in the reusable graph.

tools/snn deliverables: execute all five readouts, supplied dense spike arrays, event streams, masks, named outputs, and recordings through stable request and CLI contracts. Dataset generation remains optional: callers may provide an input artifact or request a reusable tool-side encoder.

Gate: hand-calculated micro-fixtures verify every readout and masked-duration case; existing input and artifact behaviour remains unchanged.

B.9 Milestone 6: graph-native MNIST training

snnlang deliverables: validate cross-entropy objectives, complete parameter-group partitions, frozen parameters, AdamW settings, gradient clipping, checkpoint selection, and optimizer-state replay. Reject objectives without a differentiable path to a trainable parameter.

tools/snn deliverables: train the milestone-2 graph natively with surrogate gradients and the standard MNIST input binding. Support save, resume, selected and final checkpoints, inference, and optimizer-state replay.

Gate: compare the legacy and graph paths on initialization, forward values, gradients, one update, short learning curves, checkpoint replay, artifacts, compiled runtime, and memory. Keep bundle execution opt-in.

B.10 Milestone 7: SHD and deeper trained graphs

Extend data binding and graph-native training to event-stream SHD and multiple hidden PING layers. Add named recordings from every layer and an explicit standard checkpoint-selection contract. Use the existing one-layer SHD and two-layer SHD cells as conformance cases rather than retrofitting their runners.

Gate: dense MNIST, one-layer SHD, and two-layer SHD train and evaluate without runner access to live model internals, while the original commands and checkpoints remain supported.

B.11 Milestone 8: regularization and training boundaries

Implement declared surrogate choices, rate regularizers, multiple optimizer groups, constraints, stop-gradient boundaries, and differentiable-reachability diagnostics. Add only features exercised by current or imminent experiments.

Gate: each training construct has a hand-checkable gradient or update fixture; unsupported constructs fail during compilation or planning, never halfway through an expensive run.

B.12 Milestone 9: online confidence-controlled leak

Make an MNIST classifier's online confidence modulate g_L, allowing low- confidence trials more PING cycles. Attempt the least specialized route in order:

1. compose existing graph operations;
2. add a generic stateful controller node;
3. add a registered PyTorch backend operation;
4. use the importable API as an explicitly experimental extension.

Do not create a dedicated `ConfidenceToLeak` primitive solely for one experiment. The chosen route declares causal timing, bounds, initial state, evidence source, target parameter, checkpoint behaviour, and differentiability.

Gate: fixed-confidence fixtures reproduce expected g_L or τ_m trajectories; a one-step feedback delay has an exact causal test; the experiment compares a matched fixed-leak control and reports accuracy, calibration, decision time, PING cycles, and spike-count or energetic cost.

B.13 Historical experiment policy

Completed experiments remain unchanged by default. They are scientific records, not migration chores.

- Run selected old cells as regression cases through their original interface.
- Create separate snnlang conformance definitions rather than editing historical runners.
- Do not mass-retrofit the experiment corpus.
- Migrate an old experiment only when it is materially extended.
- Treat a migrated implementation as new and compare it explicitly with the archived result.
- Preserve the legacy CLI even after the graph executor becomes the default.

This policy avoids changing defaults, random initialization, dataset splits, checkpoint selection, or artifact meaning merely to achieve architectural uniformity. There is no scientific prize for deleting working compatibility code early.

B.14 Milestone 10: optional convergence and retirement

Changing the default executor or retiring `COBANet` is a separate, optional milestone, not an automatic consequence of graph-native success. It requires every selected conformance case to pass through the compatibility adapter, historical checkpoints to load or have a documented conversion, artifact contracts to remain stable, compiled performance not to be materially worse, and an explicit decision to accept the migration. The legacy CLI compatibility adapter survives retirement of the legacy execution architecture.

Compute options

11 August 2026

This is the operational map for pinglab compute. It separates the persistent control plane from the machines that perform numerical work, and records the authentication boundary for each provider. Prices, balances, queues, and GPU stock change. The commands below query those values when a decision is made instead of preserving a stale snapshot here.

Never place a password, TOTP seed, private SSH key, RunPod API key, or Modal token in this repository. Authentication commands either use an encrypted local key or open the provider's own interactive login flow.

1. Local Mac

When to use. Use the Mac for editing, Demolab preview, plotting, analysis of collected results, dry-run dispatch plans, and plumbing-scale tests. It is the shortest feedback loop. It is not a CUDA training machine.

Provider overview. This is the local development workstation. It has no scheduler and no marginal compute charge. The working checkout is `/Users/eoin/pinglab`.

How to use. Local access requires the macOS user session rather than a separate infrastructure login.

```
cd /Users/eoin/pinglab
uv sync --dev

# Run an experiment locally.
uv run python experiments/expNNN.py

# Preview the Demolab collection.
demolab dev
```

Keep cloud commands in dry-run mode until spending has been explicitly authorised.

2. Hetzner control plane

When to use. Use Hetzner for the persistent Codex session, orchestration, monitoring, result collection, and the public development preview. Do not use its small CPU and memory allocation for substantial numerical experiments.

Provider overview. Hetzner supplies the always-on Linux host named `pinglab-codex`. Caddy terminates HTTPS for `pl-hetzner.eoinmurray.info` and proxies to the Demolab development server on port `3000`. The server is a control plane, not a GPU worker.

How to use. The Mac alias `hetzner` authenticates as `eoin` with the dedicated Ed25519 identity configured in `~/.ssh/config`. The private key remains on the Mac. The Hetzner host has its own dedicated key for Olorin.

```
ssh hetzner

# Inspect the persistent session and preview service.
tmux list-sessions
tmux attach -t shd-autoresearch
systemctl --user status demolab-dev.service

# Reach Olorin from the control plane.
ssh olorin 'hostname; whoami'
```

3. Olorin

When to use. Use Olorin for large single-GPU or multi-GPU work when a GPU is visibly idle and the Division F fair-use policy permits the allocation. Its large VRAM makes it the first choice for workloads that do not fit on consumer GPUs.

Provider overview. Olorin is a shared Division F machine with four NVIDIA RTX PRO 6000 Blackwell Server Edition GPUs, each with approximately 96 GB of VRAM. It currently has no Slurm scheduler. Availability is therefore manual and may change between inspection and launch.

How to use. The Mac alias `olorin` connects through `gate.eng.cam.ac.uk` using `~/.ssh/olorin_codex`. Hetzner has a separate dedicated key. Passwords are not stored. Check utilization immediately before every launch, select only an idle GPU, and keep heavy files under `/scratch/em586`.

```
ssh olorin
nvidia-smi

# Inspect utilization without opening an interactive shell.
ssh olorin \
  'nvidia-smi --query-gpu=index,utilization.gpu,memory.used,memory.total \
  --format=csv,noheader'

# On Olorin, after confirming that GPU N is idle:
cd /scratch/em586/pinglab
CUDA_VISIBLE_DEVICES=N uv run python experiments/expNNN.py
```

Never launch on all four GPUs merely because the machine accepts the command. Shared infrastructure without a scheduler requires more restraint, not less.

4. CSD3 Wilkes3, Service Level 2

When to use. Use SL2 for planned production runs where a queue of hours is acceptable. It is the default scheduled backend for long, reproducible A100 jobs and should consume the purchased allocation before commercial cloud credits.

Provider overview. Wilkes3 is Cambridge's Slurm-managed A100 cluster. The SL2 GPU account is `OLEARY-SL2-GPU`; the Ampere partition uses QOS `gpu1`. Account balance and scheduler estimates are live administrative state and must be queried at submission time.

How to use. Connect as Cambridge user `em586`. SSH requires the Raven/UIS password plus the six-digit token labelled `CSD3: SSH Login`. The `csd3` alias uses `ControlMaster` so subsequent commands reuse the authenticated connection. An SSH key can replace the password factor, but mandatory TOTP remains unless Research Computing Services arranges an automation-specific solution.

```
# Establish or reuse the multiplexed connection.
ssh csd3

# Inspect purchased hours and account associations.
ssh csd3 mybalance
ssh csd3 'sacctmgr show assoc user="$USER" format=Account,Partition,QOS'

# Submit one A100 job on SL2.
sbatch \
  --account=OLEARY-SL2-GPU \
  --partition=ampere \
  --qos=gpu1 \
  --gres=gpu:1 \
  job.slurm

# Ask Slurm for an estimate without submitting.
sbatch --test-only \
  --account=OLEARY-SL2-GPU --qos=gpu1 job.slurm
```

Command-line `sbatch` options override matching `#SBATCH` lines, so one job script can target either service level.

5. CSD3 Wilkes3, Service Level 3

When to use. Use SL3 for non-urgent work that can wait several days, for overflow, or for jobs submitted well before their results are needed. It is a poor interactive backend.

Provider overview. SL3 uses the same Wilkes3 A100 hardware and Ampere partition but a lower-priority service level. The GPU account is `OLEARY-SL3-GPU` and its QOS is `gpu2`.

How to use. Authentication is identical to SL2. Toggle the account and QOS at submission time:

```
sbatch \
  --account=OLEARY-SL3-GPU \
  --partition=ampere \
  --qos=gpu2 \
  --gres=gpu:1 \
  job.slurm

sbatch --test-only \
  --account=OLEARY-SL3-GPU --qos=gpu2 job.slurm

squeue -u "$USER"
```

The names are easy to invert: SL2 maps to `gpu1`, while SL3 maps to `gpu2`.

6. RunPod

When to use. Use RunPod for urgent burst capacity when Olorin is occupied and Wilkes3 is queued. Prefer a 4090 when the workload fits in 24 GB and stock exists; use a 5090 for additional VRAM or faster turnaround. Always run the smoke test before a fleet launch because allocation does not guarantee that the container will become usable promptly.

Provider overview. RunPod provides per-second GPU pods. Pinglab targets Secure Cloud in `EU-RO-1`, attaches the shared network volume, and uses `ghcr.io/eoinmurray/pinglab:cu128`, the same image used by experiment dispatch. GPU stock and regional prices are volatile. Pods must be reaped after failures because a rented but unusable pod can still bill.

How to use. `runpodctl doctor` stores the account API key in the user's RunPod configuration. Do not commit the key. The account also holds an SSH public key, optional S3 credentials for volume collection, and optional container-registry authentication for GHCR pulls.

```
# One-time interactive authentication.
runpodctl doctor

# Read-only inventory and account checks.
runpodctl gpu list
runpodctl datacenter list
runpodctl pod list -o json
runpodctl user -o json

# Free plan, then an explicitly paid capacity check.
uv run python experiments/helpers/runpod_smoke.py
uv run python experiments/helpers/runpod_smoke.py --live

# Experiment dispatch remains a dry-run without --live.
uv run python experiments/exp022.py --runpod --gpu 5090
uv run python experiments/exp022.py --runpod --gpu 5090 --live

# Kill switch for an exp022 fleet.
uv run python experiments/exp022.py --runpod --reap
```

Every pod-creating command spends money. Dry-run first, obtain explicit approval, launch, monitor, collect, and verify that `runpodctl pod list` is empty.

7. Modal

When to use. Use Modal when reliable serverless startup, automatic scaling, and reduced infrastructure management justify a higher GPU-hour price. It is useful as an escape hatch when RunPod image or SSH transport is unreliable. Only runners with an implemented Modal backend can use it.

Provider overview. Modal runs containerized functions and bills GPU, CPU, and memory by execution time. It provides managed scheduling rather than a persistent SSH host.

Pinglab's Modal integration is narrower than its RunPod integration, so backend support must be confirmed in the selected runner.

How to use. `modal setup` opens Modal's authentication flow and stores a local token. Never commit token values. As with RunPod, pinglab requires `--live` before a runner dispatches paid work.

```
# One-time interactive authentication.
uv run modal setup

# Confirm the runner's options before dispatch.
uv run python experiments/exp073.py --help

# Free plan, followed by an explicitly authorised live dispatch.
uv run python experiments/exp073.py --modal
uv run python experiments/exp073.py --modal --live
```

Modal is not a drop-in flag for every experiment. If a runner does not expose `--modal`, adding a backend is implementation work rather than a command-line choice.

8. C3 Cloud

When to use. Consider C3 Cloud when UK-hosted commercial GPU capacity or institutional procurement matters, or when RunPod and Modal are unsuitable. Treat it as a fallback until pinglab has a tested deployment path.

Provider overview. C3 Cloud offers on-demand NVIDIA GPU machines, including A100, H100, and L40-class hardware. Provisioning is VM-oriented and can include a cold-start delay. Pinglab currently has no C3 dispatcher, network-volume contract, automated teardown, or capacity smoke test.

How to use. Authenticate through the C3 dashboard, provision a machine with an SSH public key, record the assigned hostname, and connect using the corresponding private key. Do not upload or share the private key. The exact command is supplied by the provisioned instance:

```
ssh -i ~/.ssh/<c3-key> <user>@<assigned-host>

# On the instance, verify the accelerator before installing or launching
work.
nvidia-smi
```

Before a real experiment, C3 still needs a documented image or bootstrap procedure, artifact collection, a spending ceiling, and verified teardown. Until those exist, it is available infrastructure rather than an operational pinglab backend.

Decision order

Use the smallest adequate option. Develop locally; orchestrate from Hetzner; use an idle Olorin GPU for opportunistic free work; submit planned production to Wilkes3 SL2; use

RunPod for urgent overflow; choose Modal when managed reliability is worth the premium; leave SL3 for work that can wait; and treat C3 as an integration candidate.

Cloudflare R2 archive

11 August 2026

Cloudflare R2 stores selected experiment scratch that is expensive to reproduce, especially trained checkpoint banks under `temp/experiments/`. Git remains the record for source code and published artifacts. R2 is a manual backup, not a live training filesystem.

The supported interface is `experiments/helpers/archive.py`. It stores snapshots at:

```
r2:pinglab/archive/<experiment-slug>/<producing-git-sha>/
```

Access and authentication

The Mac currently has the `r2:` rclone remote. Hetzner does not. Credentials consist of an Access Key ID, Secret Access Key, and the EU R2 endpoint. Never commit or paste their values.

To configure a new machine, create an **Object Read & Write** token scoped to the `pinglab` bucket in the Cloudflare dashboard. Then configure rclone interactively:

```
rclone config
```

Use these choices:

1. Remote name: `r2`.
2. Storage: `s3`.
3. Provider: `Cloudflare`.
4. Enter the Access Key ID and Secret Access Key manually.
5. Endpoint: the EU endpoint shown by Cloudflare.
6. ACL: `private`.

Prefer a separate bucket-scoped token for each machine. See Cloudflare's token guide and rclone guide.

Verify access without writing:

```
rclone listremotes
rclone config redacted r2
rclone lsd r2:
rclone lsf --dirs-only r2:pinglab/archive
```

Do not print the unredacted rclone configuration.

Archive a run

Check the local scratch, then archive it:

```
du -sh temp/experiments/exp022
uv run python experiments/helpers/archive.py archive exp022
```

The helper determines the producing Git commit, copies the tree, uploads `MANIFEST.json`, and verifies it with `rclone check`. It uses `copy`, never `sync`, so a partial local tree cannot delete existing remote objects.

List snapshots

```
uv run python experiments/helpers/archive.py list exp022

rclone tree r2:pinglab/archive/exp022 --max-depth 2
rclone size r2:pinglab/archive/exp022
rclone cat r2:pinglab/archive/exp022/<sha>/MANIFEST.json
```

Restore a snapshot

Restore the latest snapshot:

```
uv run python experiments/helpers/archive.py restore exp022
```

Or restore a specific producing commit:

```
uv run python experiments/helpers/archive.py restore exp022 <sha>
```

Restore copies into `temp/experiments/exp022/` without deleting unrelated local files. Validate configuration files and load representative checkpoints before using the restored bank.

Safety rules

- Never run `rclone sync` against an archive prefix.
- Archive only completed runs worth preserving.
- Do not use R2 as mutable training storage.
- Keep snapshots from different producing commits separate.
- Revoke and replace a token immediately if it may have leaked.
- Do not confuse R2 credentials with `RUNPOD_S3_ACCESS_KEY_ID` and `RUNPOD_S3_SECRET_ACCESS_KEY`; those access RunPod storage.

The helper deliberately has no delete command. Remote deletion should remain a separate, explicit administrative action.

Training-run guide

11 August 2026

Contents

Abstract

Exp022 defines the collection's shared training runs and checkpoint bank. It specifies the motivation, parameterization, output-layer shape, and downstream consumers for six training-run types. Five run types comprise 87 trained cells spanning reference models and controlled sweeps over spike budget, inhibitory timescale, integration timestep, and recurrent initialization. TR-06 adds three PING cells trained across variable input rates with ten spiking output LIF neurons; each class logit is the corresponding neuron's output firing rate. Together, these runs provide the checkpoints used by the collection's training-dependent experiments.

Summary

All runs map Poisson-encoded pixels through 1,024 excitatory neurons to ten spiking output LIF neurons. COBA and PING use the same input-weight mean and readout initialization scale. COBA disables recurrent E/I coupling; PING adds a $1024 \to 256 \to 1024$ E/I feedback loop and uses stronger backward-pass gradient damping to stabilize training through that recurrent path.

Unless a run-specific table says otherwise, every cell uses the following contract.

Parameter	Default	Meaning
Dataset	MNIST	784 normalized pixels encoded as independent Poisson channels
Presentation duration	200 ms	One static digit per training presentation
Integration timestep	0.1 ms	2,000 recurrent updates per presentation
Epochs	50	Training horizon for every production cell
Seeds	42, 43, 44	Three independently initialized cells per configuration
Minibatch	256	Presentations per optimization step
Optimizer learning rate	4×10^{-4}	Shared learning rate
Input population	784	One channel per image pixel
Excitatory population	1,024	Learned stimulus representation
Inhibitory population	256	PING feedback population; silent when E/I coupling is disabled
τ_{AMPA}	2 ms	Fixed excitatory synaptic decay
τ_{GABA}	6 ms	Default inhibitory decay at the gamma operating point
Input sparsity	0.95	Sparsity of the input projection
Input-weight mean	0.9	Shared by COBA and PING so input initialization is not an architecture-specific factor
Readout initialization scale	225	Shared by COBA and PING so the initial classifier scale is matched
E/I loop strength	COBA: 0; PING: 1	The forward architectural difference under comparison
Gradient damping	COBA: 1; PING: 1,000	Backward-pass stabilization for the recurrent PING loop; it does not change forward dynamics
Surrogate slope	1	Spike-gradient surrogate parameter
Default readout	`mem-mean`	A **spiking** $1024 \to 10$ output-LIF layer. Its neurons emit spikes and reset, but classification logits are their membrane voltages averaged over time—not their spike counts
Stored projection shapes	784×1024; 1024×256; 256×1024; 1024×10	Input→E, E→I, I→E, and E→class, in source-to-destination orientation

Training-run guide

TR-01 — Canonical full-data reference

The full-data COBA and PING cells are used for headline accuracy. They train on all pooled MNIST with no spike-budget penalty, so the comparison is not affected by the smaller dataset or regularization used in the sweeps. This experiment uses these cells directly.

Key parameter	Value	Why it differs
Architectures	COBA and PING	Provides a feedforward control and recurrent E/I model
Training pool	70,000 samples	Uses all pooled MNIST rather than the sweep default of 7,000
Input rate	25 Hz maximum-pixel rate	Fixed-rate collection baseline
Spike budget	Off	Measures unconstrained capacity
Cells	2 architectures × 3 seeds = 6	Across-seed comparison for both models
Readout shape	1024 → 10 spiking LIF outputs	`mem-mean`: mean membrane voltage supplies the logits

---

TR-02 — Spike-budget sweep

This run measures the trade-off between accuracy and firing rate as the spike budget is tightened. It uses a smaller training pool to keep the multi-seed sweep manageable; only the spike cap and its penalty change across conditions. Its absolute rates should therefore not be compared directly with the full-data cells. The resulting checkpoints are used by exp024, exp025, exp037, exp038.

Key parameter	Value	Why it differs
Architectures	COBA and PING	Direct architecture comparison
Training pool	7,000 samples	Keeps the 36-cell sweep tractable
Spike budget θ_u	off, 5, 2, 1, 0.5, 0.2 spikes/ neuron/trial	Spans unconstrained through severe sparsity
Penalty strength	10^{-3} when enabled	Applies the upper-rate regularizer
Cells	2 × 6 settings × 3 seeds = 36	Error bars at every frontier point
Readout shape	1024 → 10 spiking LIF outputs	`mem-mean`: mean membrane voltage supplies the logits

---

TR-03 — Inhibitory-timescale sweep

This run changes τ_{GABA} to test how the inhibitory timescale affects gamma frequency, firing rate, and accuracy. All other scientific settings are held fixed, with 6 ms as the standard condition. The resulting checkpoints are used by exp041, exp042, exp046.

Key parameter	Value	Why it differs
Architecture	PING	The manipulated quantity belongs to the recurrent inhibitory loop
Training pool	7,000 samples	Sweep-scale default
τ_{GABA}	4.5, 6, 9, 12, 18, 27 ms	Moves the inhibitory rhythm across a broad timescale range
Cells	6 settings × 3 seeds = 18	Across-seed estimate at each decay
Readout shape	1024 → 10 spiking LIF outputs	`mem-mean`: mean membrane voltage supplies the logits

---

TR-04 — Integration-timestep sweep

This run changes the integration timestep while keeping each presentation at 200 ms. It tests whether the observed dynamics depend on numerical resolution and measures the extra compute required by finer timesteps. The 0.05 ms cells need the large-memory Cambridge request. The resulting checkpoints are used by exp044.

Key parameter	Value	Why it differs
Architecture	PING	Tests the recurrent reference model
Training pool	7,000 samples	Sweep-scale default
Δt	0.05, 0.1, 0.25, 0.5, 1 ms	Changes numerical resolution while holding 200 ms physical time fixed
Steps/presentation	4,000; 2,000; 800; 400; 200	Compute and activation memory scale inversely with Δt
Cells	5 settings × 3 seeds = 15	Across-seed stability check
Readout shape	1024 → 10 spiking LIF outputs	`mem-mean`: mean membrane voltage supplies the logits

---

TR-05 — Recurrent-initialization sweep

This run tests whether training preserves the PING loop or learns a useful loop from weaker initial conditions. Recurrent initialization and trainability change together, while the feedforward network and classifier remain fixed to the PING recipe. The resulting checkpoints are used by exp049.

Key parameter	Value	Why it differs
Architecture	PING	Manipulates the recurrent loop directly
Training pool	7,000 samples	Sweep-scale default
Loop conditions	frozen PING; trainable PING init; trainable zero init; trainable 0.1 init	Separates built-in dynamics from recurrence learned during task training
Trainable projections	W_{EI} and W_{IE} only in trainable conditions	The frozen condition is the mechanistic control
Cells	4 conditions × 3 seeds = 12	Across-seed comparison
Readout shape	1024 → 10 spiking LIF outputs	`mem-mean`: mean membrane voltage supplies the logits

---

TR-06 — Variable-rate streaming bank

This run trains PING across the input rates used by exp082. One rate is sampled uniformly for each presentation, and the ten output LIF neurons produce class logits from their firing rates rather than their mean membrane voltages. Expressing the logits in hertz makes them comparable across presentation durations. During streaming inference, the output neurons reset at digit boundaries while the hidden PING state continues. The resulting checkpoints are used by exp082.

Key parameter	Value	Why it differs
Architecture	PING	Target model for streaming inference
Training pool	7,000 samples	Uses the shared sweep-scale training set
Input-rate set	0.5, 0.75, 1, 1.5, 2, 3, 5, 7.5, 10, 15, 25 Hz	Denser sampling within the interval selected by exp080
Sampling rule	Uniform categorical, independently per presentation	Makes rate variation part of the training distribution
Readout	`spike-rate`	Hidden E spikes drive ten spiking LIF class neurons; each logit is that class neuron's spike count divided by presentation duration in seconds
Readout shape	1024 → 10 spiking LIF outputs	Ten class neurons emit and reset throughout the presentation
Cells	1 recipe × 3 seeds = 3	Checkpoint bank expected by exp082

Results by training run

Each completed run reports one training-curve figure and one representative seed-42 raster. The figures summarize all configurations and seeds; the raster is a diagnostic example, not an across-seed statistic.

TR-01 results — Canonical full-data reference

Run status: complete · 6/6 cells represented

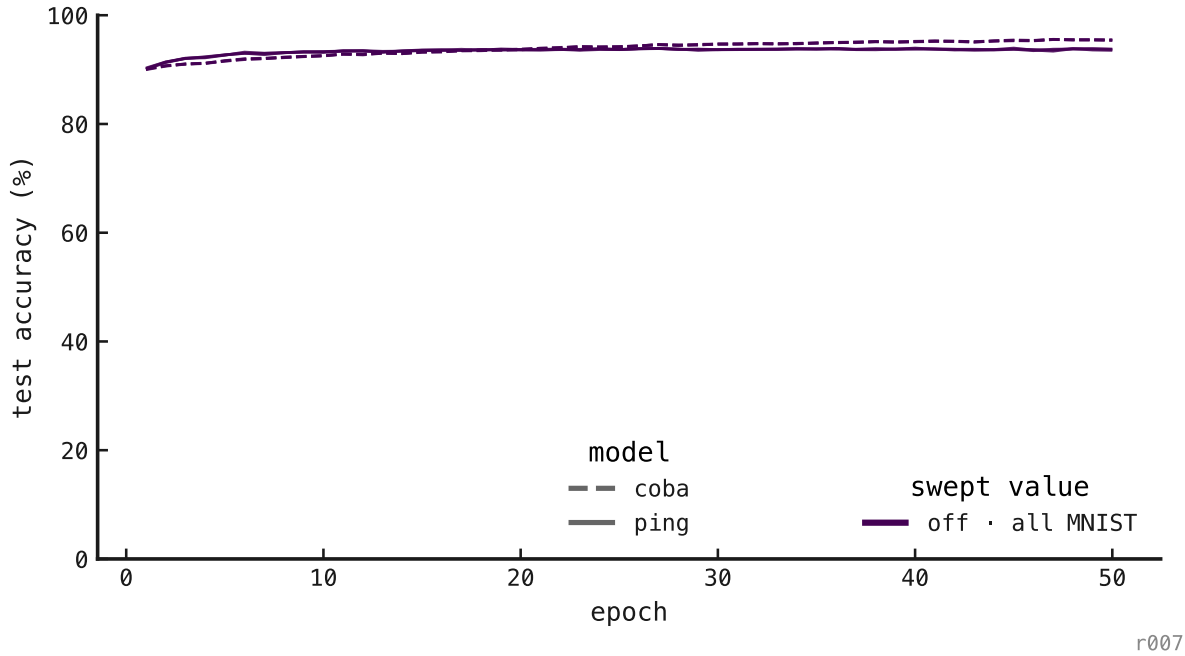


Figure 13: Training curves for the six full-data reference cells.

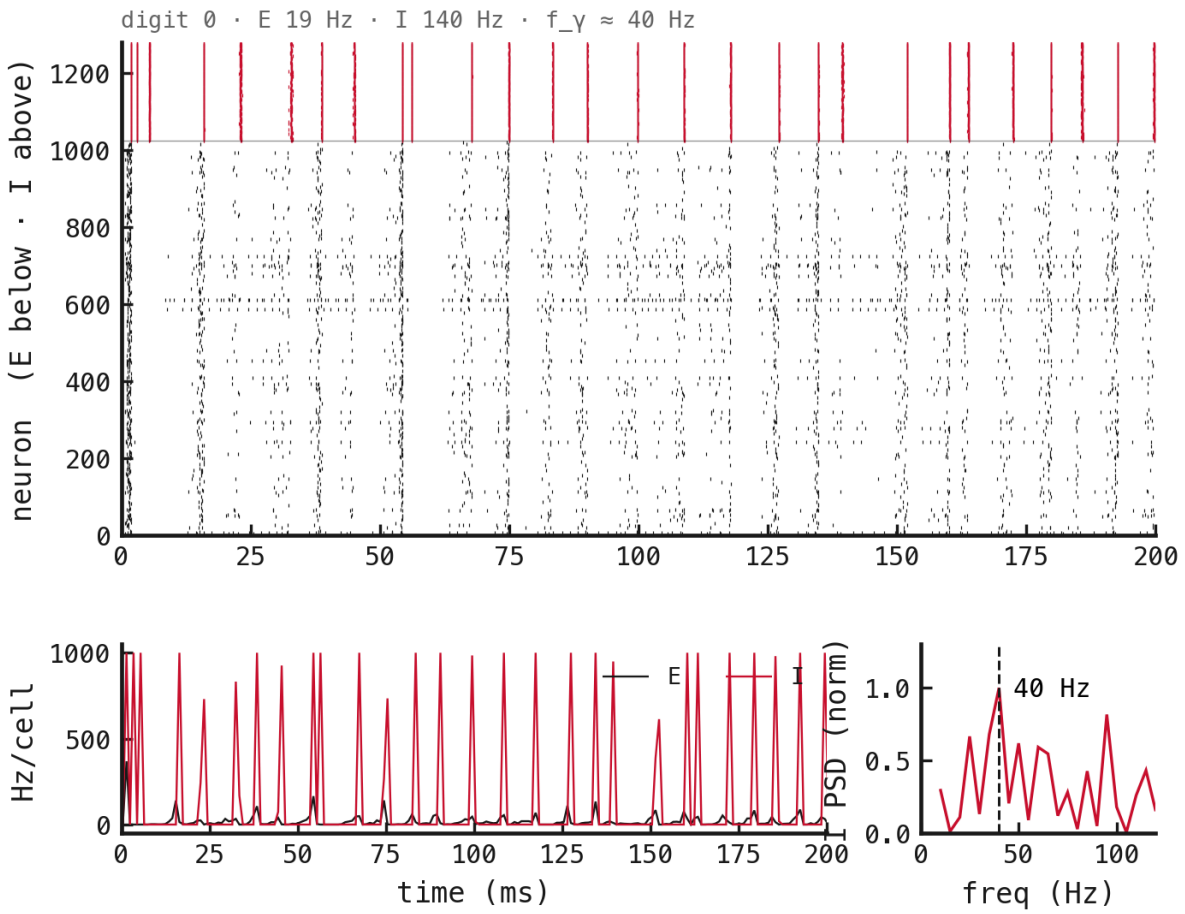


Figure 14: Sample raster: canonical PING, seed 42, held-out digit-0 probe.

TR-02 results — Spike-budget sweep

Run status: complete · 36/36 cells represented

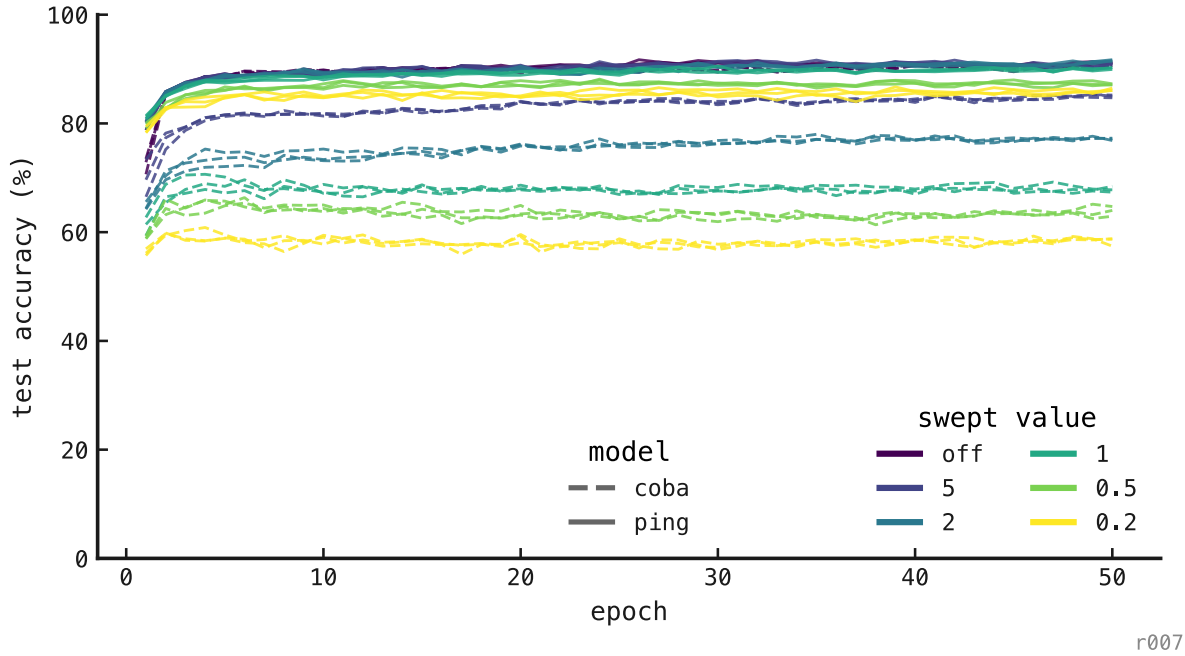


Figure 15: Training curves across both architectures, six spike budgets, and three seeds.

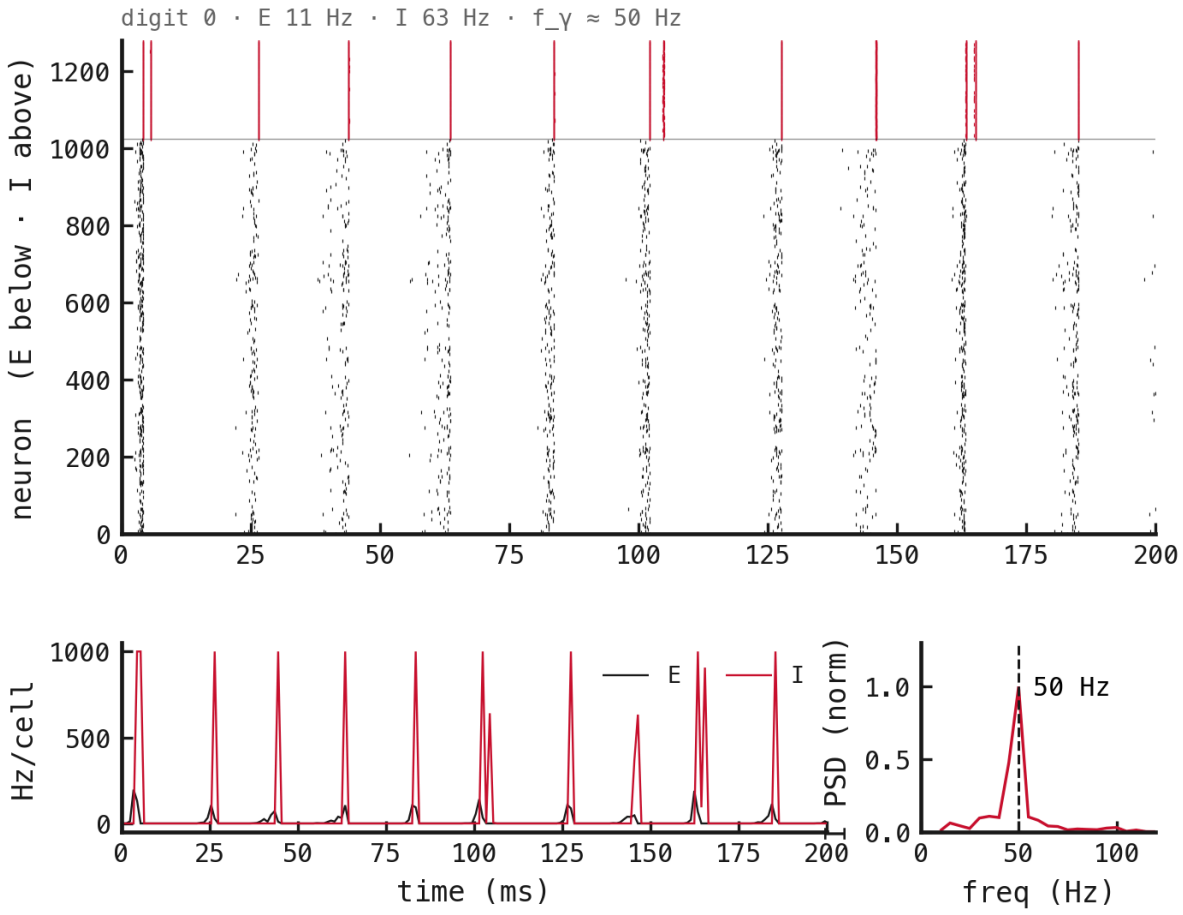


Figure 16: Sample raster: PING with spike budget off, seed 42.

TR-03 results — Inhibitory-timescale sweep

Run status: complete · 18/18 cells represented

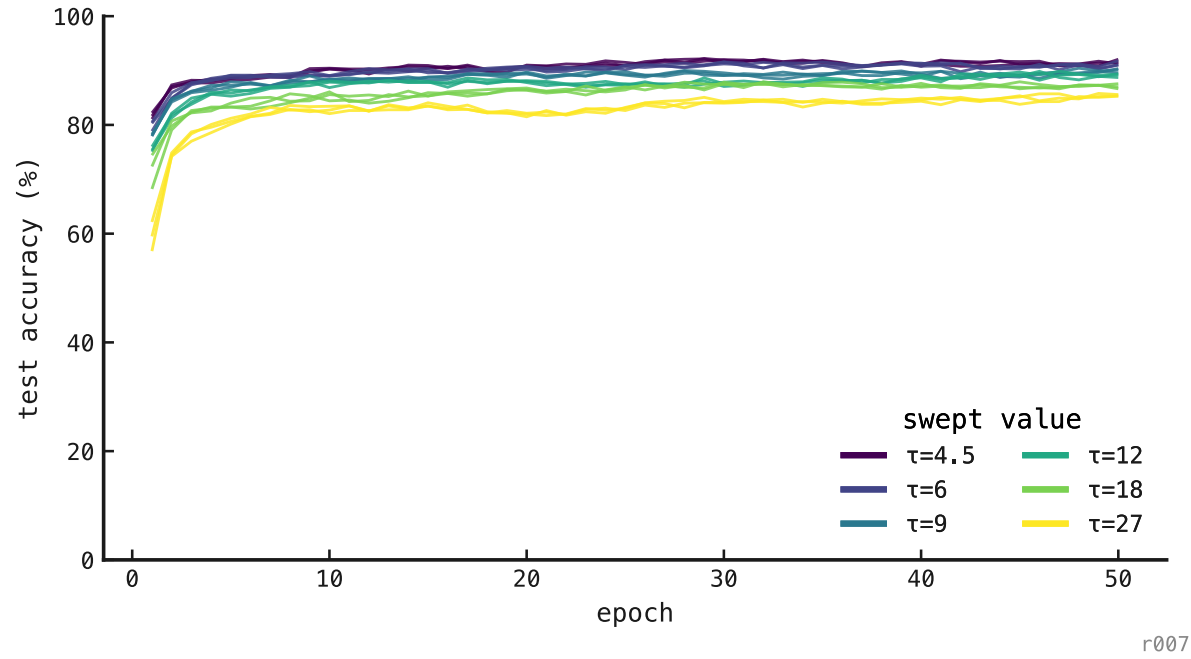


Figure 17: Training curves across six τ_{GABA} values and three seeds.

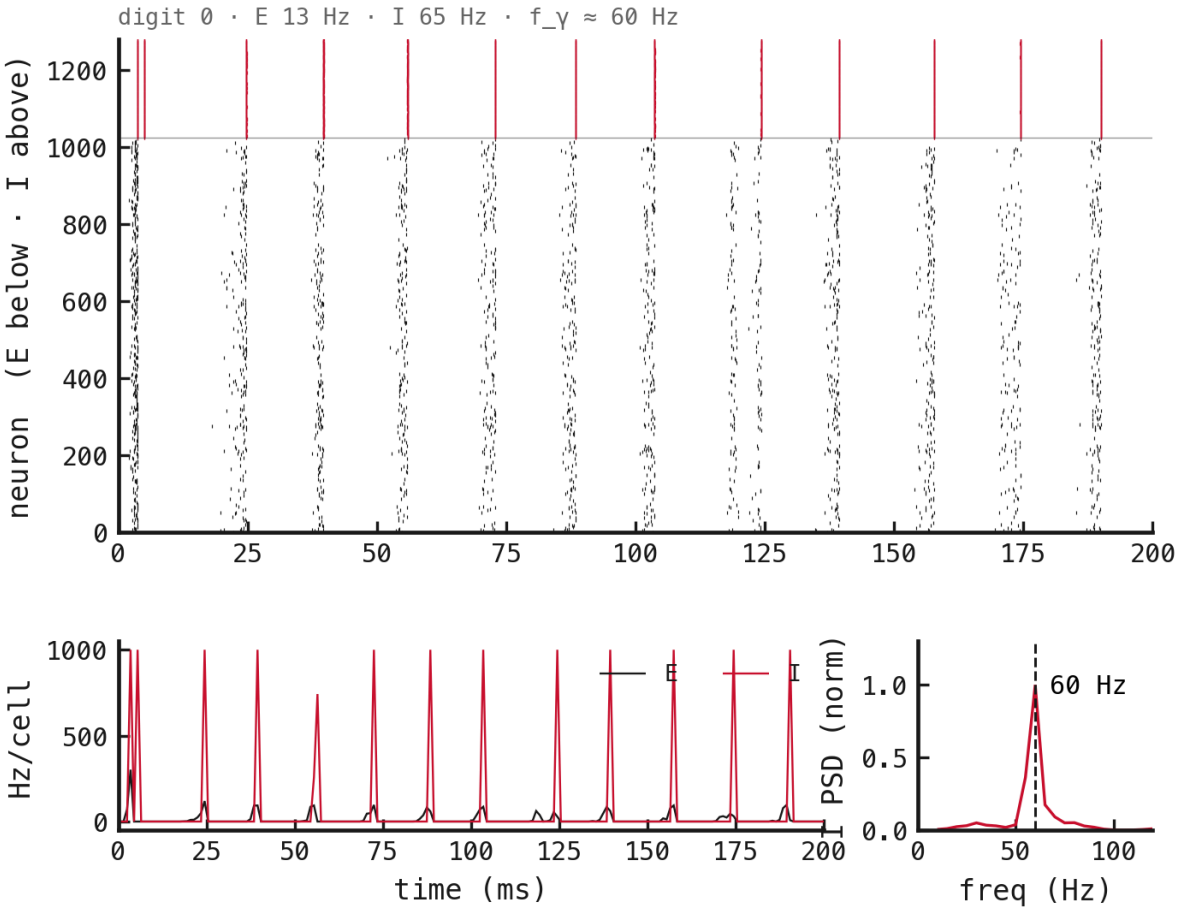


Figure 18: Sample raster: PING at $\tau_{\text{GABA}} = 6$ ms, seed 42.

TR-04 results — Integration-timestep sweep
Run status: complete · 15/15 cells represented

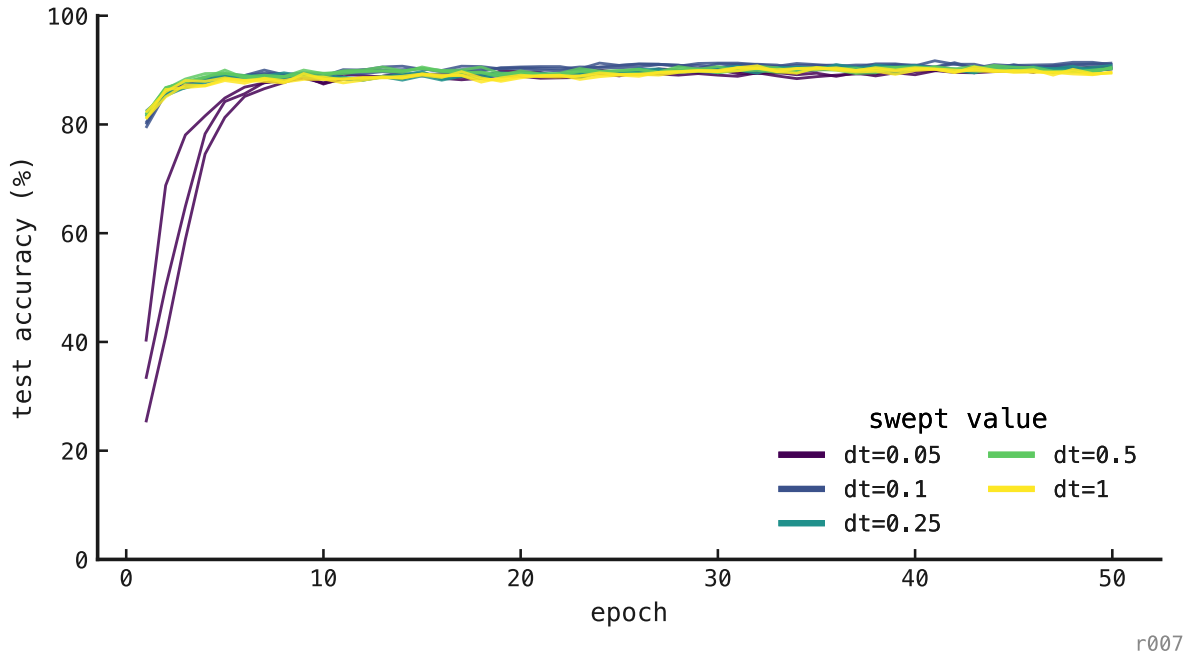


Figure 19: Training curves across five integration timesteps and three seeds.

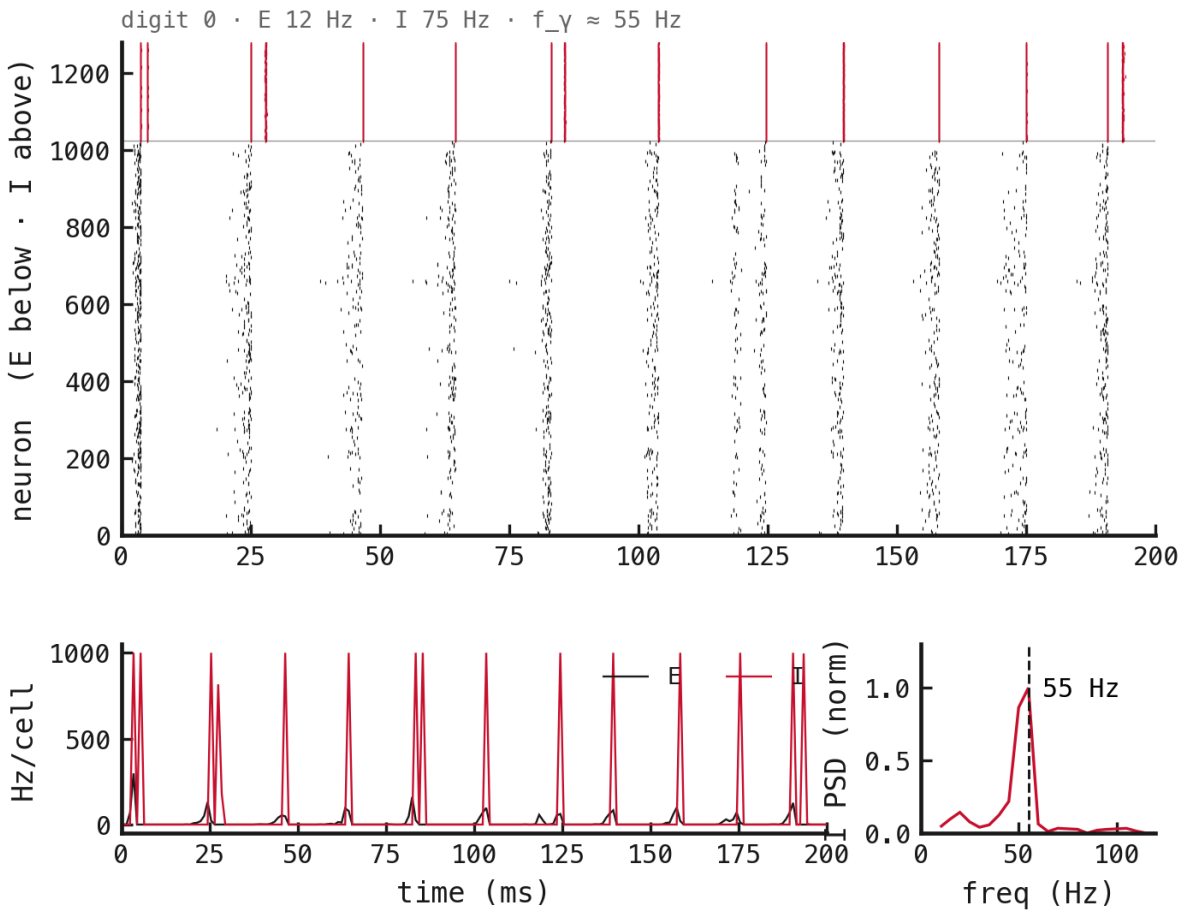


Figure 20: Sample raster: PING at $\Delta t = 0.1$ ms, seed 42.

TR-05 results — Recurrent-initialization sweep
Run status: complete · 12/12 cells represented

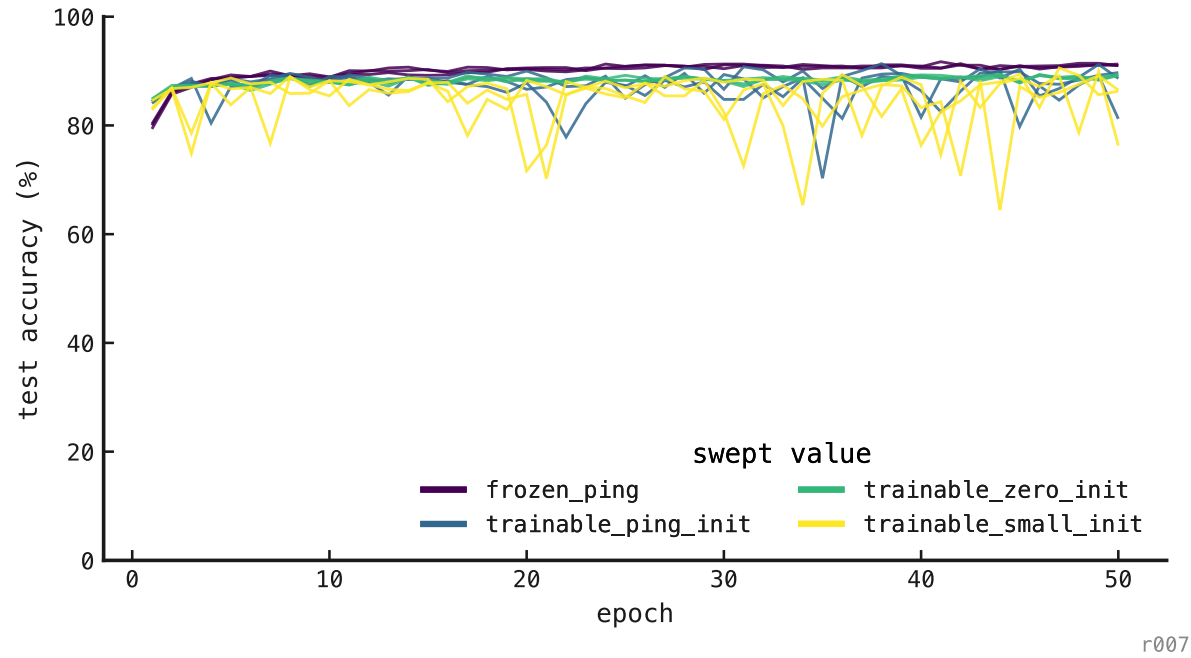


Figure 21: Training curves across four recurrent-loop conditions and three seeds.

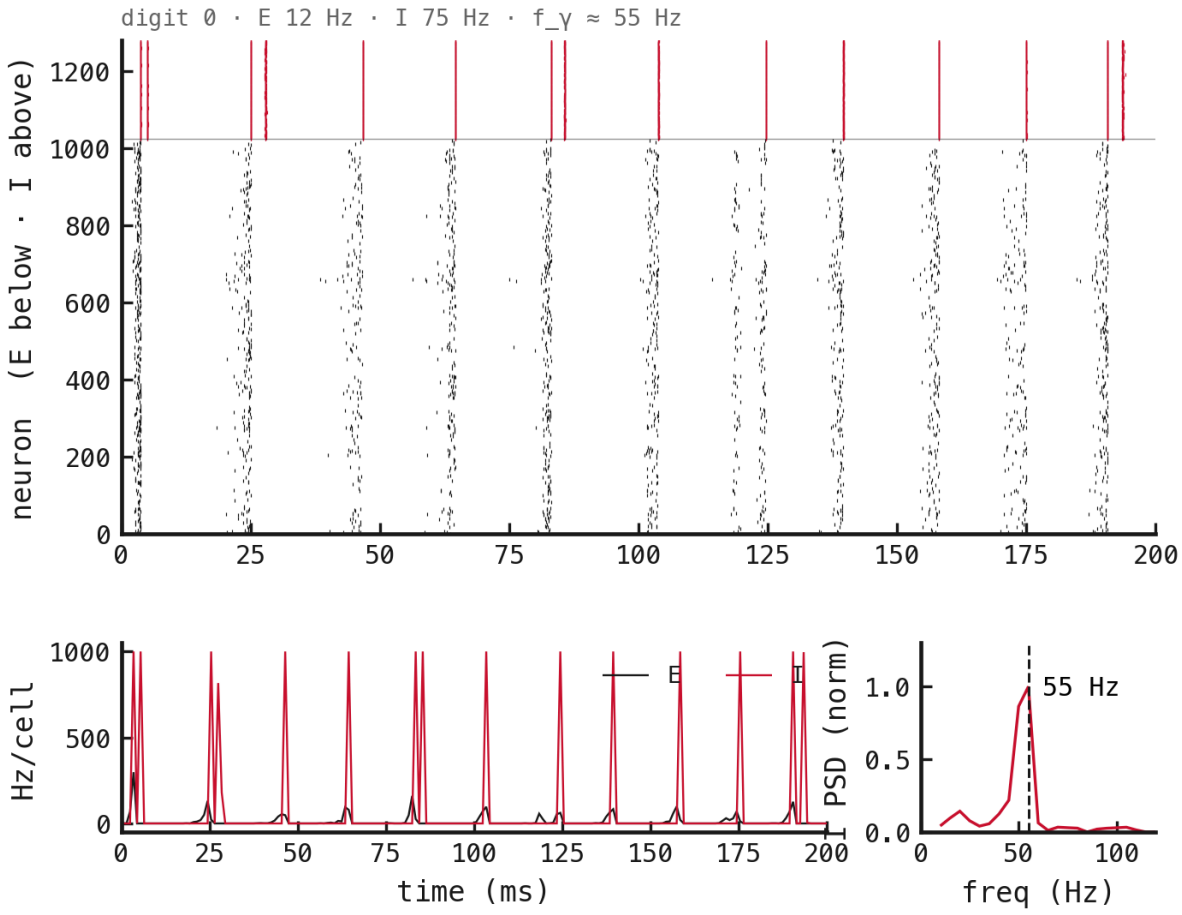


Figure 22: Sample raster: frozen PING recurrent loop, seed 42.

TR-06 results — Variable-rate streaming bank
Run status: pending · 0/3 cells trained

TODO — training curves. Add the three variable-rate learning curves after the Cambridge jobs complete.

TODO — sample raster. Add a seed-42 E/I/output raster at a declared held-out rate, plus low- and high-rate diagnostics if one raster hides a rate-dependent failure.

PING fundamentals

13 May 2026

Abstract

Strips PING down to its biophysical fundamentals and characterises it *in isolation from any task* — no training, no readout, no loss, just a free-running network driven by Poisson input. The E → I → E loop produces gamma at ≈ 30 Hz with the standard $\tau_{\text{AMPA}} + \tau_{\text{GABA}}$ cycle period, and the f–I curves show the loop is dynamic-range compression: PING holds E an order of magnitude below COBA across two orders of magnitude of drive, while COBA saturates near its refractory ceiling.

Method

The architecture characterised here is sketched atop Figure 1. An input layer drives the excitatory population through W_{in}; the E→I→E loop is formed by W_{ei} (E to I) and W_{ie} (I back to E), with no I→I synapse. Disabling the I→E loop (*ei-strength 0*) recovers the COBA reference; engaging it (*ei-strength 1.5*) produces the gamma rhythm.

The full conductance-based model that produces PING is documented in COBANet this notebook is the empirical companion. Two conditions run on the same network — *ei-strength 0* (loop disabled, the COBA reference) and *ei-strength 1.5* (loop engaged, the PING regime) — each driven by uniform Poisson input fed through W_{in} (untrained network). Because PING's loop clamps the E rate while COBA's is unclamped, a single input rate can't show both legibly: the two rasters therefore use **per-condition input rates** — COBA at 5 Hz (a legible ≈ 24 Hz async raster), PING at 45 Hz (into gamma, $f_\gamma \approx 40$ Hz). Population spectra are Welch PSDs on the population-mean E trace (one window per trial of length T, parabolic-interpolated peak; same recipe as exp041 and exp049).

The run scale reports the geometry that actually produced the figures. The two loop conditions (COBA, *ei-strength 0*; PING, *ei-strength 1.5*) are the grid; the raster/PSD input rates are per-condition (COBA 5 Hz, PING 45 Hz) and the free-running f–I panels sweep uniform Poisson drive (2–100 Hz, $T = 400$ ms).

Rate response. The f–I panels of Figure 1 sweep spatially **uniform** Poisson input (all channels at the same rate, no digit structure) on the *untrained, free-running* network, comparing COBA against PING on a shared rate axis — the bare dynamic-range compression of the architecture.

Results

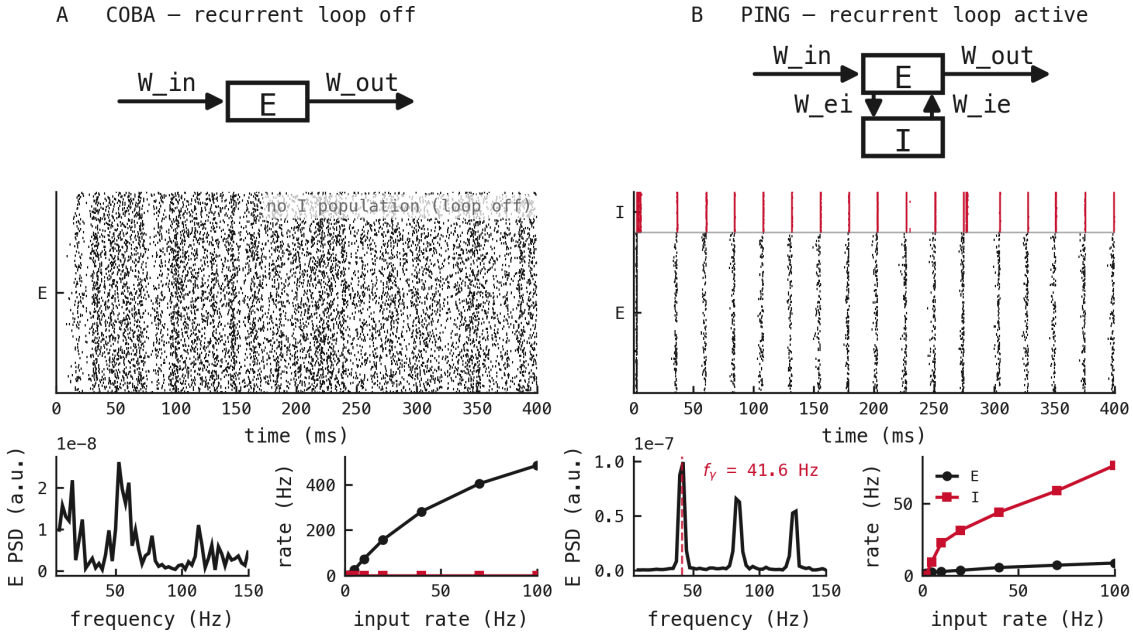


Figure 23: The architecture and its behaviour, off and on, on the same network. Each column carries its schematic on top — **A** COBA (input→E→output, no I population) and **B** PING (the same plus the recurrent E↔I loop, W_{ei} down and W_{ie} up) — directly above the plots it produces. Below each schematic is one combined raster (E black in the lower portion, I red in the upper portion — same axes, no gap) above a bottom row pairing the Welch PSD of the population-mean E trace with the free-running f–I curve (mean per-cell rate vs input rate, E black / I red; each column has its own y-axis — COBA runs to its ≈ 400+ Hz ceiling, PING is on a smaller scale so its loop-clamped E and climbing I are legible). Rasters and spectra are on uniform Poisson input (400 ms) at per-condition rates (COBA 5 Hz, PING 45 Hz — see Methods); the f–I curves sweep that drive on the untrained network. All runs are at $\Delta t = 0.1$ ms. **A — COBA** (*ei-strength 0*, loop off): asynchronous E firing, no I activity, no recurrent gamma peak, and an E f–I curve that climbs steeply with input rate and saturates toward the refractory ceiling. **B — PING** (*ei-strength 1.5*, loop on): the I population fires a synchronous burst once per cycle and the E raster breaks into vertical bands (the I volley trails each E volley by the AMPA delay); the PSD carries a clean gamma peak (dashed red line; f_γ measured from this raster's E-population PSD, marked on the plot); and its f–I shows the E rate held to single digits across the whole sweep (note the much smaller y-axis than COBA's) while inhibition climbs — the loop's dynamic-range compression. The single E→I→E loop is the only change: it converts asynchronous firing into a gamma rhythm *and* clamps the E rate.

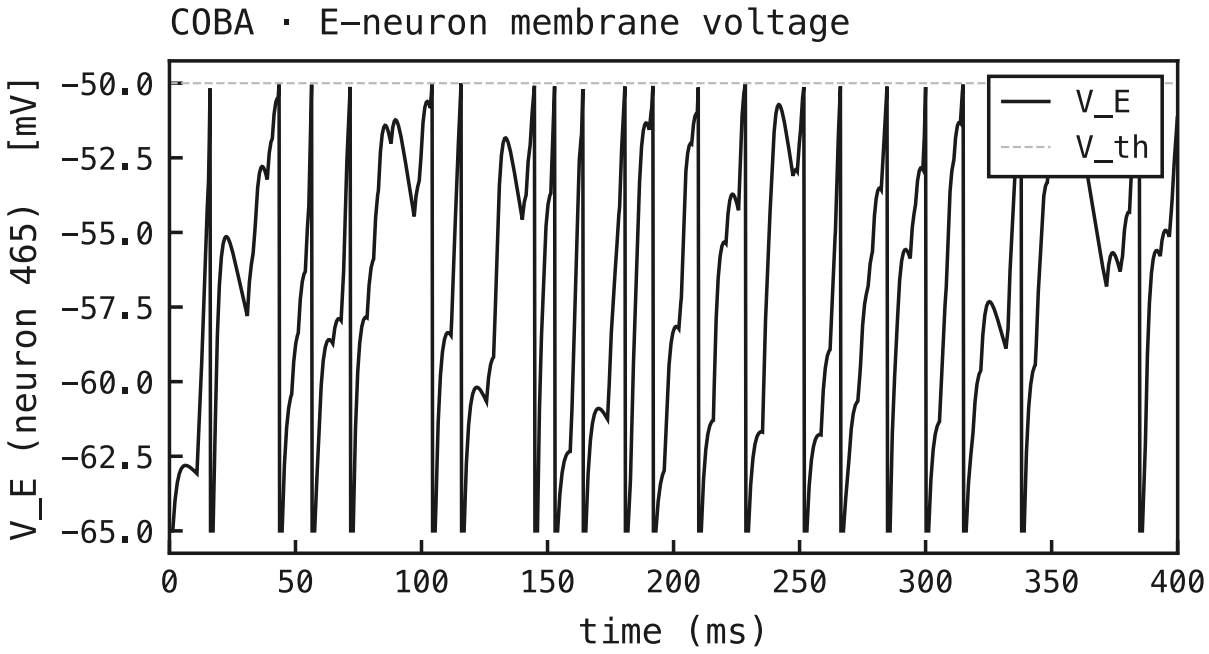


Figure 24: Membrane voltage V_E (black) of the most active E cell, recurrent loop **off** (COBA, *ei-strength 0*). Driven only by feedforward input, the cell charges toward threshold ($V_{\text{th}} = -50$ mV, faint dashed) and hard-resets to $V_{\text{reset}} = -65$ mV each time it fires — irregular, input-paced spiking with no rhythmic structure. The rhythm-free control for Figure 3a.

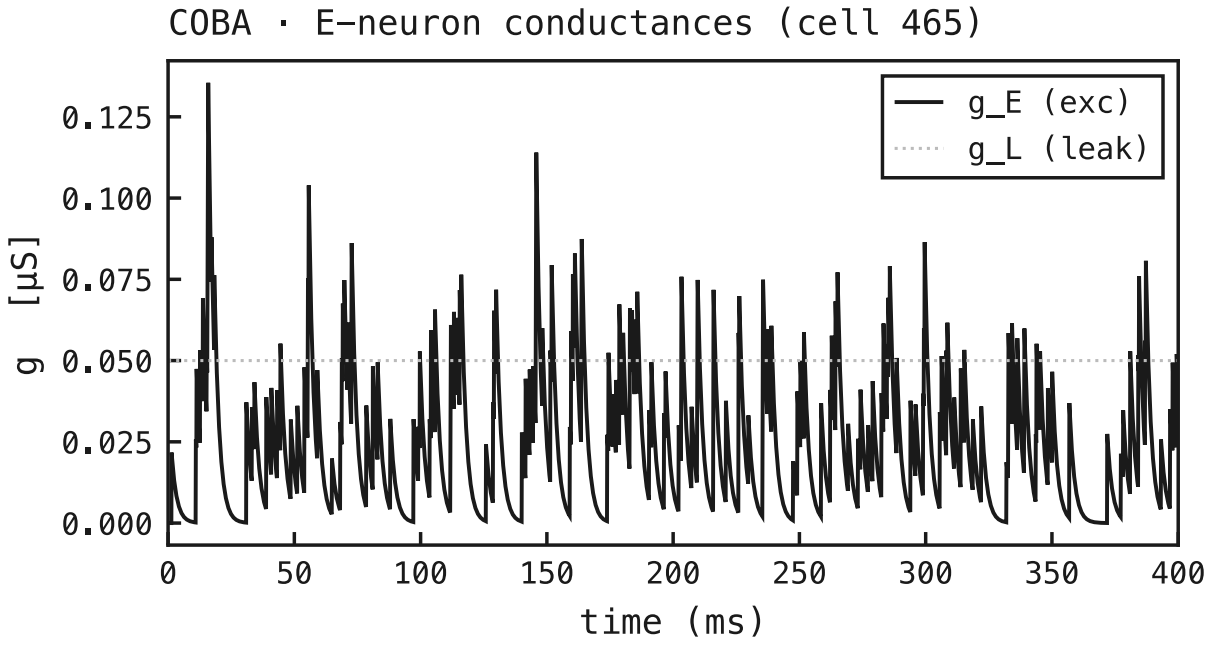


Figure 25: Synaptic and leak conductances on the same E cell, COBA mode. Only the excitatory conductance g_E (black) is active — it steps up with each input spike and decays with τ_{AMPA} — while the leak g_L (faint dotted) is fixed. There is no inhibitory g_I because the I→E loop is off. All conductances are **non-negative**: they count open channels, not current direction or sign.

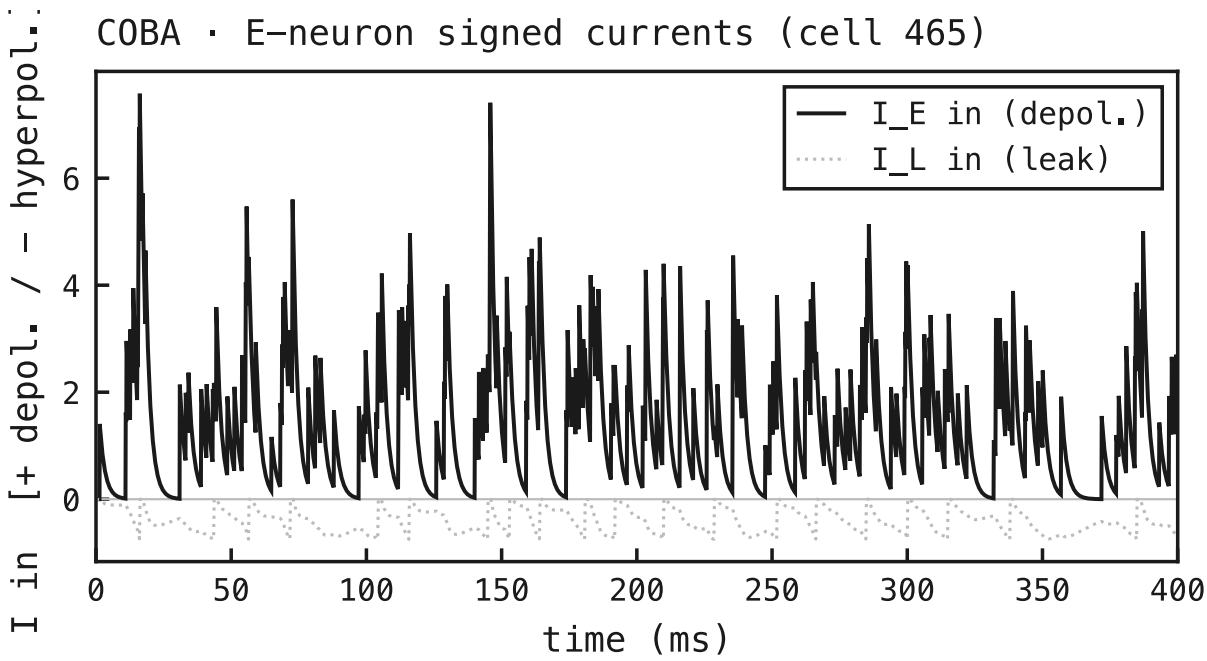


Figure 26: Signed synaptic currents into the E cell, COBA mode, with $I_X^{\text{in}} = -g_X(V - E_X)$ (positive = depolarising). With no inhibition, the only synaptic current is the depolarising excitatory current I_E^{in} (black); the leak current (faint) hovers near zero. Compare Figure 3c, where the loop introduces an inhibitory current of the *opposite* sign.

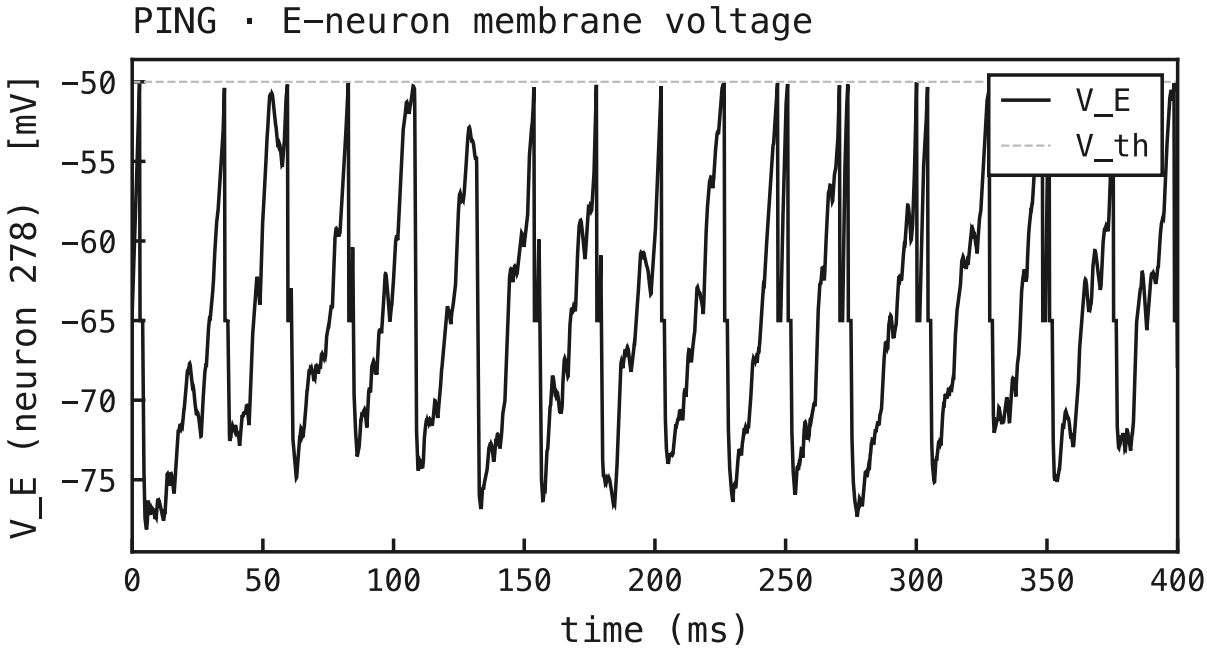


Figure 27: Membrane voltage V_E (black) of the most active E cell, recurrent loop **on** (PING, *ei-strength 1.5*), same input as 2a. Rhythmic inhibitory bursts now hold the cell below threshold most of the time: each I-burst drags V_E toward $E_i = -80$ mV, and the membrane recovers — and can fire — only as the inhibition decays between bursts. The loop has turned the irregular firing of 2a into cycle-gated firing.

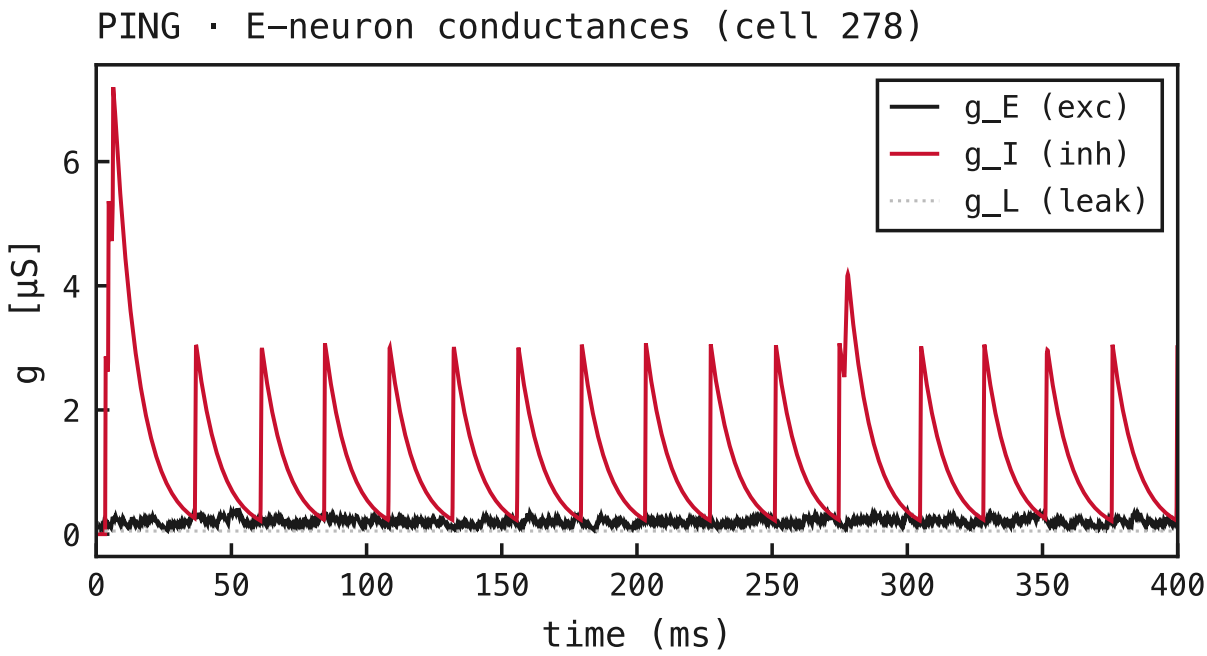


Figure 28: Conductances on the same E cell, PING mode. The inhibitory conductance g_I (red) now dominates — it spikes once per gamma cycle as the synchronous I-burst arrives through W_{ie}, then decays with τ_{GABA} — while g_E (black) carries the smaller feedforward input and g_L (faint) is fixed. All three traces are **non-negative**: g_I is large and positive, and that is what shunts the cell — it carries no sign of its own.

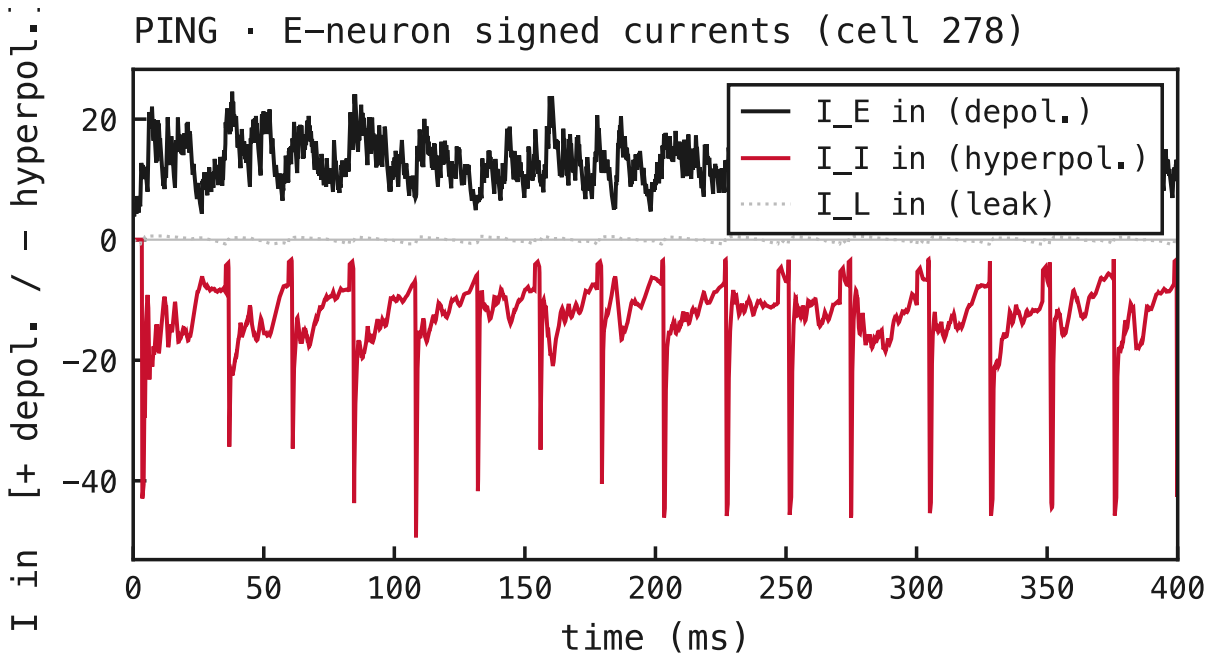


Figure 29: The load-bearing panel. Signed currents into the E cell, PING mode. The excitatory current $I_E^{\text{in}} = -g_E(V - E_E)$ (black) is depolarising; the inhibitory current $I_I^{\text{in}} = -g_I(V - E_I)$ (red) is **negative (hyperpolarising) even though g_I is positive** (Figure 3b) — the minus sign comes entirely from the driving force $(V - E_I) > 0$ whenever V sits above $E_i = -80$ mV. The COBA principle made literal: *the sign lives in the driving force, never in the conductance.* The sharp negative pulses are the per-cycle inhibitory kicks that gate the rhythm.

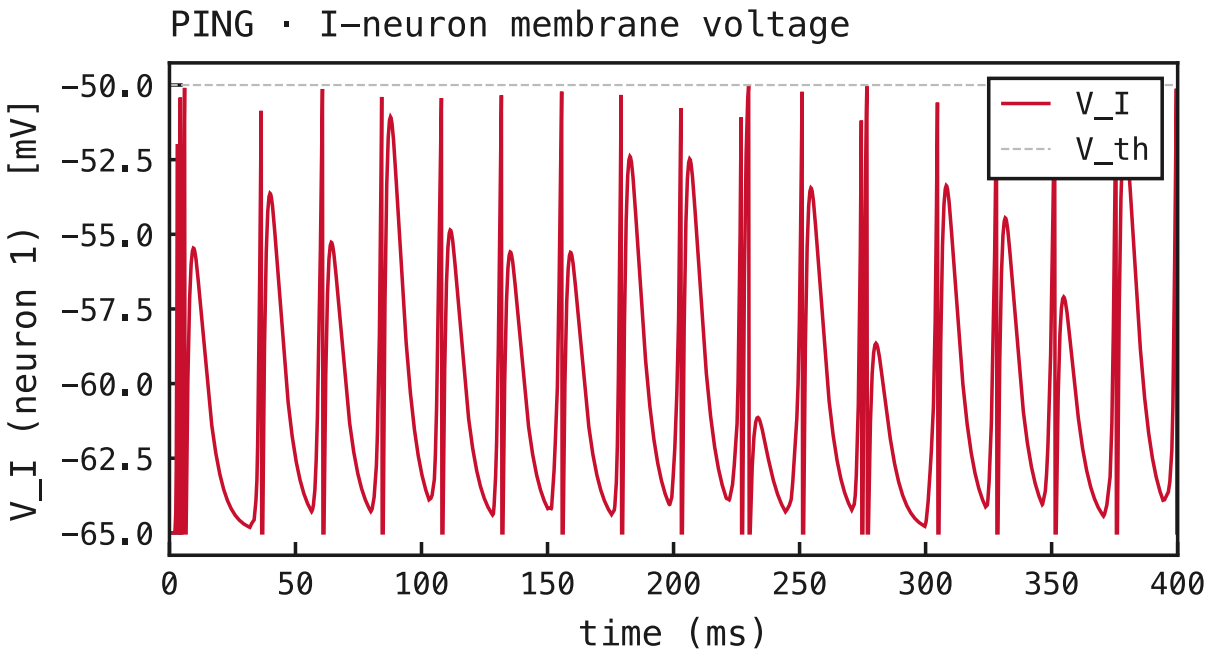


Figure 30: Membrane voltage V_I (red) of the most active I cell, PING mode. The I cell integrates the excitatory volley from the E population and fires once per gamma cycle, so V_I tracks the E-burst envelope — ramping up as E cells fire, crossing threshold ($V_{\text{th}} = -50$ mV, faint dashed), resetting, and waiting for the next volley. This single I-spike per cycle delivers the inhibitory burst seen on the E cell in 3b–3c.

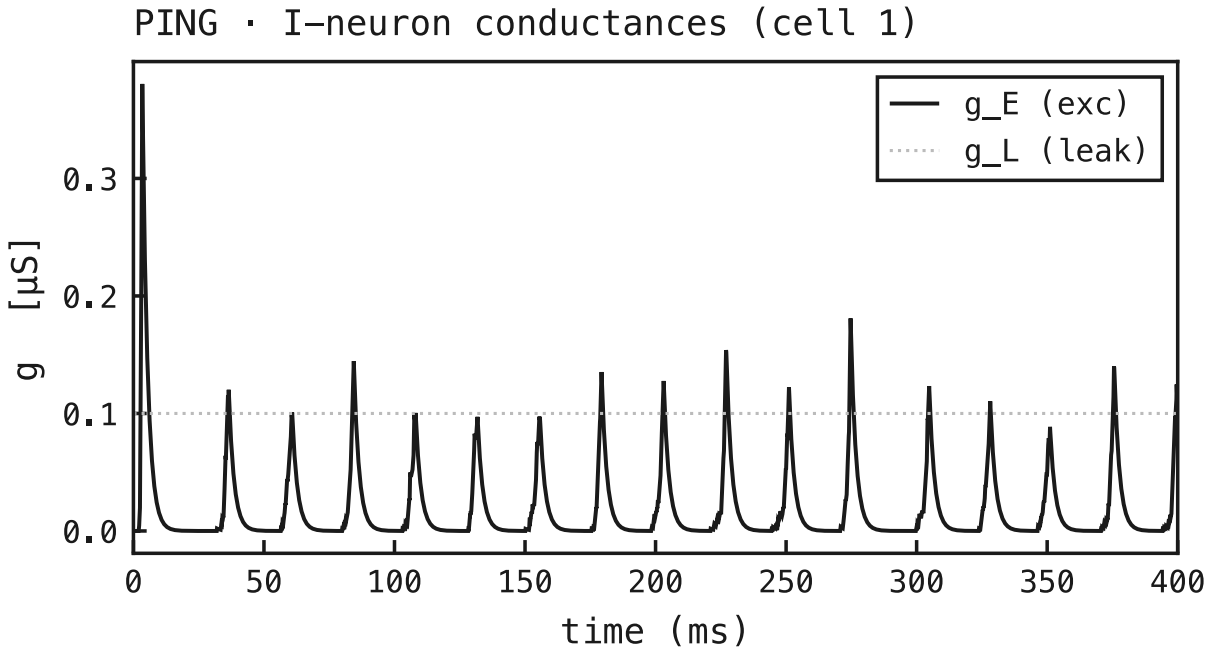


Figure 31: Conductances on the I cell, PING mode. The I cell receives **only excitation**: g_E (black) is the arriving E-spike envelope (delivered via W_{ei}) and g_L (faint) is the fixed leak. There is no inhibitory conductance — the architecture has no I→I synapse — which is exactly why the I population can synchronise into the sharp once-per-cycle bursts that clock the loop.

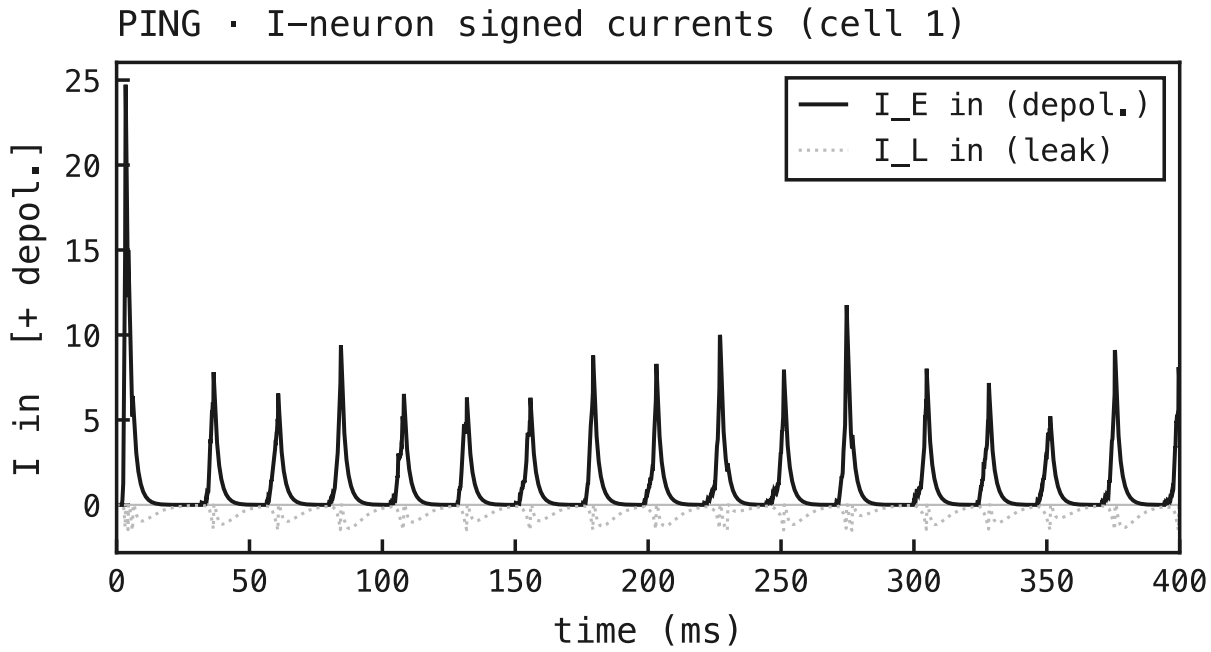


Figure 32: Signed currents into the I cell, PING mode. With no inhibitory input, only the depolarising excitatory current I_E^{in} (black) and a small leak current contribute — the I cell is driven purely up to threshold. Read 3a–3f together and the loop is in cross-section: E spikes ramp g_E on I (3e–3f) → I fires once per cycle (3d) → that spike delivers the inhibitory burst g_I on E (3b) → its negative current shuts E down (3c, 3a) → g_I decays → E refires. That delayed E→I→E loop is PING.

Accuracy converges, firing rate does not

2 June 2026

The trained networks this entry uses are produced once in the shared training hub, exp022 (Training), and reused here rather than retrained.

Abstract

Reads exp022′s PING and COBA θ_u = off baselines (three seeds each, 50 epochs) and asks whether the firing rate converges once the accuracy has. It does not, not for COBA. Test accuracy plateaus by ≈ 15 epochs in both architectures, but PING's E rate locks to a tight ≈ 10 Hz attractor (cross-seed to 0.1 Hz) while COBA's keeps climbing through epoch 50. Accuracy is a point; for COBA, the rate is a manifold the optimiser drifts along at constant accuracy.

Method

Reads the shared θ_u = off baselines (coba and ping, three seeds each) from the training hub, exp022 (Training), which fixes the recipe (50 epochs, mem-mean readout, no rate regulariser), and plots their per-epoch training history. The question, following exp041 / exp044: once test accuracy has plateaued, does the firing rate also settle? **Converged** means a last-10-epoch slope below 0.1 pp/ep (accuracy) or 0.05 Hz/ep (rate).

Results

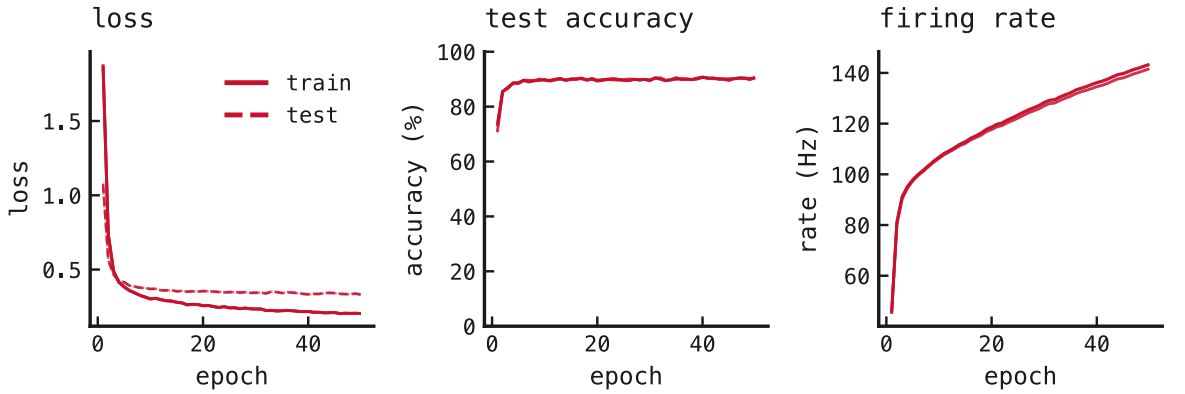


Figure 33: COBA, three seeds. Loss (train solid, test dashed) and **accuracy plateau by ≈ 15 epochs**, but the **E rate keeps climbing to ≈ 143 Hz and is still rising at epoch 50**. Accuracy has converged, the rate has not.

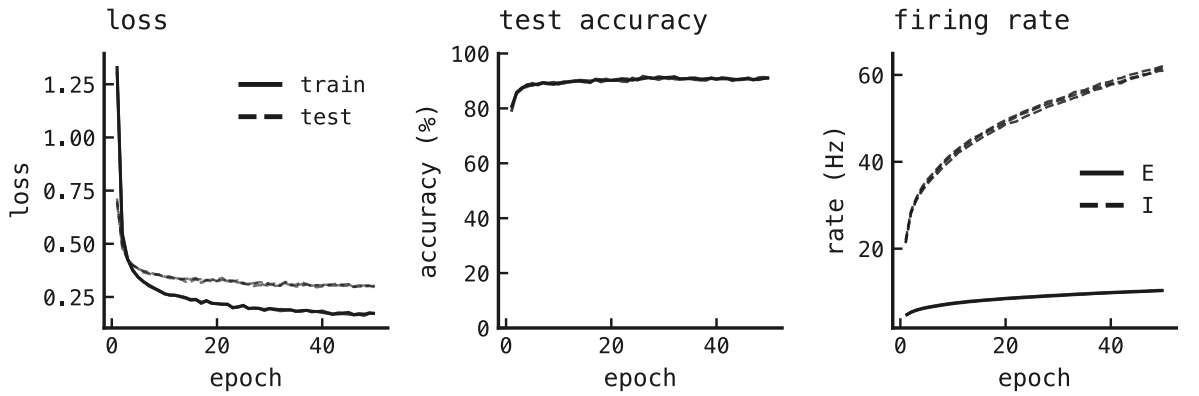


Figure 34: PING, three seeds. Loss and accuracy plateau by ≈ 15 epochs as in COBA, but here the **E rate settles into a tight band near ≈ 10 Hz**, converged. (The I rate, dashed, keeps climbing to ≈ 61 Hz, driven by the still-growing input weights, the same force that drives COBA's E rate up.) The loop pins the excitatory rate; without it, COBA's drifts.

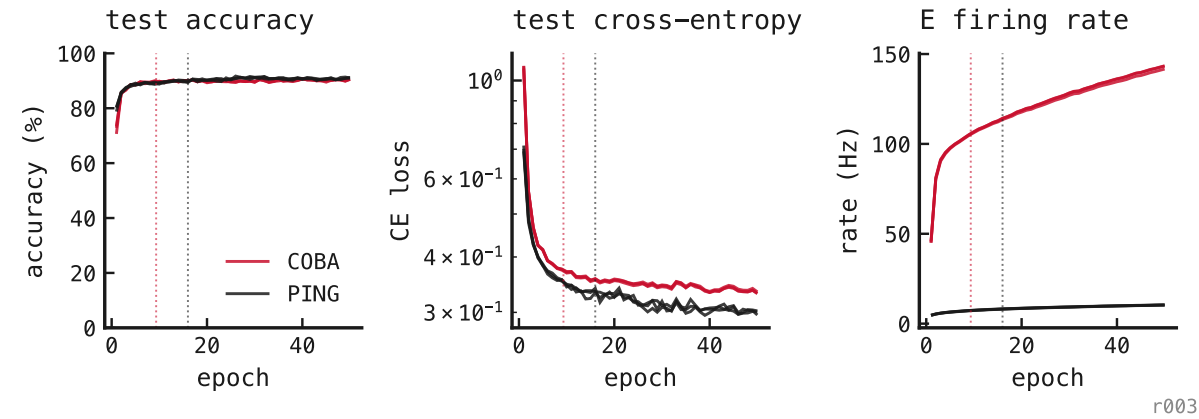


Figure 35: Test accuracy (left), test cross-entropy on a log axis (middle), and E firing rate (right) vs epoch; three seeds each, COBA red / PING black. Dotted verticals mark each model's accuracy-convergence epoch (first epoch within 1% of final accuracy). Accuracy is flat past ≈ epoch 20, yet **cross-entropy keeps falling well to its right**: COBA's is still dropping at epoch 50. COBA's E rate climbs ≈ 45 → 143 Hz in lockstep; PING reaches the same low loss early and **stays pinned near ≈ 10 Hz**.

Discussion

The rate climb is what the loss spends to keep gaining confidence. Cross-entropy stops at *certain*, not *correct*:

$$\mathrm{CE} = -\log p_y = \log\left(1 + \sum_{k \neq y} e^{z_k - z_y}\right)$$

- z_y, z_k: logits for the true class and class k
- p_y: softmax probability of the true class
- $m = z_y - \max_{k \neq y} z_k$: decision margin

Accuracy needs only the *sign* of m; cross-entropy keeps shrinking with the *size* of the gaps. So past the convergence line the argmax is fixed but the loss still falls by widening margins,

which means scaling logits up. With a mem-mean readout, $z \approx W_{\text{out}} \cdot (\text{E activity})$, so wider margins cost either weight or spikes. COBA takes the spike route (rate ≈ 45 → 143 Hz, still climbing); PING sharpens its readout through the loop and holds ≈ 10 Hz. The loop buys confidence for free; without it, confidence costs spikes.

So the rate doesn't fail to converge: it tracks a loss that never stops rewarding margin. Each cell's per-epoch metrics already record *test_margin*, *test_confidence* and *test_logit_scale*, so the prediction that COBA's margin rises with its rate while PING's plateaus can be read straight from them.

Next steps

Figure 3 infers confidence from the rate. The margin, confidence and logit scale that exp022 already logs per epoch let us measure it directly:

- Plot per-epoch **margin** $\langle \boldsymbol{m} \rangle$, **confidence** $\langle \boldsymbol{p_y} \rangle$ and **logit scale** vs epoch, COBA vs PING: the direct version of Figure 3′s middle panel.
- Overlay each against the E rate to test the lockstep: confidence up with rate for COBA, both flat for PING.
- Confirm the split quantitatively: does COBA's margin keep rising at epoch 50 (slope > 0) while PING's plateaus (slope ≈ 0), matching the cross-entropy and rate slopes here.

PING locks E rate ≈10× below COBA

30 May 2026

The trained networks this entry uses are produced once in the shared training hub, exp022 (Training), and reused here rather than retrained.

Abstract

Head-to-head comparison of COBA (recurrent inhibitory loop disabled) against PING (loop active) on MNIST under matched architecture and training recipe. PING locks the hidden E rate to ≈ 12 Hz while COBA runs at ≈ 181 Hz; accuracy is within a few points across the two, so the loop buys better than an order-of-magnitude per-spike economy. The rate-vs-accuracy frontier traced by sweeping the θ_u rate regulariser shows the floor is structural, not a trade-off the optimiser can navigate away.

Method

Training recipe (canonical / medium tier):

Parameter	Value
Integration timestep Δt	0.1 ms
Trial duration T	200 ms
MNIST samples (80/20 stratified split of 2000)	1600 train / 400 test (≈ 2.9% of the 70k-sample MNIST corpus)
Epochs	100

Two configurations of the same COBANet architecture, differing only in whether the E→I→E inhibitory loop is active. **COBA** (*–ei-strength 0*) disables the loop; excitatory cells drive each other but receive no structured inhibition. **PING** (*–ei-strength 1*) enables the loop, producing pyramidal-interneuron gamma (PING) oscillations at a cadence set by τ_{AMPA} and τ_{GABA}.

Architecture. $N_E = 1024$ excitatory cells, $N_I = 256$ inhibitory cells, single hidden layer. Input: 784 channels (MNIST pixels), Poisson-encoded at 25 Hz peak rate. Readout: mem-mean (time-averaged E spike vector projected through a trained linear layer W_{out}; see ar006 for the full readout specification). Dale's law enforced.

What is trainable. Only the input weights W_{in} (784×1024, 95% sparse) and the readout W_{out} (1024×10). The recurrent weights W_{ee} (fixed at zero, since Börgers-style PING needs no E→E coupling), W_{ei}, and W_{ie} are initialised once and held *requires_grad = False*. The synaptic time constants τ_{AMPA}, τ_{GABA} are module-level constants. 813k trainable parameters out of 2.4M total.

Training. Adam, lr $= 4 \times 10^{-4}$, batch size 256, gradient norm clipped to 1.0. Cross-entropy loss on 10-class MNIST (definition in ar006). Gradient stabiliser: *–v-grad-dampen 1000* (uniform scaling of per-step voltage gradients). Three seeds (42, 43, 44) for baselines, one seed (42) for sweep cells.

Recipe difference. The only parameter that differs between COBA and PING besides *–ei-strength* is the W_{in} initialisation: COBA uses mean 0.3 (std 0.03), PING uses mean 1.2 (std 0.12). PING needs stronger input drive to reliably recruit the I-loop at init.

Spike-budget regulariser. To probe the rate axis, the training loss adds a soft upper bound on per-trial spike count. For each E cell n with mean per-trial spike count $\bar{z}_n$:

$$L_{\text{rate}} = \lambda \sum_n \text{ReLU}(\bar{z}_n - \theta_u)^2,$$

with θ_u the per-cell budget (spikes/trial) and $\lambda = 10^{-3}$; only cells over budget contribute, cells under it are free, and the total loss is cross-entropy plus L_{rate}. We sweep six budgets (off, 5, 2, 1, 0.5, 0.2 spikes/trial; at $T = 200$ ms this is no penalty, 25, 10, 5, 2.5, 1 Hz), giving twelve (model, θ_u) cells.

Rate-floor decomposition. The affine law $r_E = p \cdot f_\gamma$ (exp041, exp046) factors the E rate into per-cycle participation p and gamma frequency f_γ. Both are measured at every cell: f_γ from the Welch PSD peak of the E-population trace, p via I-burst peak detection and per-(cell, cycle) spike counting (style of exp046).

Basin and landscape probes. To test basin attractivity, four PING networks are trained with W_{in} initialised at 0.05, 0.1, 0.3, and 1.2 ($\theta_u = 0.2$ from epoch 0; Figure 3). To map the loss landscape around the operating point, each network trained at the heaviest penalty ($\theta_u = 0.2$) has its W_{in} scaled by a common scalar $s \in [0.05, 3]$ at inference with all other weights frozen, metrics averaged over the test set at 24 values of s (Figures 4–5).

Results

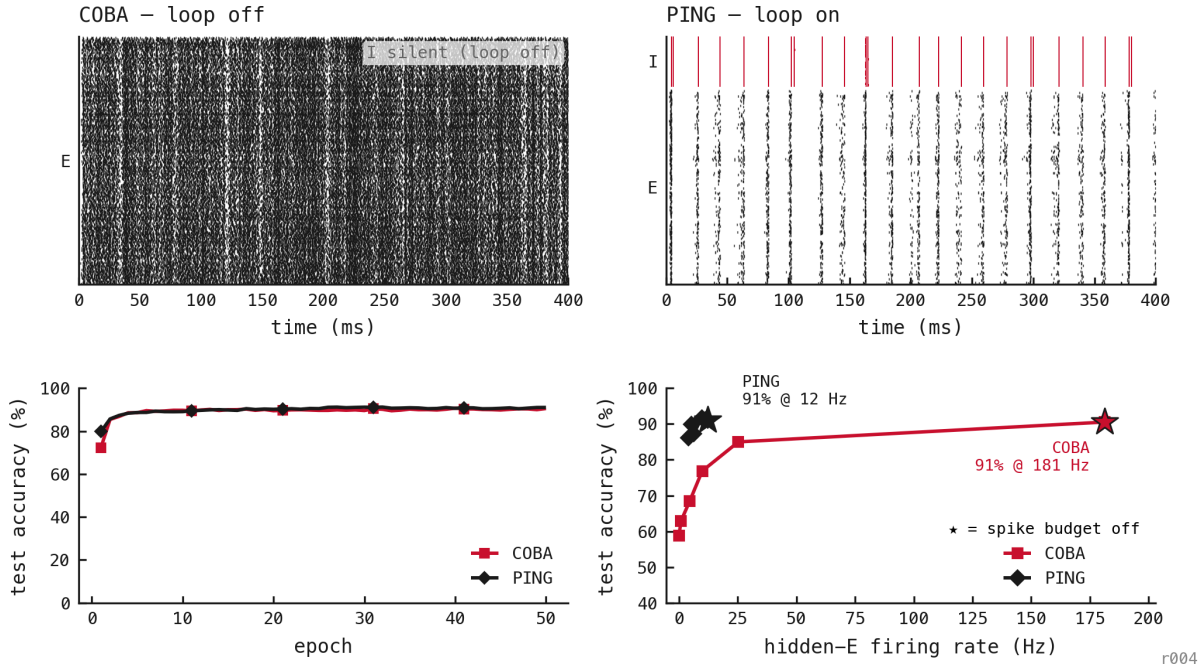


Figure 36: The headline comparison in one 2×2 frame (rasters, learning, and the accuracy–rate frontier together). **Top:** trained-baseline single-trial rasters on the same MNIST digit 0 input. COBA (*–ei-strength 0*) fires densely and asynchronously at ≈ 181 Hz (I silent, loop off); PING (*–ei-strength 1*) fires in gamma bands at ≈ 22 ms cadence (≈ 12 Hz per E cell) with synchronous I bursts (red) above E (black), on the same architecture, parameter count, and recipe, with only PING's recurrent E↔I matrices non-zero. **Bottom left:** test accuracy per epoch (mean over three seeds); both reach ≈ 91%, so both learn the task. **Bottom right:** the accuracy–rate frontier across the spike-budget penalty θ_u. PING sits up-and-left of COBA, the same accuracy at a fraction of the hidden-E rate, with the θ_u-off operating points starred and labelled (PING 91% at ≈ 12 Hz, COBA 91% at ≈ 181 Hz). COBA red, PING black; E black and I red in the rasters. *The headline of this entry: gamma gating buys an order-of-magnitude per-spike economy at matched accuracy.*

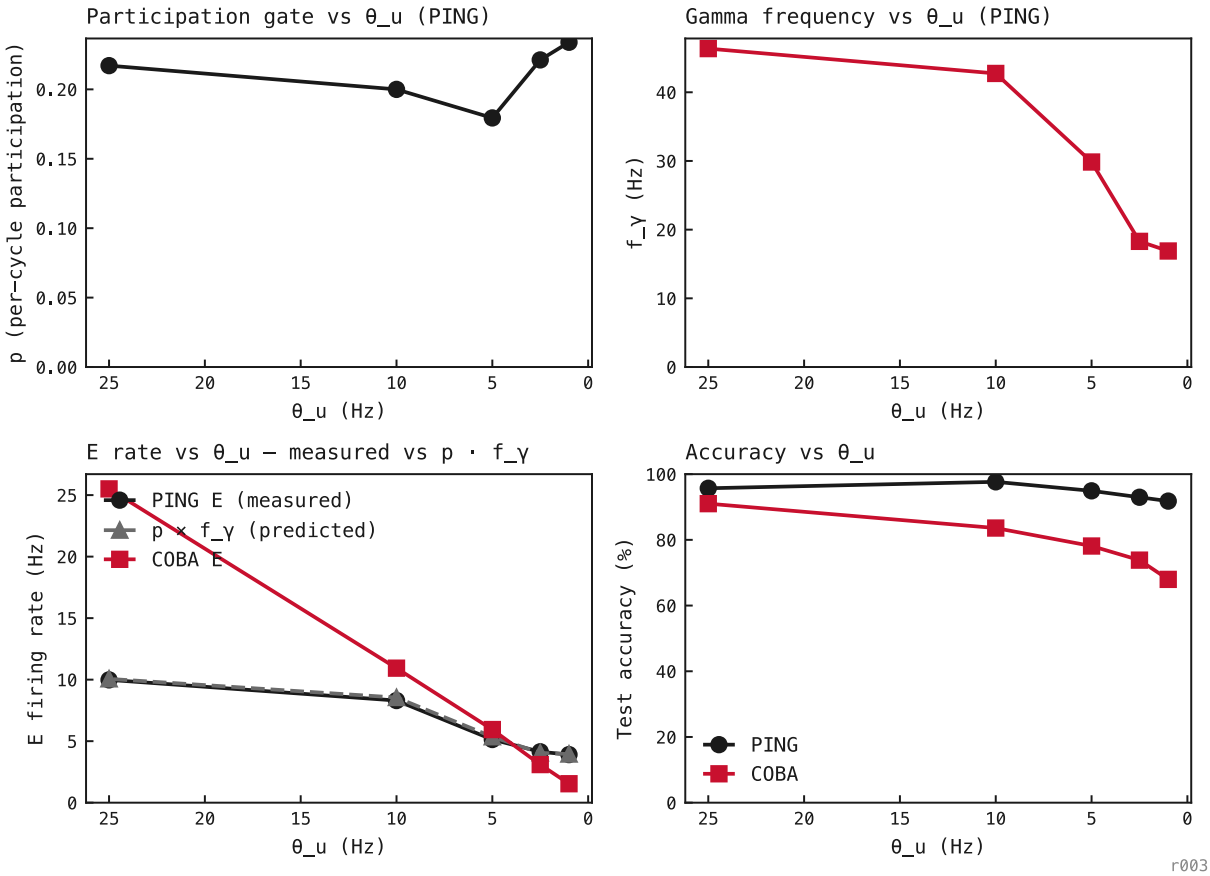


Figure 37: Five (PING, θ_u) sweep cells, 256 test trials each. ***p* stays in 0.18–0.23** across the entire sweep (the architecture protects the participation gate), while ***f*$_\gamma$ slides from ≈ 46 Hz to ≈ 17 Hz** as the penalty tightens. The grey dashed $p \cdot f_\gamma$ curve overlays the measured E rate within 4%; the rate change is entirely in f_γ. PING accuracy holds at 92–98% the whole way; COBA's collapses from 91% to 68%.

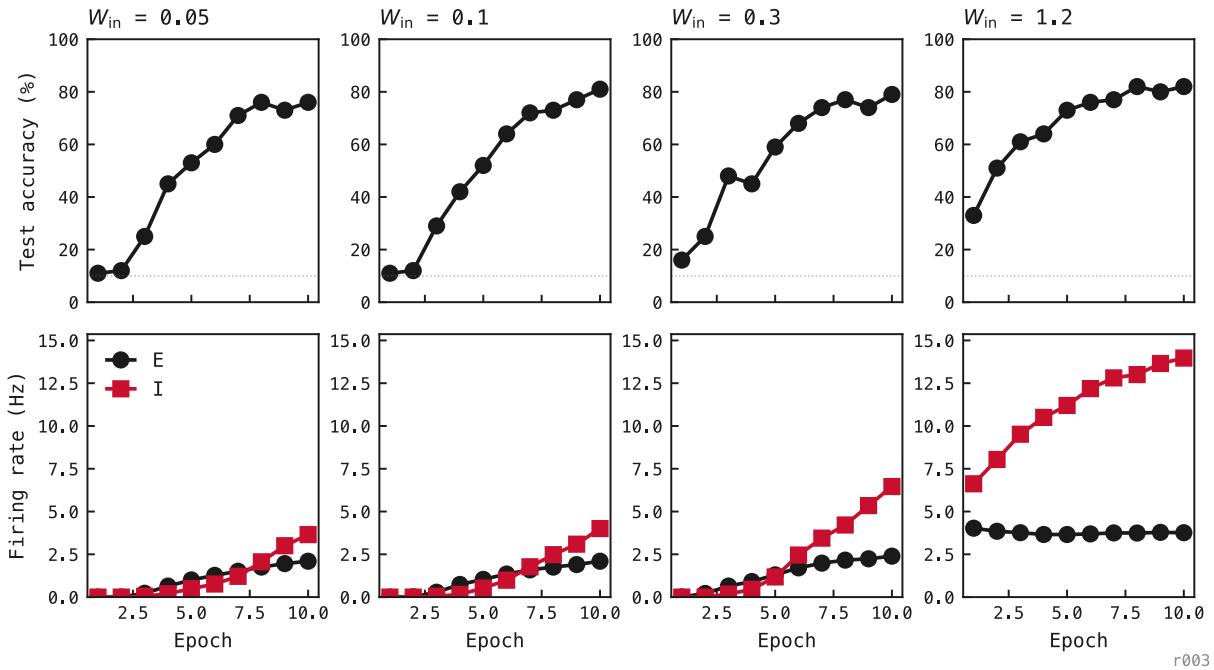


Figure 38: Per-epoch training traces from four PING networks (seed 42, $\theta_u = 0.2$ from epoch 0), one per column. Top: test accuracy. Bottom: test-set E (black) and I (red) firing rates. Recruitment is W_{in}-ordered: at $W_{in} = 0.05$ and 0.1 the I population stays silent for the first ≈ 8 epochs and engages only weakly in the final one or two; at $W_{in} = 0.3$ the loop recruits by epoch 7; at $W_{in} = 1.2$ it is active from epoch 1. The lower the input drive, the later the loop crosses the recruitment threshold. Final accuracies: 76% / 81% / 79% / 82%; final I rates: 6.9 / 2.9 / 10 / 20 Hz.

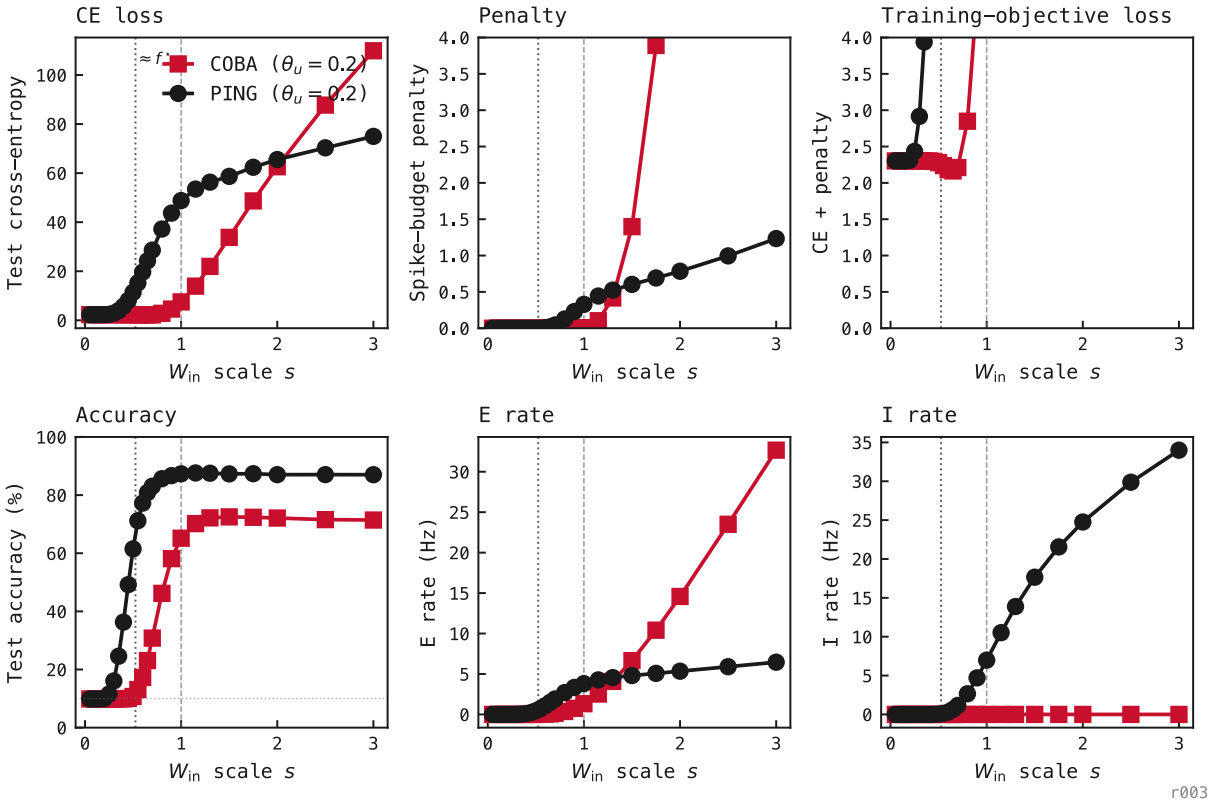


Figure 39: Inference-time W_{in} scale sweep on the two networks trained under the heaviest penalty ($\theta_u = 0.2$); every W_{in} weight multiplied by a common scalar s, all other weights frozen, 24 values of $s \in [0.05, 3]$. Top row: CE loss, spike-budget penalty L_{rate}, total objective CE + L_{rate}. Bottom row: test accuracy (chance dotted), E rate, I rate. PING black, COBA red. Vertical dashed line at $s = 1$ marks the trained operating point; dotted line marks $\approx f^*$, PING's recruitment cliff. Loss panels clipped at 4; COBA's penalty reaches ≈ 44 at $s = 3$ (rate^2 scaling).

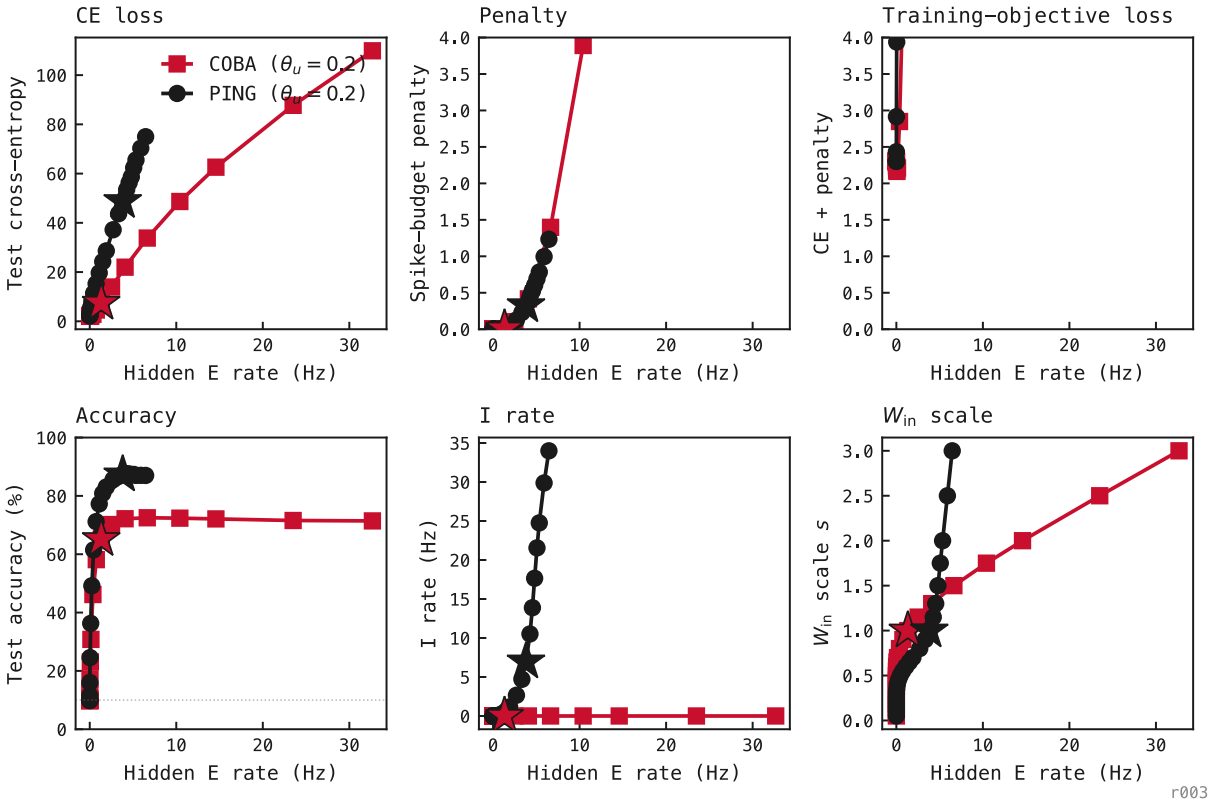


Figure 40: The same 24-point sweep as Figure 4, re-projected with hidden E rate on the x-axis. Filled stars mark each cell's trained operating point ($s = 1$): PING at $E \approx 3.8$ Hz, COBA at $E \approx 1.3$ Hz. PING's accuracy reaches its plateau ($\approx 88\%$) just past its trained operating point; COBA climbs slowly and only reaches $\approx 71\%$ even at $E \approx 33$ Hz.

DMFT model

28 May 2026

Abstract

A mean-field account of exp025′s recruitment cliff. In the 4D conductance DMFT the cliff is a Hopf bifurcation of the silent fixed point. With COBANet's own LIF f-I curve as the population gain and couplings read off the biophysics (no fitted scale), the Jacobian eigenvalues place the threshold at $I^*_{\text{ext}} = 0.6$ nA, the crossing pair's imaginary part gives a gamma rhythm at $f^* \approx 27.8$ Hz set by the synaptic timescales, and a hysteresis sweep shows the onset is supercritical: continuous and reversible.

Methods

1. The 4D DMFT model

1.1 Summary of COBA model.

We start from the COBANet model (ar003 §2): conductance-based E and I membranes, a threshold-reset rule, and three exponential synapses (no E→E; I receives no inhibition):

$$C_m^E \dot{V}^E = -g_L^E(V^E - E_L) - g_e^E(V^E - E_e) - g_i^E(V^E - E_i) \quad (1)$$

$$C_m^I \dot{V}^I = -g_L^I(V^I - E_L) - g_e^I(V^I - E_e) \quad (2)$$

$$s_{t+1} = \mathbb{1}[V \geq V_{\text{th}}], \quad V \leftarrow V_{\text{reset}} \text{ if } \; s_{t+1} = 1 \;\text{ or refractory} \quad (3)$$

$$g^E_{e,t+1} = e^{-\Delta t/\tau_{\text{AMPA}}} g^E_{e,t} + W_{\text{in}} s^{\text{inp}}_t \quad (4)$$

$$g^E_{i,t+1} = e^{-\Delta t/\tau_{\text{GABA}}} g^E_{i,t} + W_{\text{ie}} s^i_t \quad (5)$$

$$g^I_{e,t+1} = e^{-\Delta t/\tau_{\text{AMPA}}} g^I_{e,t} + W_{\text{ei}} s^e_t \quad (6)$$

1.2. Continuous-time form with tonic drive.

Recast in continuous time. The synapses (4)–(6) are the exp-Euler form of first-order filters $\tau \dot{g} = -g + \tau \sum W s$, used here as ODEs. At a constant input rate g_e^E (4) settles to a steady mean, so its excitatory current into the E membrane (1) is a near-constant depolarising drive; we replace it by a tonic current I_{ext}, the swept control parameter. The E membrane then carries I_{ext} in place of g_e^E:

$$C_m^E \dot{V}^E = -g_L^E(V^E - E_L) - g_i^E(V^E - E_i) + I_{\text{ext}} \quad (7)$$

$$C_m^I \dot{V}^I = -g_L^I(V^I - E_L) - g_e^I(V^I - E_e) \quad (8)$$

Here g_i^E is the inhibition onto E and g_e^I the excitation onto I, each a continuous-time exponential filter of the presynaptic spikes:

$$\tau_{\text{AMPA}} \dot{g}_e^I = -g_e^I + \tau_{\text{AMPA}} W^{EI} s^E(t) \quad (9)$$

$$\tau_{\text{GABA}} \dot{g}_i^E = -g_i^E + \tau_{\text{GABA}} W^{IE} s^I(t) \quad (10)$$

with $s^E(t), s^I(t)$ the population spike trains and W^{EI}, W^{IE} the recurrent weight matrices; (9)–(10) are the continuous forms of (5)–(6).

1.3. Homogeneous coupling and population means.
Now resolve the populations: index E cells by $j \in \{1, ..., N_E\}$ and I cells by $k \in \{1, ..., N_I\}$. The recurrent drive in (9)–(10) is the presynaptic sum $W^{EI}s^E = \sum_j W^{EI}_{kj} s^E_j$ (and $W^{IE}s^I = \sum_k W^{IE}_{jk} s^I_k$). Replace each random weight by its population mean, $W^{EI}_{kj} \to w^{EI}$ and $W^{IE}_{jk} \to w^{IE}$; the sums become

$$\sum_j W^{EI}_{kj} s^E_j \longrightarrow w^{EI} \sum_j s^E_j = w^{EI} N_E E(t) \quad (11)$$

$$\sum_k W^{IE}_{jk} s^I_k \longrightarrow w^{IE} \sum_k s^I_k = w^{IE} N_I I(t) \quad (12)$$

introducing the *population-mean firing rates*

$$E(t) \equiv \frac{1}{N_E} \sum_{j=1}^{N_E} s^E_j(t), \quad I(t) \equiv \frac{1}{N_I} \sum_{k=1}^{N_I} s^I_k(t). \quad (13)$$

A *smooth-rate ansatz* (short-window averaging) treats $E(t), I(t)$ as continuous, dropping weight heterogeneity and finite-size noise, the shot noise $\mathrm{Var}[E(t)] \propto E(t)/N_E$ that vanishes only as $N_E, N_I \to \infty$. At finite $N_E = 1024, N_I = 256$ the residual $O\big(N^{-1/2}\big)$ fluctuations smear the Hopf onset and sustain a weak noisy gamma below threshold, both seen in the spiking simulations.

With no cell index left, every E cell sees the same g^E_i and every I cell the same g^I_e, collapsing the per-cell conductances to population means. Defining lumped couplings

$$\widetilde{W}^{EI} \equiv w^{EI} N_E, \quad \widetilde{W}^{IE} \equiv w^{IE} N_I \quad (14)$$

the conductance dynamics become

$$\tau_{\mathrm{AMPA}} \dot{g}^I_e = -g^I_e + \tau_{\mathrm{AMPA}} \widetilde{W}^{EI} E \quad (15)$$

$$\tau_{\mathrm{GABA}} \dot{g}^E_i = -g^E_i + \tau_{\mathrm{GABA}} \widetilde{W}^{IE} I \quad (16)$$

Two equations, down from $N_E + N_I$. (The fan-in scale $\widetilde{W}$ folds into W^{EI}, W^{IE} in 1.6.)

Running system, end of 1.3: conductances are now two population means; the membrane is still per-cell but sees those means:

$$\begin{aligned}
C^E_m \dot{V}^E_j &= -g^E_L \big(V^E_j - E_L\big) - g^E_i \big(V^E_j - E_i\big) + I_{\mathrm{ext}} \\
C^I_m \dot{V}^I_k &= -g^I_L \big(V^I_k - E_L\big) - g^I_e \big(V^I_k - E_e\big) \\
\tau_{\mathrm{AMPA}} \dot{g}^I_e &= -g^I_e + \tau_{\mathrm{AMPA}} \widetilde{W}^{EI} E \\
\tau_{\mathrm{GABA}} \dot{g}^E_i &= -g^E_i + \tau_{\mathrm{GABA}} \widetilde{W}^{IE} I
\end{aligned}$$

1.4. Driving-force linearisation.
The synaptic current is conductance times a *driving force*, $-g(V - E_{\mathrm{rev}})$, a g–V product, hence nonlinear. Freeze V at rest, $V_{\mathrm{rest}} = E_L = -65$ mV, *in the driving force only* (leak and threshold keep their full V-dependence, handled by the f-I curve in 1.5). Each driving force becomes a fixed voltage gap:

$$\Delta V_{\mathrm{inh}} \equiv V_{\mathrm{rest}} - E_i = -65 - (-80) = 15 \text{ mV} \quad (17)$$

$$\Delta V_{\text{exc}} \equiv V_{\text{rest}} - E_e = -65 - 0 = -65 \text{ mV}$$

The synaptic currents in (7)–(8) then lose their V-dependence and become proportional to conductance alone:

$$-g_i^E\big(V_j^E - E_i\big) \approx -g_i^E \Delta V_{\text{inh}} \quad (18)$$

$$-g_e^I\big(V_k^I - E_e\big) \approx -g_e^I \Delta V_{\text{exc}} = +g_e^I \cdot |E_e - V_{\text{rest}}| \quad (19)$$

(inhibition pulls V down, $\Delta V_{\text{inh}} = +15$ mV; excitation pushes it up, $|\Delta V_{\text{exc}}| = 65$ mV). Removing the g–V coupling reduces COBA to a current-based (CUBA) form; the cost is shunting: with V fixed we ignore that conductance also lowers the effective time constant ($\tau_{\text{eff}} = C_m/g_{\text{tot}}$).

Running system, end of 1.4: the synaptic currents are now linear in conductance (no V left in the driving force):

$$\begin{aligned}
C_m^E \dot{V}_j^E &= -g_L^E\big(V_j^E - E_L\big) - g_i^E \Delta V_{\text{inh}} + I_{\text{ext}} \\
C_m^I \dot{V}_k^I &= -g_L^I\big(V_k^I - E_L\big) + g_e^I \ |\Delta V_{\text{exc}}| \\
\tau_{\text{AMPA}} \dot{g}_e^I &= -g_e^I + \tau_{\text{AMPA}} \widetilde{W}^{EI} E \\
\tau_{\text{GABA}} \dot{g}_i^E &= -g_i^E + \tau_{\text{GABA}} \widetilde{W}^{IE} I
\end{aligned}$$

1.5. Population rate from an f-I curve.
Under (17)–(19) the membrane equations (7)–(8) read $C_m \dot{V} = -g_L(V - E_L) + I_{\text{syn}}$, LIF with a synaptic current. A LIF cell under constant net current I fires at its f-I rate $\varphi(I)$; replacing each cell's spikes by that rate gives

$$E(t) \approx \varphi_E\big(I_{\text{eff}}^E(t)\big), \quad I(t) \approx \varphi_I\big(I_{\text{eff}}^I(t)\big) \quad (20)$$

with effective input currents (from (7)–(8) with (17)–(19) substituted)

$$I_{\text{eff}}^E(t) = I_{\text{ext}}(t) - g_i^E(t) \Delta V_{\text{inh}} \quad (21)$$

$$I_{\text{eff}}^I(t) = g_e^I(t) \ |\Delta V_{\text{exc}}| \quad (22)$$

(I receives only excitation; E receives the drive minus the GABA shunt.) The instantaneous-rate replacement (20) holds only for slow inputs; in reality $E(t)$ relaxes toward the f-I fixed point on τ_E, which we encode explicitly:

$$\tau_E \dot{E} = -E + \Phi_E(I_{\text{ext}} - g_i^E \Delta V_{\text{inh}}) \quad (23)$$

$$\tau_I \dot{I} = -I + \Phi_I(g_e^I \ |\Delta V_{\text{exc}}|) \quad (24)$$

where Φ_E, Φ_I are the smooth steady-state gain functions (COBANet's LIF f-I curve in the numerics; see Locating the Hopf). Two more equations down, together with (15)–(16), four equations in (E, I, g_e^I, g_i^E).

Running system, end of 1.5: a closed 4D rate model in (E, I, g_e^I, g_i^E), constants not yet absorbed:

$$\tau_E \dot{E} = -E + \Phi_E(I_{\text{ext}} - g_i^E \Delta V_{\text{inh}})$$
$$\tau_I \dot{I} = -I + \Phi_I(g_e^I \, |\Delta V_{\text{exc}}|)$$
$$\tau_{\text{AMPA}} \dot{g}_e^I = -g_e^I + \tau_{\text{AMPA}} \widetilde{W}^{EI} E$$
$$\tau_{\text{GABA}} \dot{g}_i^E = -g_i^E + \tau_{\text{GABA}} \widetilde{W}^{IE} I$$

1.6. Absorb the driving-force constants.
The prefactors $\Delta V_{\text{inh}}, |\Delta V_{\text{exc}}|$ in (23)–(24) and the fan-in scalings in (15)–(16) are constants carrying no dynamics; fold them into the couplings:

$$W^{EI} \equiv \widetilde{W}^{EI} \cdot |\Delta V_{\text{exc}}|, \quad W^{IE} \equiv \widetilde{W}^{IE} \cdot \Delta V_{\text{inh}} \quad (25)$$

and absorb $|\Delta V_{\text{exc}}|$ into the I-cell argument by redefining $g_e^I \mapsto g_e^I \cdot |\Delta V_{\text{exc}}|$ (similarly g_i^E). The conductances now carry current units and the f-I curves take their argument directly: a change of variables, no dynamics lost.

1.7 The 4D system
After 1.1–1.6, the mean-field equations are

$$\tau_E \dot{E} = -E + \Phi_E(I_{\text{ext}} - g_i^E), \quad \tau_I \dot{I} = -I + \Phi_I(g_e^I) \quad (26)$$

$$\tau_{\text{AMPA}} \dot{g}_e^I = -g_e^I + \tau_{\text{AMPA}} W^{EI} E, \quad \tau_{\text{GABA}} \dot{g}_i^E = -g_i^E + \tau_{\text{GABA}} W^{IE} I \quad (27)$$

in state (E, I, g_e^I, g_i^E). Whether 4D is minimal, or a 2D reduction would do, is settled in the appendix, Is it really 4D?

1.8 4D Jacobian
At a fixed point $(E^*, I^*, g_e^{I*}, g_i^{E*})$:

$$J = \begin{pmatrix} -1/\tau_E & 0 & 0 & -\Phi_E'/\tau_E \\ 0 & -1/\tau_I & \Phi_I'/\tau_I & 0 \\ W^{EI} & 0 & -1/\tau_{\text{AMPA}} & 0 \\ 0 & W^{IE} & 0 & -1/\tau_{\text{GABA}} \end{pmatrix} \quad (28)$$

with Φ_E', Φ_I' evaluated at the fixed-point arguments.

2. Locating the Hopf

2.1 The Hopf condition and sweep
Below the cliff the network is silent: nudge it and the perturbation dies away. Above the cliff the same nudge grows into a sustained rhythm. The switch is a Hopf bifurcation of the silent fixed point: the smallest drive I_{ext}^* at which the fixed point stops damping oscillations and starts amplifying them.

Linear stability makes this precise. Each mode of the linearised system evolves as $e^{\lambda t}$, where λ is an eigenvalue of the Jacobian (28): a negative real part decays, a positive one grows. The Hopf is the instant an oscillating mode (a complex-conjugate pair) crosses from the left half-plane to the right ($\text{Re}\,\lambda = 0$, $\text{Im}\,\lambda \neq 0$), so the silent state gives way to a growing oscillation rather than a static shift. To find it we sweep I_{ext}, track the fixed point, and diagonalise J at each step; I_{ext}^* is the first drive where the leading pair reaches the axis (Figure 2). The local analysis holds only if a single pair crosses while the rest stay damped (a simple Hopf); two

pairs crossing at once (a double-Hopf) would seed tori the linearisation cannot see. Timescales: $\tau_E = 20$, $\tau_I = 5$, $\tau_{\text{AMPA}} = 2$, $\tau_{\text{GABA}} = 6$ ms.

2.2 Gain and couplings from COBANet

The gain Φ and the couplings come from COBANet, not from a fit. For Φ we use the LIF f-I curve: a cell under white-noise input of mean μ and standard deviation σ_V fires at the Siegert rate

$$\varphi(\mu) = \left[\tau_{\text{ref}} + \tau_m\sqrt{\pi}\int_{(V_{\text{reset}}-\mu_V)/\sigma_V}^{(V_{\text{th}}-\mu_V)/\sigma_V} e^{u^2}(1+\operatorname{erf} u)\,\mathrm{d}u\right]^{-1}, \quad \mu_V = E_L + \mu/g_L \tag{29}$$

with COBANet's $\tau_m, g_L, V_{\text{th}}, V_{\text{reset}}, \tau_{\text{ref}}$ per population. The couplings are $W^{EI} = w^{EI}N_E\,|\Delta V_{\text{exc}}|$ and $W^{IE} = w^{IE}N_I\Delta V_{\text{inh}}$ (eqs 14, 25): the fan-in normalisation ($w \mapsto w/N$) makes $w^{EI}N_E, w^{IE}N_I$ the ei-strength values s and rs ($\approx$ 1 and 2 µS), times the driving forces (17). With Φ in physical units I_{ext} is a current in nanoamps: the absolute scale is fixed by the biophysics, not chosen. The membrane-noise std σ_V is the one free parameter, set to 4 mV; the located Hopf is insensitive to it (f^* moves under 1 Hz over $\sigma_V \in [3, 6]$ mV).

3. Calculating its frequency

At the crossing $\lambda = \pm i\omega^*$, so $f^* = \omega^*/2\pi$ is read from the Jacobian at threshold. To test whether inhibitory decay sets the period, we re-find the Hopf at each τ_{GABA} of exp041′s retrained sweep and compare f^* to the spiking f_γ (Figure 3).

4. Classifying sub- or supercritical

4.1 The hysteresis sweep

Above the Hopf a limit cycle exists; whether it appears gently or abruptly decides whether exp025′s recruitment cliff is the graded, reversible onset that picture assumes. We test it with a hysteresis sweep: step I_{ext} quasi-statically up through I^*_{ext} and back down, at each step integrating (26)–(27) to steady state from the previous step's end state (so any coexisting cycle is carried along), and record the peak-to-peak amplitude $A = \max_t E - \min_t E$.

A *stable* cycle born at threshold makes the rising and falling branches coincide, with A returning to zero at I^*_{ext}: a reversible, supercritical onset. An *unstable* cycle sitting below threshold instead acts as a basin boundary: the silent state jumps to a distant large-amplitude cycle that survives as the drive is lowered back, so the branches split into a hysteresis loop with a bistable window: subcritical (Figure 4).

4.2 The cycle waveform

The same integration gives the cycle's shape above onset: E and I are near-sinusoidal, with the E burst leading the I burst by the round-trip synaptic delay: E recruits I, I shunts E a few milliseconds later (Figure 5).

Results

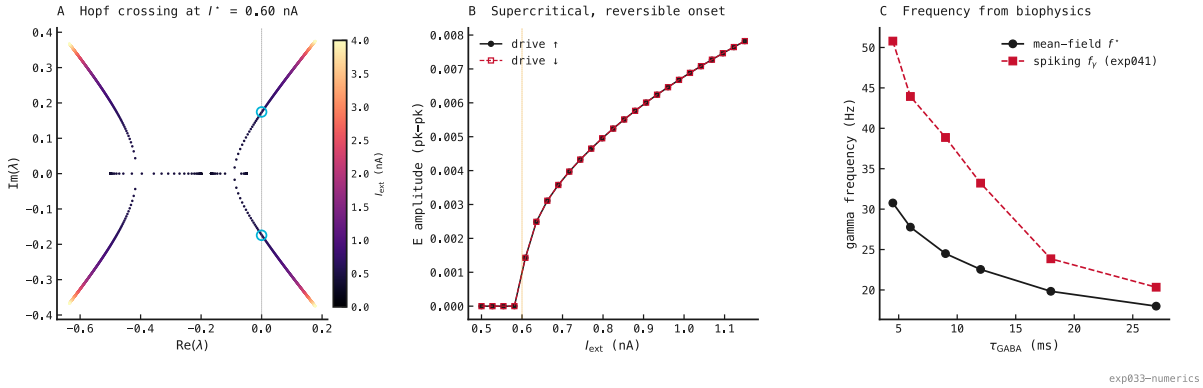


Figure 41: Summary of the analysis; the detailed panels follow. **A**, the Hopf: one eigenvalue pair crosses into the right half-plane at $I^* = 0.6$ nA, fixing a gamma rhythm at $f^* \approx 27.8$ Hz (full plot, Figure 2). **B**, the onset is supercritical and reversible, the up/down branches coinciding (Figure 4). **C**, the predicted frequency falls with τ_{GABA}, tracking the spiking measurement (Figure 3). exp025′s recruitment cliff is a supercritical Hopf, set by the synaptic time constants, derived below from COBANet's f-I curve and biophysical couplings with no fitted scale.

Frequency

Sweeping I_{ext} and diagonalising J at each step locates the Hopf at

$$I^*_{\mathrm{ext}} = 0.6 \text{ nA}, \quad \omega^* = 0.175 \text{ rad/ms}, \quad f^* = \frac{\omega^*}{2\pi} \approx 27.8 \text{ Hz},$$

in the PING gamma band, from COBANet's f-I curve and biophysical couplings with no fitted scale. One complex pair crosses the imaginary axis while the others stay damped, a simple Hopf (Figure 2). The predictive content is the *dependence* on the synaptic timescales: across a τ_{GABA} sweep f^* tracks the spiking network's gamma qualitatively, not quantitatively (Figure 3).

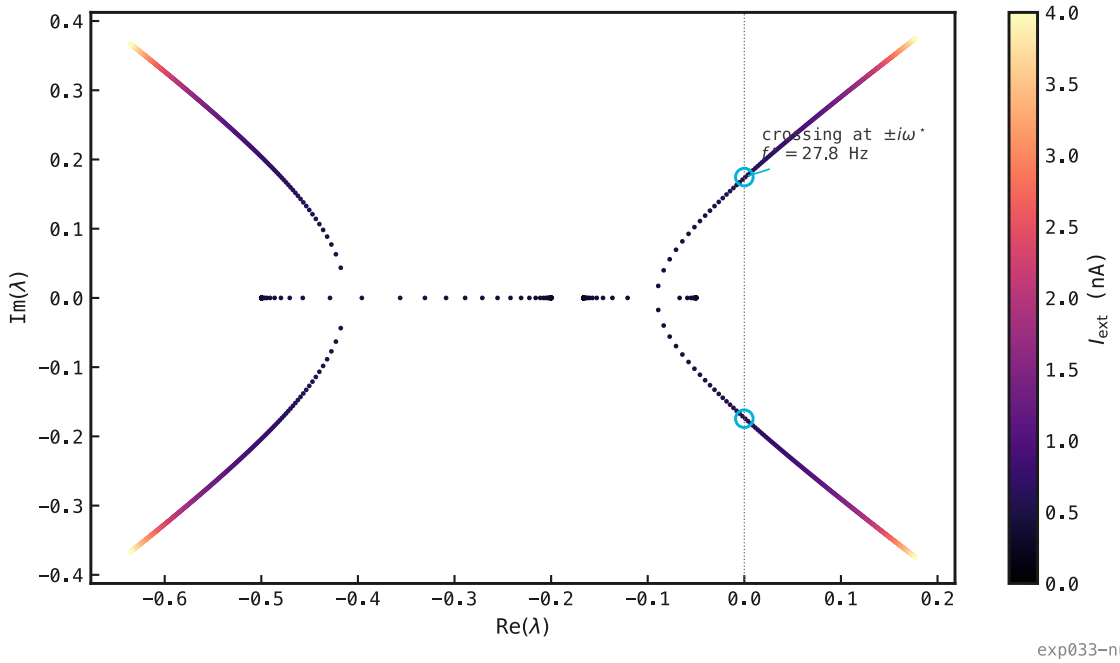


Figure 42: **How to read it.** Each dot is one eigenvalue λ of the Jacobian (28). Its horizontal position is the growth rate Re λ (negative means that mode decays, positive means it grows), and the dotted line at Re $\lambda = 0$ is the stability boundary: everything to its left is damped. Its vertical position is the oscillation frequency Im λ (zero = a non-oscillating mode, on the real axis). The Jacobian is 4×4, so each drive contributes four dots; sweeping I_{ext} (colour, dark→bright) drags them along the curves shown. Follow the upper and lower arms: one complex-conjugate pair lifts off the real axis and marches rightward, touching the boundary at $\pm i\omega^*$ (cyan circles) when $I^*_{\text{ext}} = 0.6$ nA. That crossing is the Hopf, where the network tips from damping to amplifying, and the height of the crossing sets the rhythm, $f^* = \omega^*/2\pi \approx 27.8$ Hz. The other two eigenvalues stay left of the line throughout, so a single oscillation goes unstable: a simple Hopf. (A second pair reaches the axis only at far higher drive, above the recruitment cliff.)

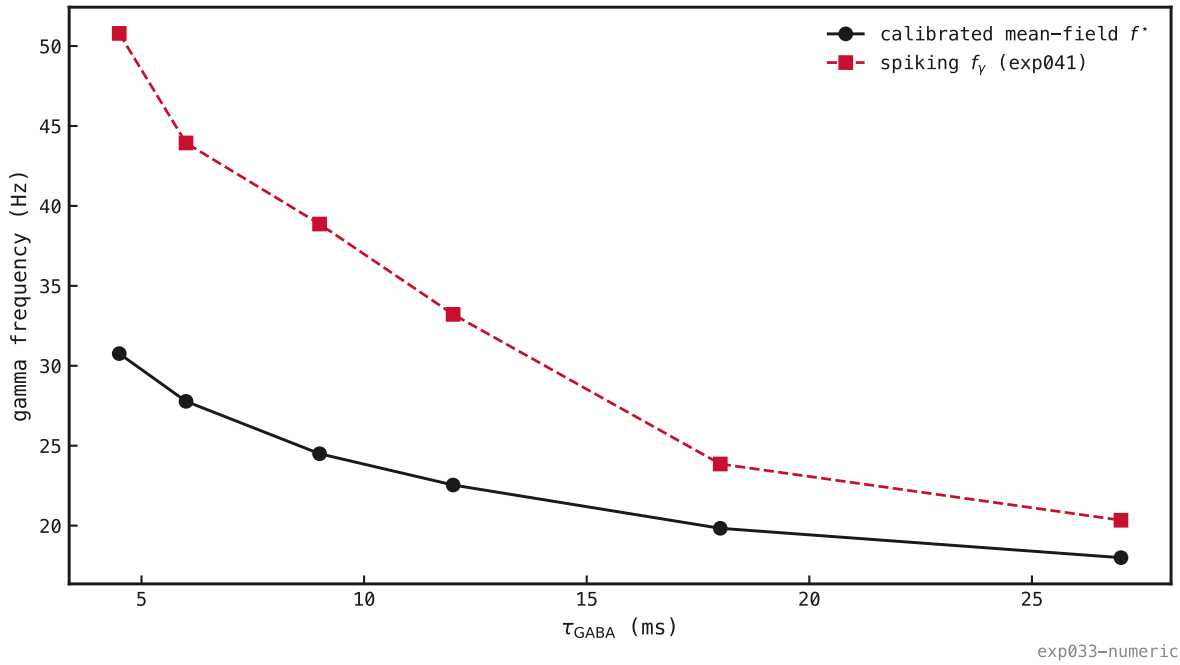


Figure 43: Both fall monotonically with τ_{GABA}: the inhibitory decay is the clock in both. The match is qualitative, not quantitative. The calibrated mean-field is *flatter* and runs below the spiking network across the sweep, the gap largest at short τ_{GABA} (27.8 vs 43.9 Hz at the canonical 6 ms) and narrowing as τ_{GABA} grows. The reduction captures the mechanism but not the full sensitivity, expected since it drops the spike-synchrony of the E-volley that sharpens the cycle most at short τ_{GABA}.

Super or subcritical

The rising and falling sweeps coincide (Figure 4): the amplitude grows continuously from zero at I^*_{ext} and retraces exactly on the way down, with no hysteresis (loop width 0 nA, branch gap 2×10^{-6}) and no cycle coexisting with the silent state. This is a supercritical Hopf, the recruitment cliff of exp025. The rising branch obeys the predicted $A^2 \propto (I_{\mathrm{ext}} - I^*_{\mathrm{ext}})$ (slope 1.1×10^{-4}, $R^2 = 0.999$), so $A \propto \sqrt{I_{\mathrm{ext}} - I^*_{\mathrm{ext}}}$ and $\mathrm{d}A/\,\mathrm{d}I_{\mathrm{ext}}$ diverges at threshold: most of the amplitude appears in a narrow band of drive above I^*_{ext}.

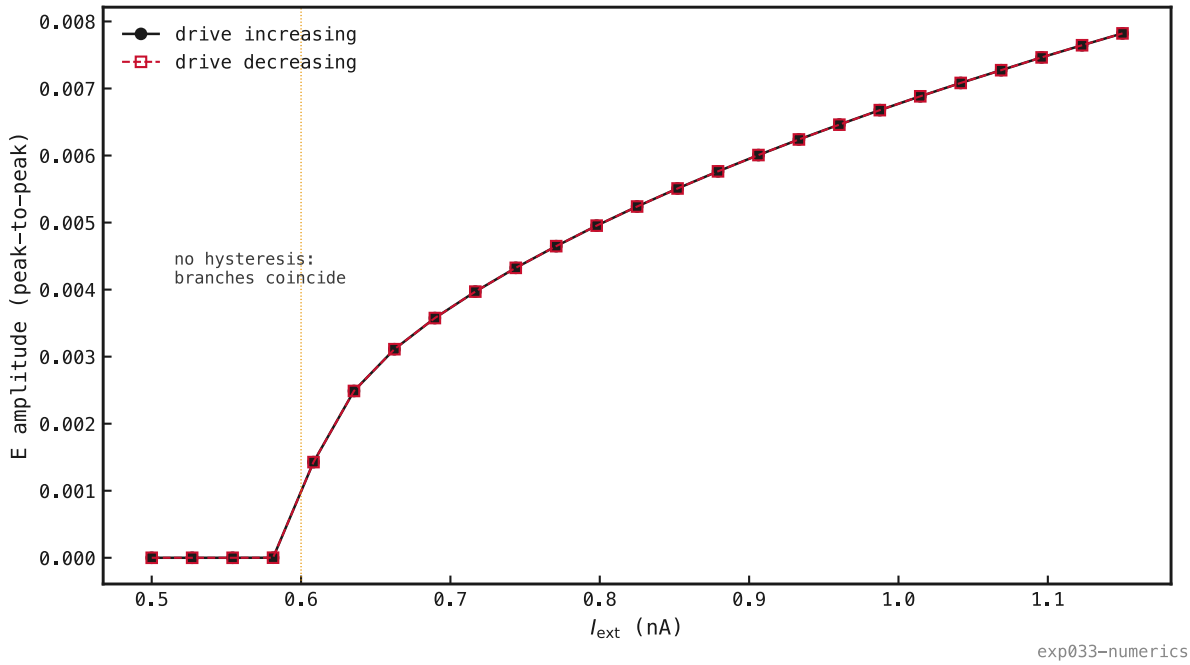


Figure 44: Quasi-static up/down ramp of the drive across I^*_{ext}. The rising branch (black, drive increasing) and falling branch (red, drive decreasing) coincide: the cycle turns on and off at the same drive, with amplitude growing continuously from zero. No loop, no bistable window: the reversible onset of a supercritical Hopf. (A subcritical onset would show the two branches separating into a hysteresis loop.)

Above onset, E leads I by ≈ 5.3 ms, the loop delay the 2D reduction omits.

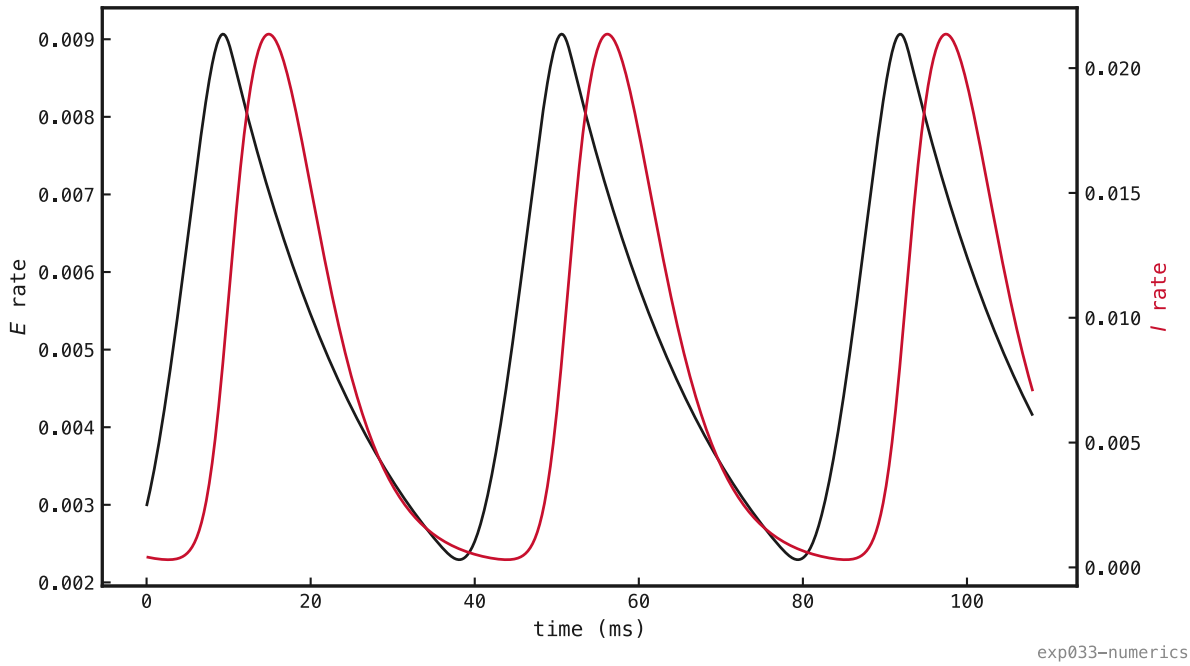


Figure 45: E (black) and I (red) over the limit cycle at $I_{\text{ext}} = I^* + 0.4$ nA. The E burst leads the I burst by ≈ 5.3 ms: E recruits I, I shunts E a few milliseconds later, and the cycle repeats, the loop delay the 2D reduction omits.

Appendix

Is it really 4D?

A Hopf gives a closed-loop trajectory, so the long-run motion is planar, so there should be a 2D description. This appendix collects the attempts. Verdict: a 2D *description* exists (a centre manifold), but the *vector field* does not reduce below three.

1. **A Hopf gives planar (two-dimensional) dynamics, so 4D looks suspect.** A limit cycle is a closed loop, traversed with two coordinates (amplitude and phase), so the motion lies in a plane. If it is 2D, a 2D model should exist; steps 3–5 hunt for it.

2. **The geometry confirms the motion is 2D (Figures 6–7).** The four variables cycle in loop order (E, then g_e^I, then I, then g_i^E, each lagging the last, Figure 6). Projected onto every variable pair (Figure 7), the cycle is one closed loop on a thin 2D sheet, a *centre manifold* (the loop is a 1D curve on it). Only (E, g_e^I) nearly collapses to a line (AMPA trails the E rate); the rest enclose real area, so no variable pair can stand in for the sheet.

3. **Route A: the textbook Wilson-Cowan model (slave the conductances).** The standard 2D tool is two rates with instantaneous coupling, the 4D model with infinitely fast synapses. Slave each conductance to its filter's steady value (15)–(16), $g_e^I = \tau_{\text{AMPA}} W^{EI} E$ and $g_i^E = \tau_{\text{GABA}} W^{IE} I$, and substitute into (26):

$$\tau_E \dot{E} = -E + \Phi_E(I_{\text{ext}} - \tau_{\text{GABA}} W^{IE} I), \quad (30)$$

$$\tau_I \dot{I} = -I + \Phi_I(\tau_{\text{AMPA}} W^{EI} E). \quad (31)$$

   Its divergence (the Jacobian trace),

$$\frac{\partial \dot{E}}{\partial E} + \frac{\partial \dot{I}}{\partial I} = -\frac{1}{\tau_E} - \frac{1}{\tau_I} < 0, \quad (32)$$

   is a negative constant, so Bendixson–Dulac forbids a periodic orbit: no Hopf, for any drive or coupling. It has dropped the round-trip delay (E excites I after τ_{AMPA}, I shunts E after τ_{GABA}): zero-lag inhibition damps but cannot overshoot. The reduction is valid only if synapses are fast relative to the rhythm, which they are not ($\tau_{\text{GABA}} \approx 6$ ms is the gamma period's order).

4. **Route B: quasi-steady-state the rates instead.** The dual move: slave the rates, $E = \Phi_E(I_{\text{ext}} - g_i^E)$ and $I = \Phi_I(g_e^I)$, into the conductance equations (27), giving a 2D system in (g_e^I, g_i^E):

$$\tau_{\text{AMPA}} \dot{g}_e^I = -g_e^I + \tau_{\text{AMPA}} W^{EI} \Phi_E(I_{\text{ext}} - g_i^E), \quad (33)$$

$$\tau_{\text{GABA}} \dot{g}_i^E = -g_i^E + \tau_{\text{GABA}} W^{IE} \Phi_I(g_e^I), \quad (34)$$

   with the same negative-constant divergence,

$$\frac{\partial \dot{g}_e^I}{\partial g_e^I} + \frac{\partial \dot{g}_i^E}{\partial g_i^E} = -\frac{1}{\tau_{\text{AMPA}}} - \frac{1}{\tau_{\text{GABA}}} < 0, \quad (35)$$

   so no cycle: it rings down (Figure 8). (These rates are the membrane variables, already reduced to an f-I rate.)

5. **Route C: lump into fast and slow timescales.** Slave the two fastest variables, the AMPA conductance ($\tau_{\text{AMPA}} = 2$ ms) and the I rate ($\tau_I = 5$ ms), keeping the two slowest $\{E, g_i^E\}$ ($\tau_{\text{GABA}} = 6$, $\tau_E = 20$ ms):

$$\tau_E \dot{E} = -E + \Phi_E(I_{\text{ext}} - g_i^E), \quad (36)$$

$$\tau_{\text{GABA}} \dot{g}_i^E = -g_i^E + \tau_{\text{GABA}} W^{IE} \Phi_I(\tau_{\text{AMPA}} W^{EI} E). \quad (37)$$

   Trace $-1/\tau_E - 1/\tau_{\text{GABA}} < 0$: no cycle. The split is forced anyway: the constants interleave, $\tau_{\text{AMPA}} = 2 < \tau_I = 5 < \tau_{\text{GABA}} = 6 < \tau_E = 20$ ms, so "fast" and "slow" each mix a conductance with a rate.

6. **All three fail for one structural reason.** The network is a pure ring: the single loop $E \to g_e^I \to I \to g_i^E \to E$, no recurrent E→E or I→I and no self-drive, so each variable's only diagonal Jacobian term is its own decay and every gain Φ' sits off-diagonal. Eliminate *any* two variables and the 2D trace is $-1/\tau_a - 1/\tau_b < 0$; Bendixson–Dulac then rules out a cycle. Routes A–C are three of the $\binom{4}{2} = 6$ ways to pick the kept pair; the runner sweeps all six and none crosses. A 2D model that *does* oscillate has added a destabiliser PING lacks: recurrent excitation or a cubic self-gain, i.e. a positive diagonal term. We do not add one: PING has no W^{EE} to supply it, and bolting one on would make a self-excitation oscillator (van der Pol / FitzHugh–Nagumo), not PING: a different mechanism for the gamma, not this network's.

7. **Three dimensions survive.** Slave only the fastest lag, the AMPA conductance $g_e^I = \tau_{\text{AMPA}} W^{EI} E$ (step 2′s near-degenerate pair), leaving a three-lag ring:

$$\tau_E \dot{E} = -E + \Phi_E(I_{\text{ext}} - g_i^E), \quad (38)$$

$$\tau_I \dot{I} = -I + \Phi_I(\tau_{\text{AMPA}} W^{EI} E), \quad (39)$$

$$\tau_{\text{GABA}} \dot{g}_i^E = -g_i^E + \tau_{\text{GABA}} W^{IE} I, \quad (40)$$

   This still Hopfs. Located like the 4D bifurcation (sweep I_{ext}, diagonalise the 3×3 Jacobian, find the complex-pair crossing), it gives $I_{\text{ext}}^* = 0.7$ nA and $f^* = 37$ Hz, both above the 4D values, since dropping the AMPA lag stiffens the loop. The same sweep finds no crossing for either 2D reduction. Figure 8: 4D and 3D sustain the rhythm; all three 2D reductions ring down.

8. **Resolution: the 2D description is the centre manifold, not a coordinate pair.** The 2D description that exists is the centre manifold, the plane of the two critical, oscillatory eigenvectors, tilted across all of (E, I, g_e^I, g_i^E): a plane in the *eigenbasis*, not any coordinate pair. So it is genuinely 2D yet unreachable by dropping two physical variables: a coordinate projection lands on the wrong plane, where the gains go off-diagonal and Bendixson–Dulac kills the cycle (steps 3–6). The *attractor* is a 1D loop on a 2D manifold; the *vector field* does not reduce below three, since a Hopf needs three first-order lags in the $E \to I \to E$ ring to build the destabilising phase. 4D is the natural model, 3D the floor, and the 2D centre manifold where the cycle lives, not a model in the original variables.

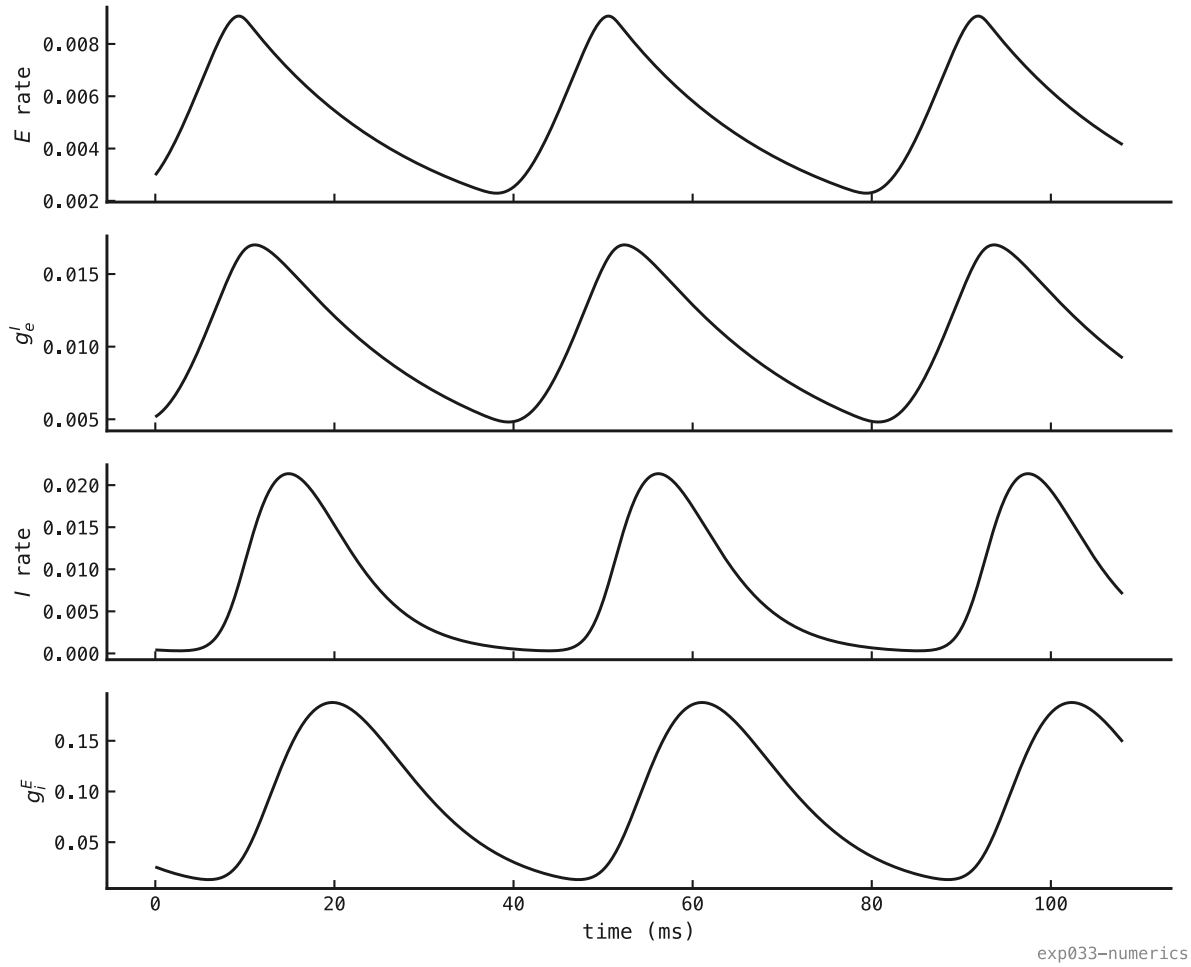


Figure 46: The four state variables over the limit cycle ($I_{\text{ext}} = I^* + 0.4$ nA), in loop order $E \to g_e^I \to I \to g_i^E$. Each peaks after the one above it: g_e^I tracks the E rate almost rigidly (the near-degenerate pair of step 2), then drives I, which fills g_i^E, which shunts E: one round trip of the ring per gamma cycle.

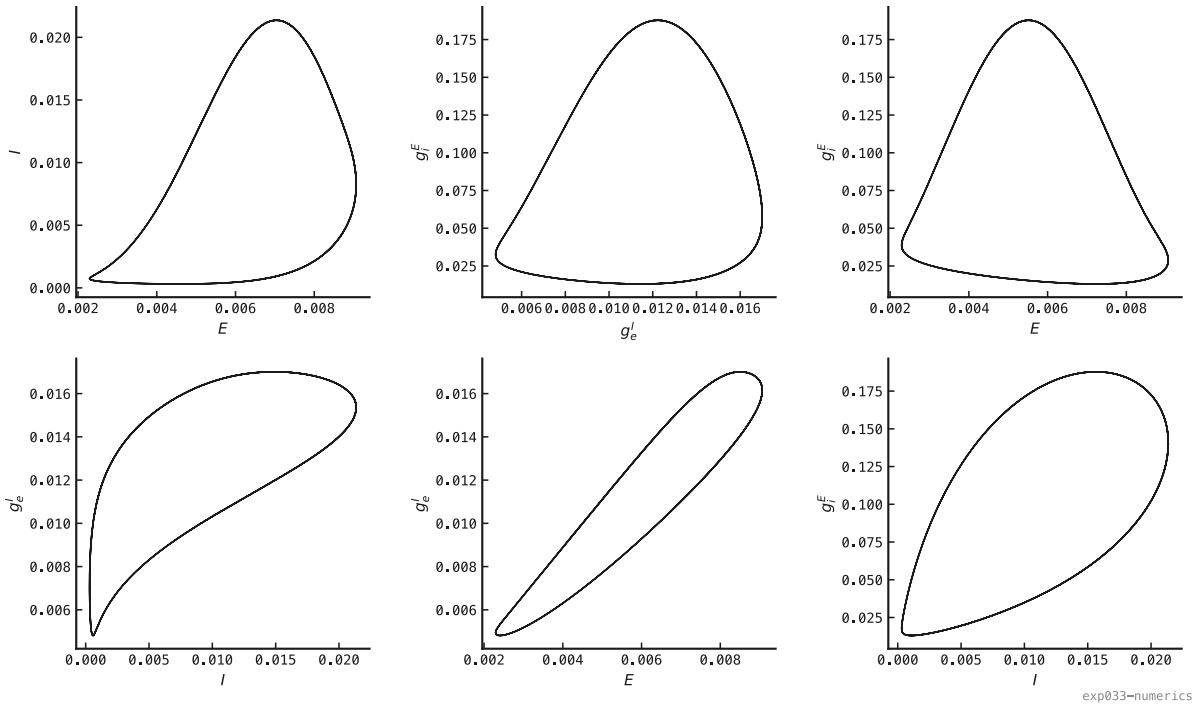


Figure 47: The 4D limit cycle ($I_{\text{ext}} = I^* + 0.4$ nA) projected onto all six variable pairs. Every projection is one closed loop: the trajectory lives on a 2D centre manifold. The (E, g_e^I) panel collapses almost to a line: the fast AMPA conductance tracks the E rate, which is why slaving it (the 3D reduction) preserves the dynamics, while the other pairs enclose genuine area and cannot be collapsed.

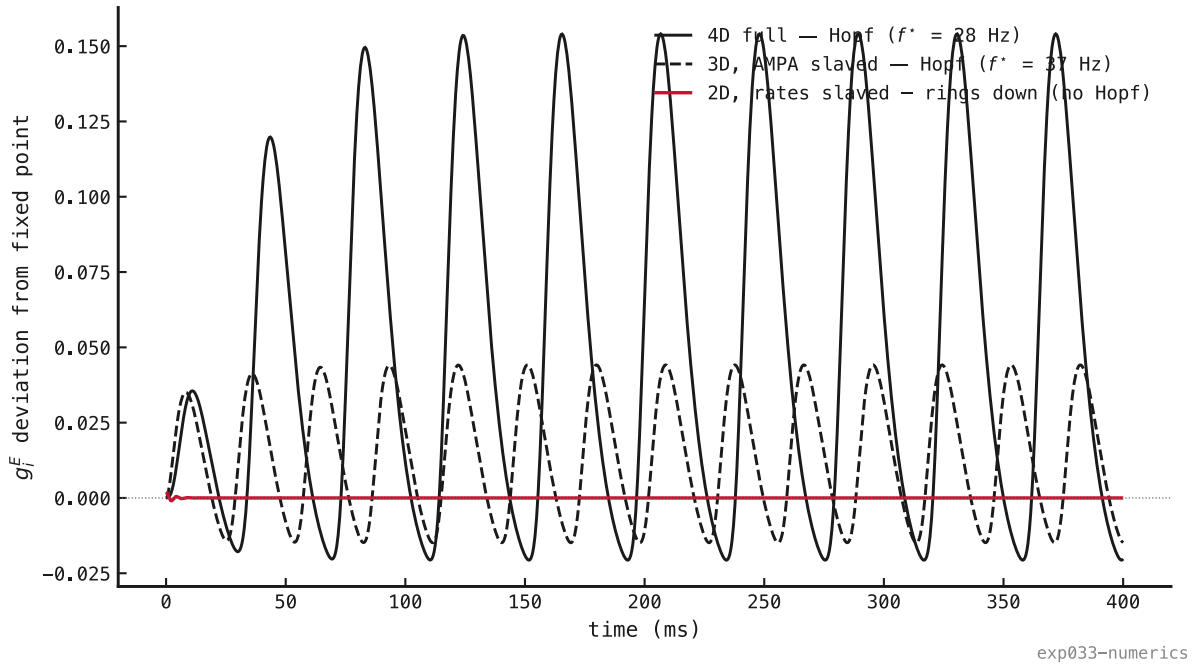


Figure 48: The reductions tested by simulation: g_i^E after a small kick at a common drive ($I_{\text{ext}} = 1$ nA, above every threshold). The 4D model (black, $f^* = 28$ Hz) and the 3D AMPA-slaved reduction (black dashed, $f^* = 37$ Hz) both sustain the limit cycle; the 2D rate-slaved reduction (red) rings down to its fixed point: no Hopf, as eq (35) requires.

PING tolerates 80% dropped spikes but collapses on added noise

30 May 2026

The trained networks this entry uses are produced once in the shared training hub, exp022 (Training), and reused here rather than retrained.

Abstract

Perturbs the hidden spike stream of trained PING and COBA networks at inference (drop spikes, add spikes) to ask whether the PING rate floor is dynamical or informational. PING tolerates dropping ≈ 80% of its emitted spikes (accuracy 85% at that level, from 91% unperturbed) yet collapses once added Poisson noise passes ≈ 81% of its own baseline rate. COBA is roughly flat to both, holding 89% across the whole add sweep (which reaches only ≈ 22% of its far higher baseline). The asymmetry, drops forgiven while adds break the gating, is the gamma cycle made visible: a structural feature of the architecture, not a readout-side trade-off.

Method

Parameter	Value
Integration timestep Δt	0.1 ms
Trial duration T	200 ms
Held-out MNIST test samples	500
Baseline training epochs (in exp022)	30

The PING and COBA baseline definitions and training recipe are in exp025, where the rate-floor mechanism is also worked out. This entry tests whether the floor is **dynamical** (locked by the cycle period) or **informational** (locked by the readout's spike-count requirement) by perturbing the hidden spike stream of the trained networks at inference.

The perturbation. A per-step callback on the COBANet's *_hidden_perturb_fn* slot fires every timestep of every trial, with no warm-up, schedule, or exclusion. Trials are $T = 200$ ms at $\Delta t = 0.1$ ms $\rightarrow$ 2000 fires per trial, applied across the held-out test set against the same trained network. At each timestep t:

1. *update conductances* from the previous step's spikes: $g_e^{(t)} \leftarrow g_e^{(t-1)} \cdot d_{\text{AMPA}} + \tilde{\boldsymbol{s}}^{E,(t-1)} W_{ee} + \text{input}^{(t)} W_{\text{in}}$, with the analogous update for $g_i^{(t)}$ from $\tilde{\boldsymbol{s}}^{I,(t-1)} W_{ie}$ and the E→I conductance from $\tilde{\boldsymbol{s}}^{E,(t-1)} W_{ei}$;
2. *LIF step* integrates $V^{(t)}$ and emits raw spike vectors $\boldsymbol{s}^{E,(t)} \in \{0,1\}^{B \times N_E}$, $\boldsymbol{s}^{I,(t)} \in \{0,1\}^{B \times N_I}$;
3. *perturbation callback* rewrites the raw vectors $\rightarrow \tilde{\boldsymbol{s}}^{E,(t)}, \tilde{\boldsymbol{s}}^{I,(t)}$;
4. *record* the perturbed vectors into the spike buffer;
5. *readout accumulator* adds $\tilde{\boldsymbol{s}}^{E,(t)} W_{\text{out}}$ to the mem-mean integrator.

Step 1 of timestep $t+1$ consumes the perturbed spikes through W_{ee}, W_{ei}, W_{ie}: a dropped E spike fails to drive I next step and contributes nothing to the readout; an injected I spike adds inhibition next step and counts in the rate metric. E and I get the same mode and level with independent draws.

Drop mode. For each (batch, neuron, timestep) slot i, draw $u_i \sim \text{Uniform}(0, 1)$ i.i.d. and keep each emitted spike with probability $1 - p_{\text{drop}}$:

$$\tilde{s}_i = s_i \cdot \mathbb{1}\left[u_i \geq p_{\text{drop}}\right].$$

Drop thins the count fed to both the readout and the next-step conductance update while leaving the E→I→E loop intact. Sweep $p_{\text{drop}} \in \{0.0, 0.1, ..., 1.0\}$, 11 levels.

Add mode. For each slot draw $u_i \sim \text{Uniform}(0, 1)$ i.i.d. and flip silent slots on at Poisson statistics, phase-independent:

$$\tilde{s}_i = \min(s_i + \mathbb{1}[u_i < r_{\text{add}} \cdot \Delta t/1000], 1).$$

Sweep $r_{\text{add}} \in \{0, 2, ..., 40\}$ Hz per neuron, 21 levels, applied equally to E and I. Because the two architectures sit at very different baselines (COBA ≈ 181 Hz per E cell versus PING ≈ 12 Hz, a factor of ≈ 15), a fixed added rate is a much larger relative insult to PING than to COBA, so the right panel of Figure 1 expresses the added rate as a percentage of each model's own baseline E rate (the architecture-fair view). The per-step RNG is seeded separately from the input encoder, so the Poisson input stream matches the unperturbed baseline. Total: 2 models × (11 drop + 21 add) = 64 forward passes.

Results

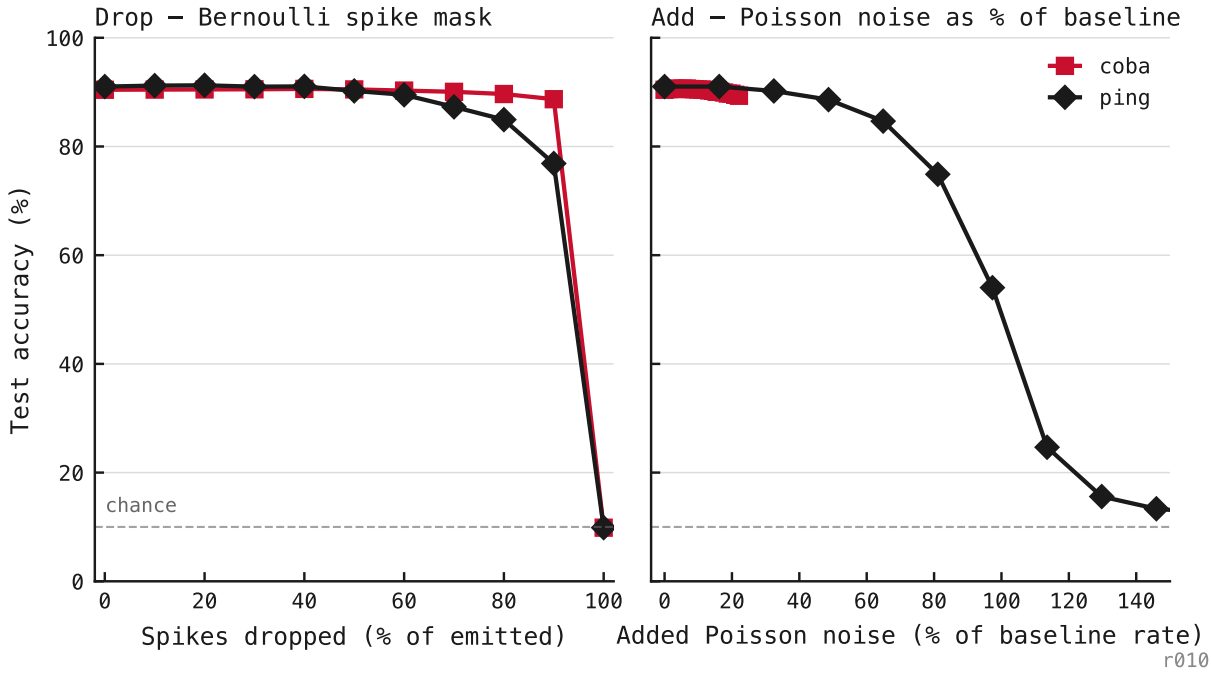


Figure 49: **Left (drop):** Bernoulli mask, percent of emitted spikes dropped. **Right (add):** Poisson injection as a percent of each model's own baseline E rate, so the two architectures are insult-matched. The asymmetry is direct: PING (black) holds accuracy to ≈ 85% at 80% drop but collapses once added noise passes ≈ 81% of its baseline, while COBA (red) stays at 89% across its whole sweep, which reaches only ≈ 22% of its far higher baseline. The dashed line marks chance.

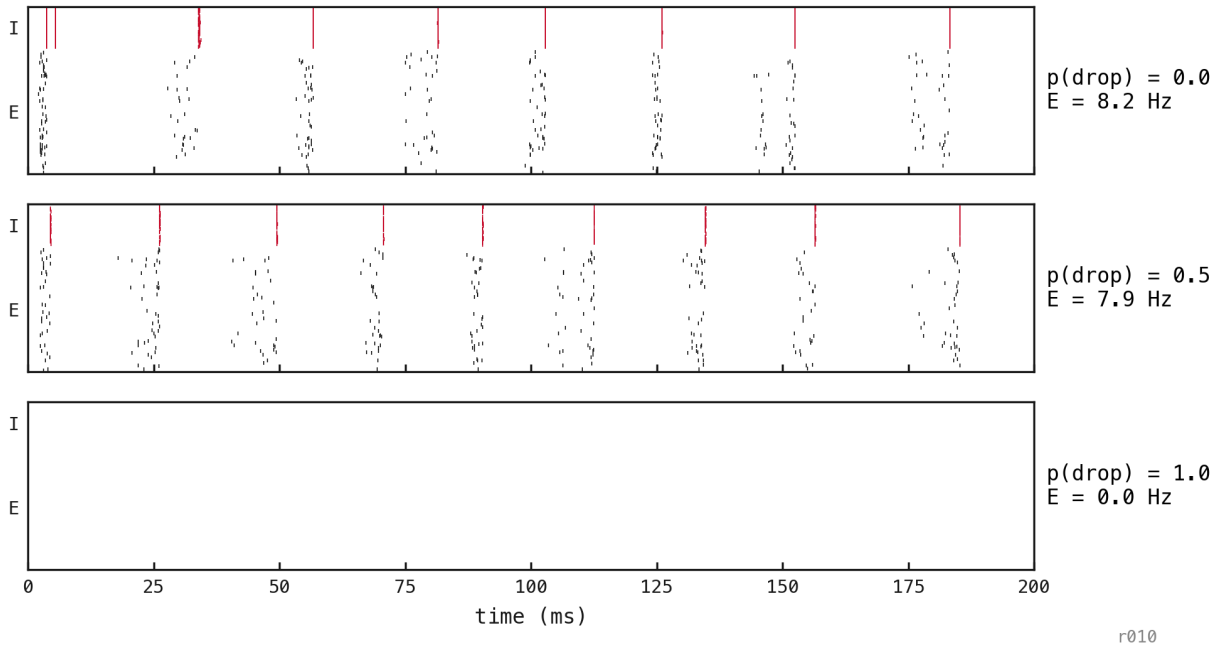


Figure 50: Trained PING replayed on the same MNIST digit 0 trial across three drop levels (0, 50, 100%); E (black) above I (red). The gamma cadence persists as spikes are thinned and only vanishes at total drop: dropping spikes thins the train without injecting phase-incoherent activity.

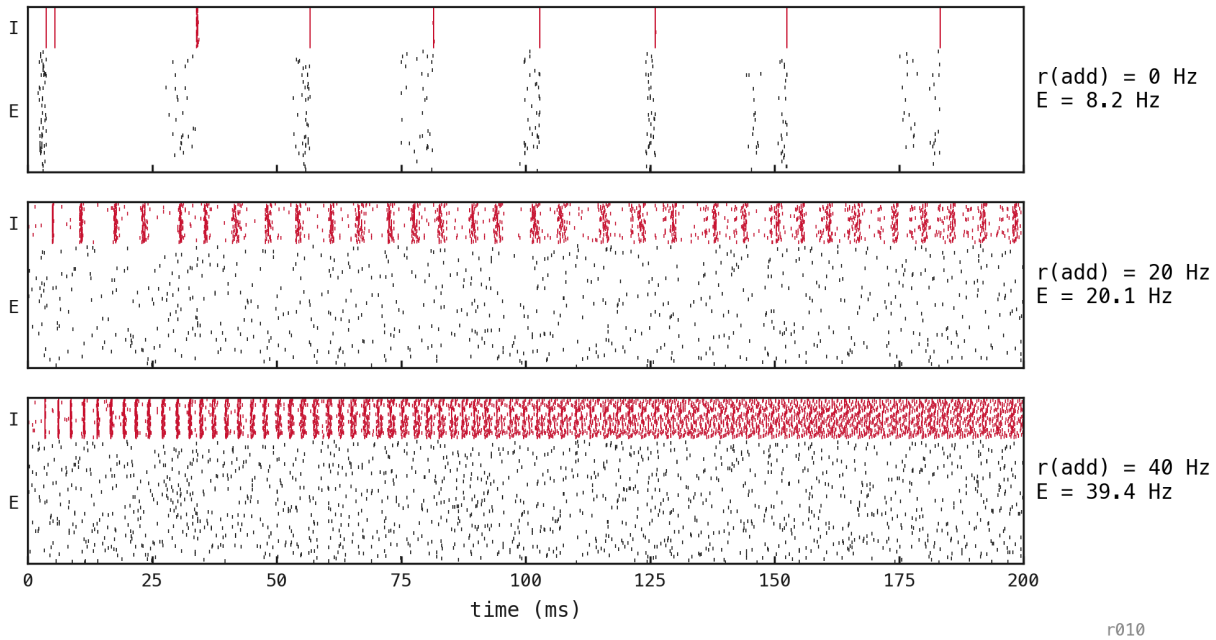


Figure 51: Trained PING replayed across three added-noise levels (0, 20, 40 Hz per neuron). The gamma banding dissolves into asynchronous firing as the added rate rises, matching the accuracy cliff in the previous figure.

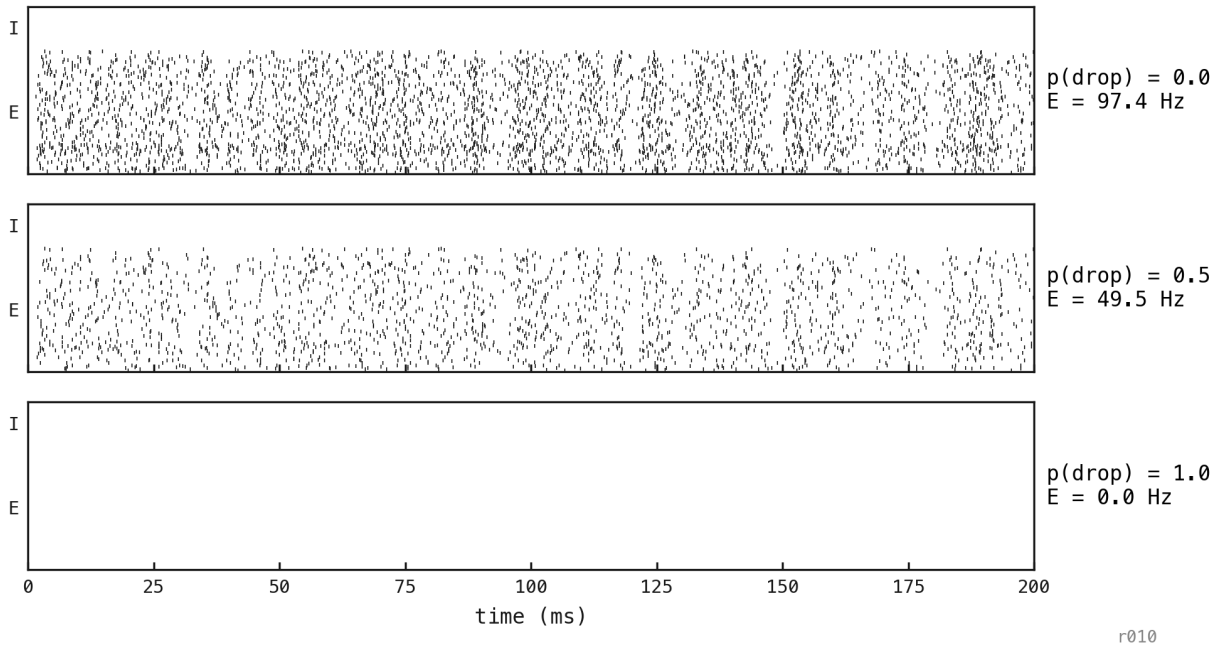


Figure 52: Trained COBA replayed across the same drop sweep. With no cycle to preserve, dropping spikes just thins a uniform asynchronous mean.

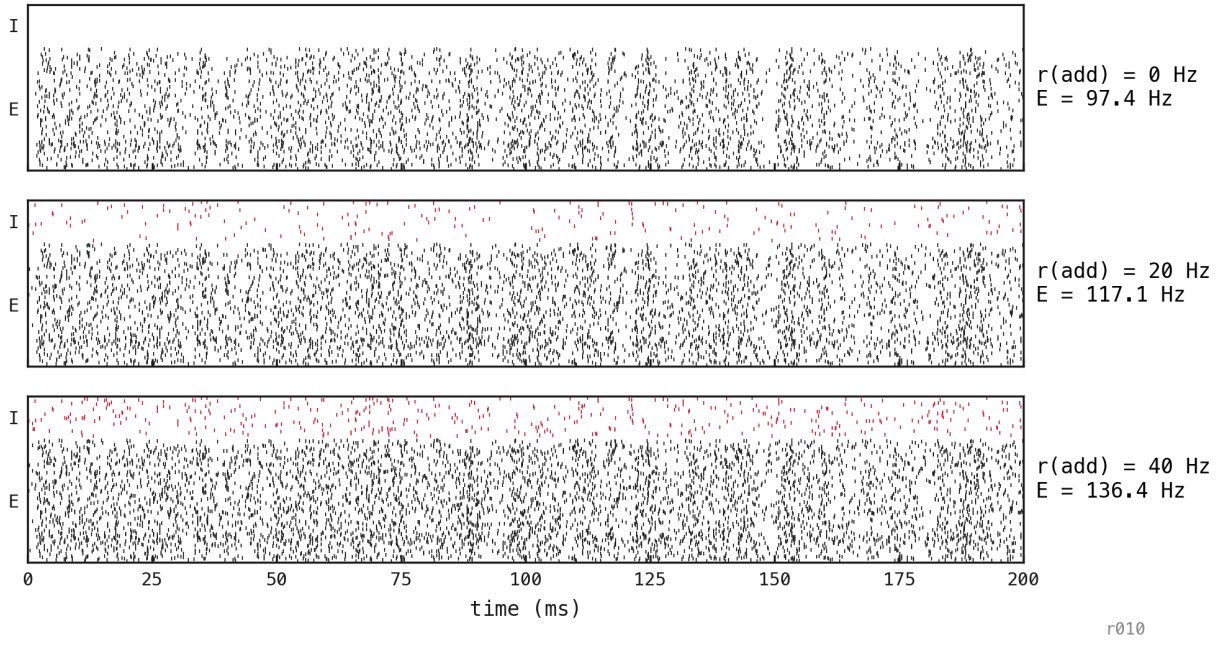


Figure 53: Trained COBA replayed across the same add sweep. Added spikes blend into COBA's asynchronous mean: no temporal structure to corrupt, just a higher mean rate.

Switching the loop on at inference cuts E rate ≈15×

30 May 2026

The trained networks this entry uses are produced once in the shared training hub, exp022 (Training), and reused here rather than retrained.

Abstract

Loads a trained COBA network and switches the I-loop on at inference by sweeping *ei_strength* from 0 to 1. The same feedforward weights that fire at ≈ 133 Hz without the loop fire at ≈ 9 Hz with it engaged, a ≈ 15× drop with no weight update. Accuracy, though, falls ≈ 36 pp (from ≈ 90% to ≈ 55%) as the loop engages, because the readout was never trained with it. PING gating is a post-hoc sparsity knob: the architecture, not the training, supplies the gamma dynamics, but using it well still needs training *with* the loop.

Method

Parameter	Value
Integration timestep Δt	0.1 ms
Trial duration T	200 ms
MNIST samples (80/20 stratified split of 7000)	5600 train / 1400 test (≈ 10% of the 70k-sample MNIST corpus)
Epochs	50

The PING and COBA baseline definitions and the training recipe are in exp025 this entry runs the COBA → PING I-loop transfer probe at eval time on the trained baselines. (The input-rate sweep / f–I curve material that previously lived here has moved to exp023, the natural home for "architectural response to drive".)

Inference-time probe. The trained COBA baseline (seed 42, θ_u = off) is loaded and *ei_strength* (the I-loop gain) is overridden at eval time across 11 values from 0 to 1. W_{in} and W_{out} load from the COBA checkpoint; W^{EI} and W^{IE} are freshly initialised (the COBA checkpoint stores these at zero, so skipping the load leaves a functional I-loop). No retraining.

Results

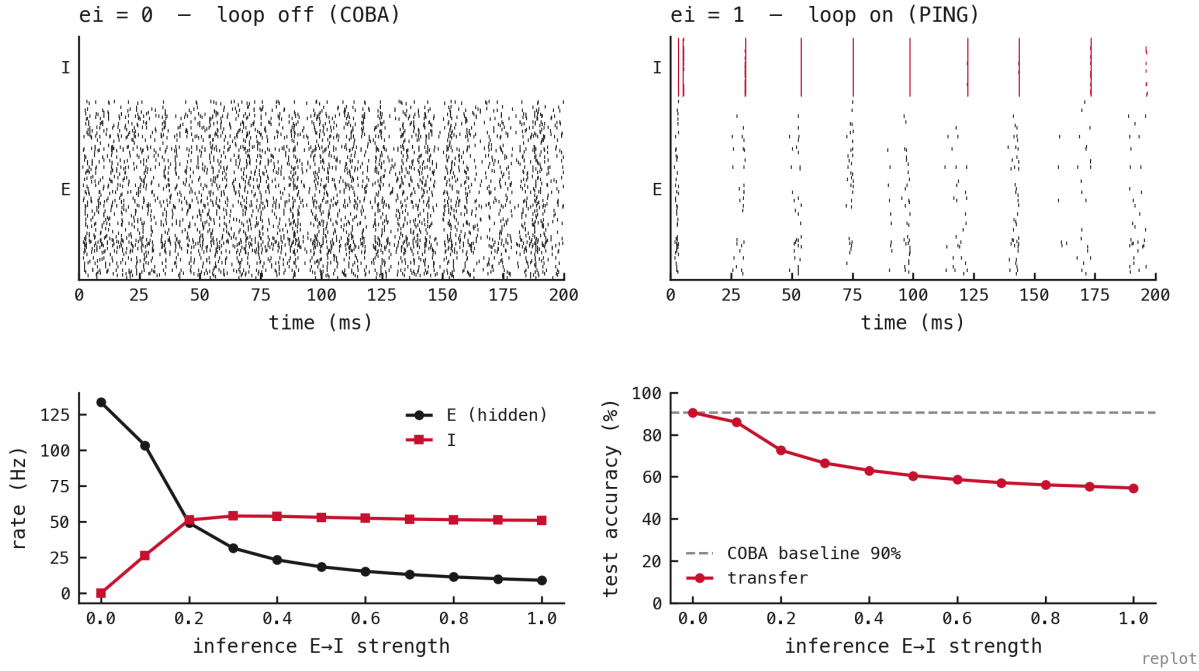


Figure 54: Switching the recurrent I-loop on *at inference* on a trained COBA network (no retraining; W^{EI}/W^{IE} freshly wired, since the COBA checkpoint stores them at zero). **Top**: the same feedforward weights fire densely and asynchronously at *ei = 0* (COBA) and in gamma bands at *ei = 1* (PING); the gamma dynamics come from the inhibitory architecture, not from training. **Bottom left**: E rate falls ≈ 15× (≈ 133 → 9 Hz) as the loop engages while I rises to ≈ 51 Hz; the suppression is continuous in loop strength. **Bottom right**: accuracy *degrades* without retraining, from the ≈ 90% COBA baseline to ≈ 55% at full strength (a ≈ 36 pp cost). So the architecture supplies the rate-gating for free, but using it well needs training *with* the loop (exp025): the sparsity is architectural, the accuracy is learned.

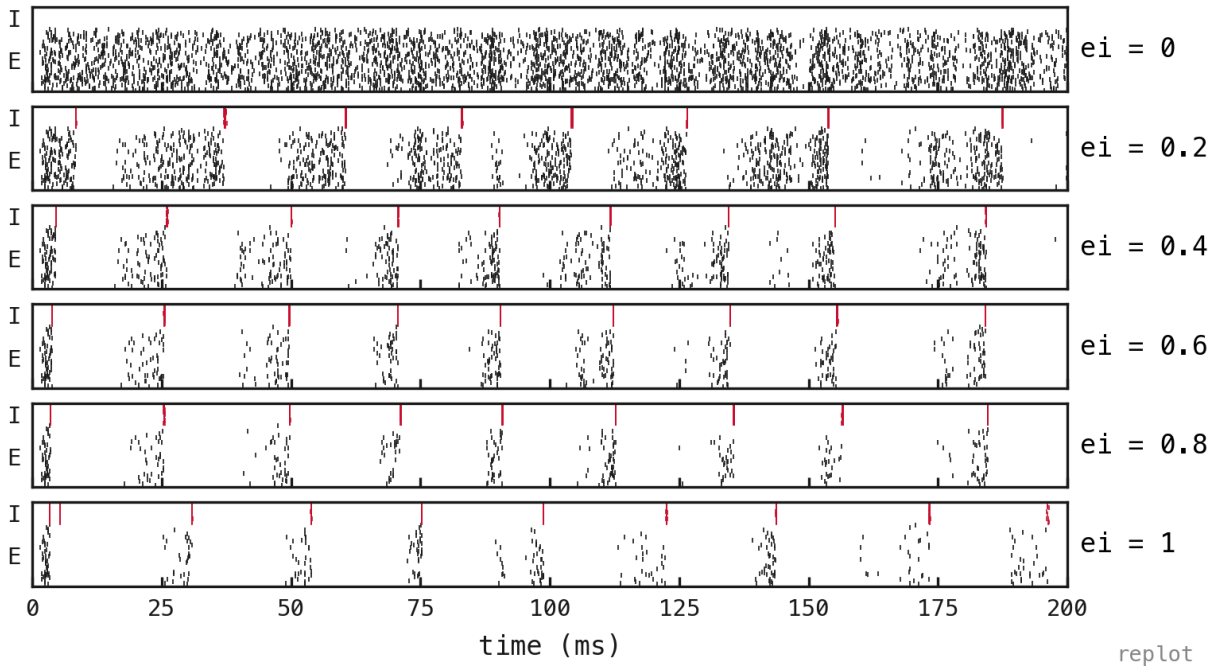


Figure 55: The full transition: trained COBA replayed at six inference-time *ei_strength* values (same trial, same feedforward weights, a fresh I-loop each row). At *ei = 0* the asynchronous-dense COBA pattern persists; by *ei ≈ 0.4* the same weights produce gamma cycles, sharpening toward *ei = 1*. The rhythm appears continuously as the loop is wired in, with no retraining at any point.

E rate is affine in gamma frequency

2 June 2026

The trained networks this entry uses are produced once in the shared training hub, exp022 (Training), and reused here rather than retrained.

Abstract

exp037 varied τ_{GABA} at inference and the per-cell E rate tracked the gamma cycle. Does that survive *re-training* at each τ_{GABA}? Yes. Across $\tau_{\mathrm{GABA}} \in \{4.5, 6, 9, 12, 18, 27\}$ ms $\times$ 3 seeds, $r_E = 1.15 + 0.183 \cdot f_\gamma$ with $R^2 = 0.994$. The slope is per-cycle E-cell participation; the intercept is a non-rhythmic baseline; accuracy stays at $\approx$ 86–92% across the sweep. The cycle clock constrains what the network can become.

Method

Sweep. Six τ_{GABA} values $\{4.5, 6, 9, 12, 18, 27\}$ ms $\times$ three seeds = 18 networks, trained in the shared hub to the gamma standard (50 epochs on MNIST, Adam at 4×10^{-4}, batch 256, mem-mean readout, no spike budget, $\Delta t = 0.1$ ms, $T = 200$ ms); only τ_{GABA} varies.

Measuring f_γ. For each cell, f_γ is the parabolic-interpolated peak of the Welch PSD on the per-trial population E trace; the fit (Figure 4) uses per-trial peak medians, which avoid the centroid bias of trial-mean PSD peaks. Parabolic interpolation with peak-bin values (y_0, y_1, y_2):

$$f_\gamma = \mathrm{freq}[\mathrm{peak}] + \frac{1}{2}\frac{y_0 - y_2}{y_0 - 2y_1 + y_2} \cdot \Delta f, \quad \Delta f = 5 \text{ Hz}.$$

It is needed because the bare 5 Hz bin quantisation would coarsen f_γ across the six conditions; on a well-isolated peak the interpolation error is $O((\Delta f)^3)$.

The predicted law. The shape $r_E = a + p \cdot f_\gamma$ is predicted by cycle dynamics, not curve-fitted. Within one cycle of duration $1/f_\gamma$, a fraction p of E cells emits exactly one spike (those nearest threshold when the I shunt drops); the rest are still recovering, so the cyclic per-cell rate is $p \cdot f_\gamma$. At long τ_{GABA} the I conductance never fully decays and the cycle dissolves into a tonic bath, leaving a feedforward baseline a independent of f_γ:

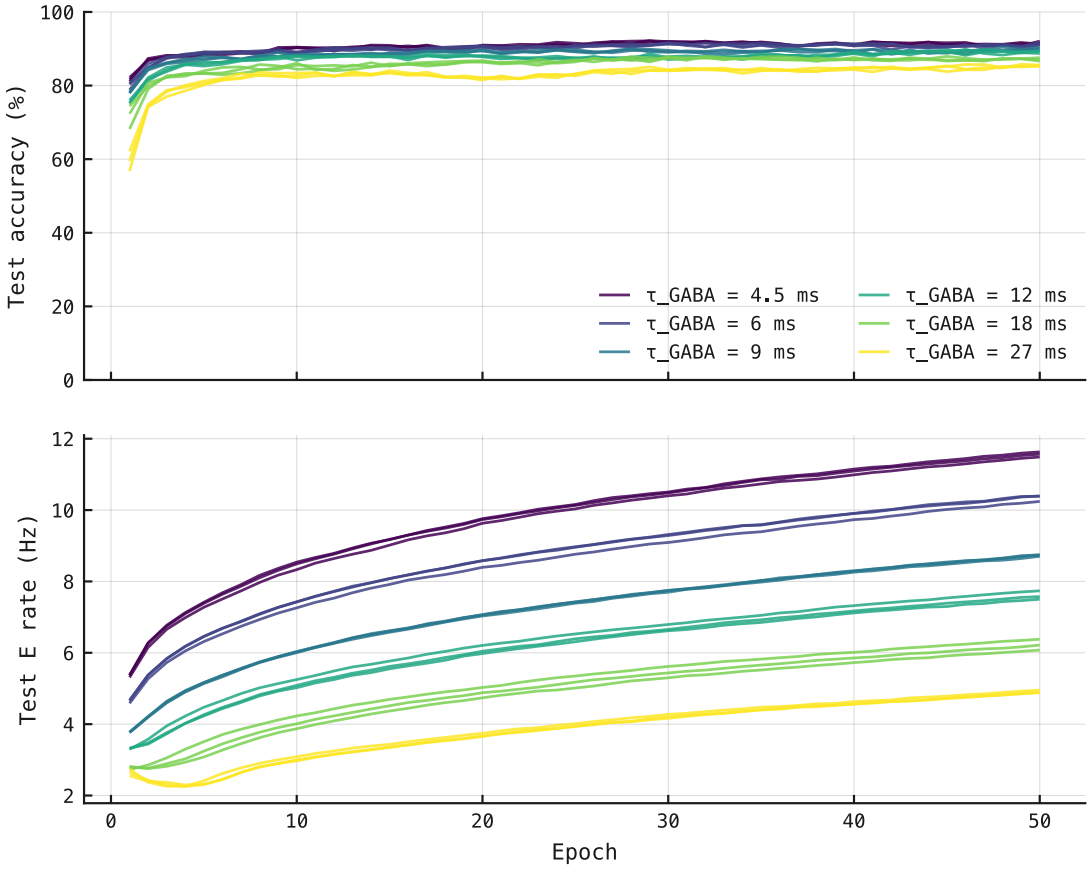
$$r_E = \underbrace{a}_{\text{feedforward baseline}} + \underbrace{p \cdot f_\gamma}_{\text{cyclic contribution}}.$$

We fit this across the 18 cells.

Convergence. Accuracy plateaus by $\approx$ epoch 15 while the E rate keeps climbing through training (Figure 1), so the fit uses the final-epoch rates. Fitting $r_E = a + p \cdot f_\gamma$ across the 18 cells is tight, and forcing the intercept through zero barely loosens it, so the law is not an artefact of the free intercept:

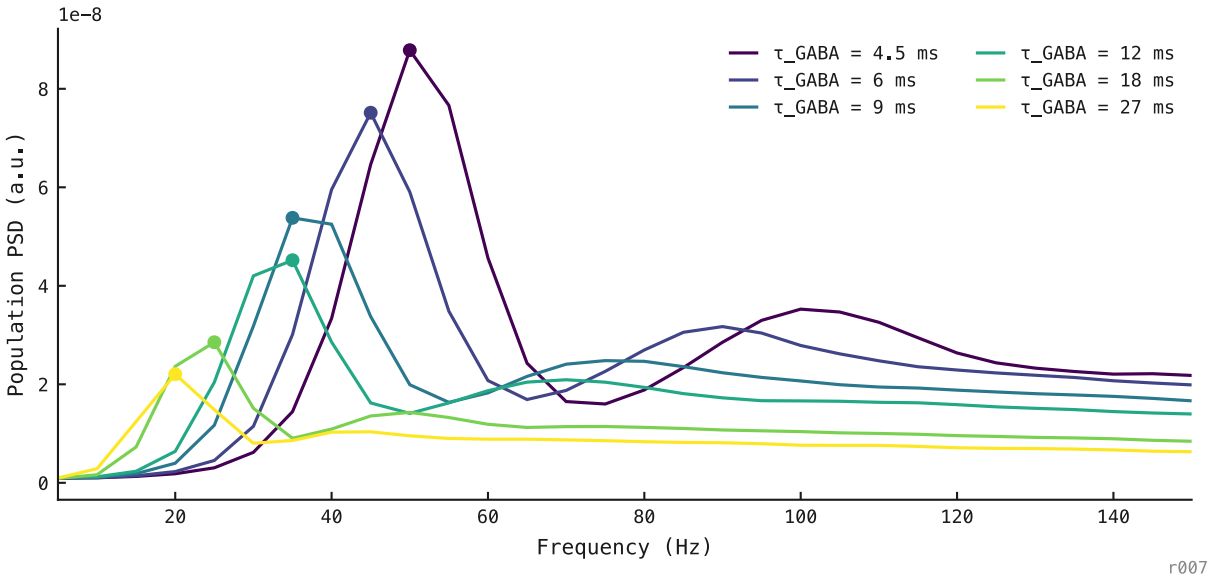
fit	a (Hz)	p (Hz/Hz)	R^2
affine	1.15	0.183	0.994
through origin	0	0.213	0.966

Results



r007

Figure 56: Per-cell accuracy (top) and E rate (bottom) over training, one line per cell. Accuracy plateaus by ≈ epoch 15; the E rate keeps climbing through the 50 epochs, so the final-epoch rates are the ones fit.



r007

Figure 57: Trial-mean Welch PSDs by τ_{GABA}; dots mark the parabolic-interpolated peak. The peak shifts cleanly from ≈ 20 Hz at $\tau_{\mathrm{GABA}} = 27$ ms to ≈ 51 Hz at $\tau_{\mathrm{GABA}} = 4.5$ ms, with no overlap between adjacent conditions.

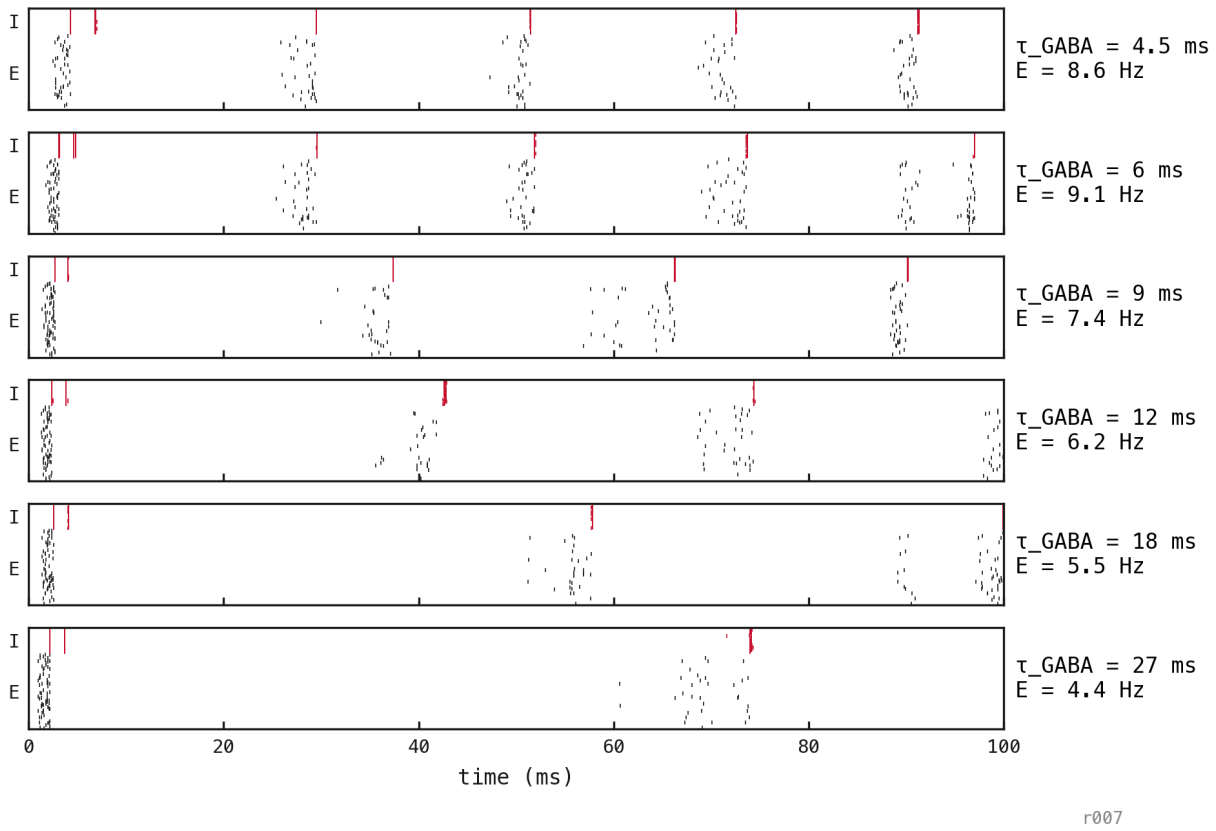


Figure 58: One MNIST trial through each network. The cycle period stretches from ≈ 20 ms ($f_\gamma \approx 51$ Hz) at short τ_{GABA} to ≈ 50 ms ($f_\gamma \approx 20$ Hz) at long, the eye and the spectrum agreeing.

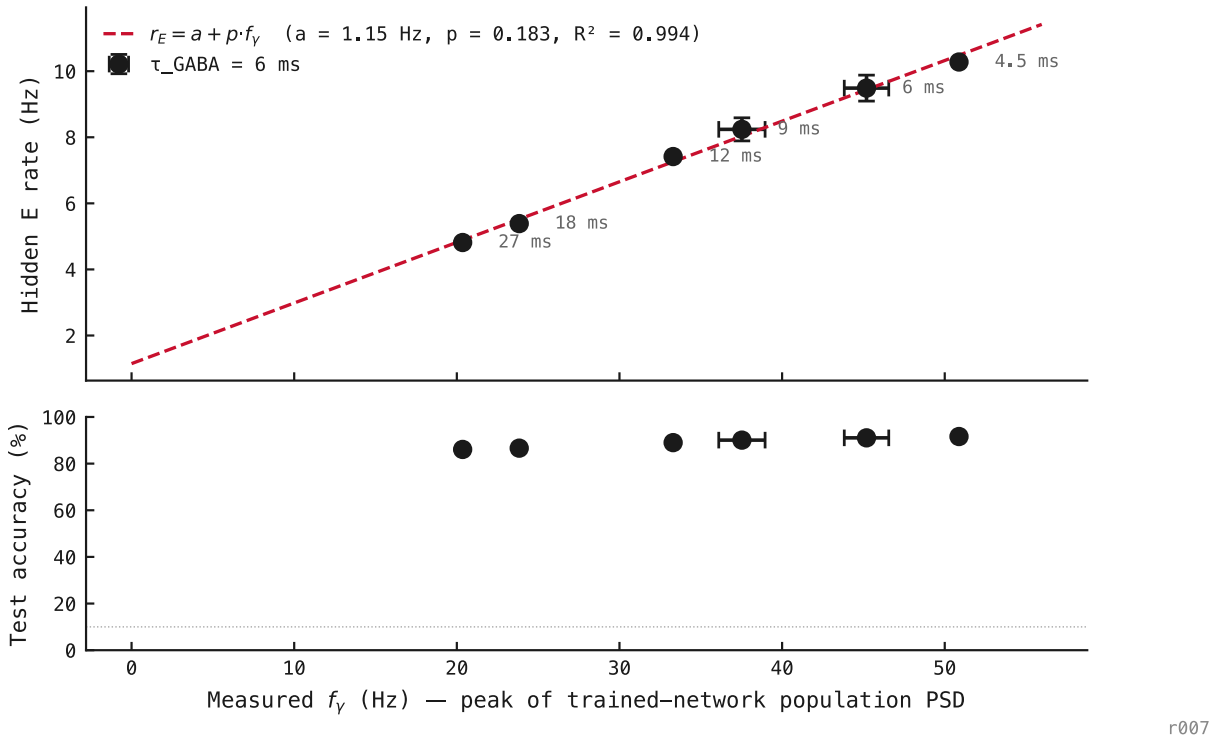


Figure 59: The law itself. Top: mean post-training E rate vs f_γ, six clusters × three seeds, error bars from seed variance; the affine fit passes through every error bar. Bottom: per-cluster accuracy is flat, so the rate change is not paid in classification.

Perturbations: gamma gates rates not just mean inhibition

2 June 2026

Abstract

exp025′s rate gap admits a cheap reading: PING fires less because the I-loop delivers more inhibition. This entry forecloses that reading and identifies *what* about the I-stream is doing the suppressing: qualitatively (rhythm vs mean), and quantitatively (which temporal precision is required). Scaffolded by ar009 §Leg 1 item 2.

Methods

Pure inference on the trained exp025 PING baseline (seed 42, θ_u = off). For each batch the I-population spike tensor $\boldsymbol{s}^I_{\text{base}} \in \{0,1\}^{T\times B\times N_I}$ is recorded from a baseline forward pass, then an override tensor replaces it in a second pass via the exp037 hidden-perturbation hook. The E-population experiences only the override I-stream through W^{IE}; the readout consumes the perturbed E spikes.

Every perturbation only *moves* spikes in time (or, for Poisson, redraws at the matched count) — none adds or removes them — so the mean per-cell I rate is matched to baseline by construction: exactly for phase-shuffle, and to within ≈ 3% for the jitter families across the range where each result is read. The one exception is cycle-coherent jitter at the largest σ: a Gaussian block offset with $\sigma = 100$ ms displaces part of each burst past the ends of the fixed presentation window, where it is clamped and lost, so the *realised* I rate falls to 40.5 Hz (24% below the 53.2 Hz baseline). Realised I is therefore plotted on every sweep, and the strict same-mean-inhibition comparison (the compound figure below) is anchored at $\sigma = 14$ ms, where realised I is still within 3% of baseline on both arms.

Five perturbation families:

1. **Baseline**: no override; trained PING dynamics.
2. **Cycle-coherent jitter**: partition the trial into blocks of length $1/f_\gamma$ (≈ 22.8 ms at the trained operating point from exp041). For each (trial, block), draw a single Gaussian offset $\Delta \sim \mathcal{N}(0,\sigma^2)$ and shift every I-spike in that block by Δ. Within-burst cross-cell synchrony is preserved exactly; only the *placement* of each burst is perturbed. Sweep $\sigma \in \{0,1,3,7,14,21,28,42,60,100\}$ ms.
3. **Per-I-cell jitter**: draw an independent Gaussian offset $\Delta_{b,n,k} \sim \mathcal{N}(0,\sigma^2)$ for *every* I-spike and shift each spike by its own offset. Destroys within-burst cross-cell synchrony while keeping the mean per-cell rate; the within-burst counterpart of cycle-coherent jitter. Sweep $\sigma \in \{0,0.5,1,2,5,9,14,21,50\}$ ms.
4. **Phase-shuffle**: per-trial permutation π_b of the time axis applied to all I-cells together: $\boldsymbol{s}^I_{\text{shuf}[t,b,n]} = \boldsymbol{s}^I_{\text{base}[\pi_b(t),b,n]}$. Preserves cross-cell co-firing within a timestep; destroys all phase structure.
5. **Rate-matched Poisson**: per-(trial, cell) Bernoulli with $p = \text{count}_{b,n}/T$. Destroys both temporal and cross-cell structure; tests the g_i variance limit.

Results

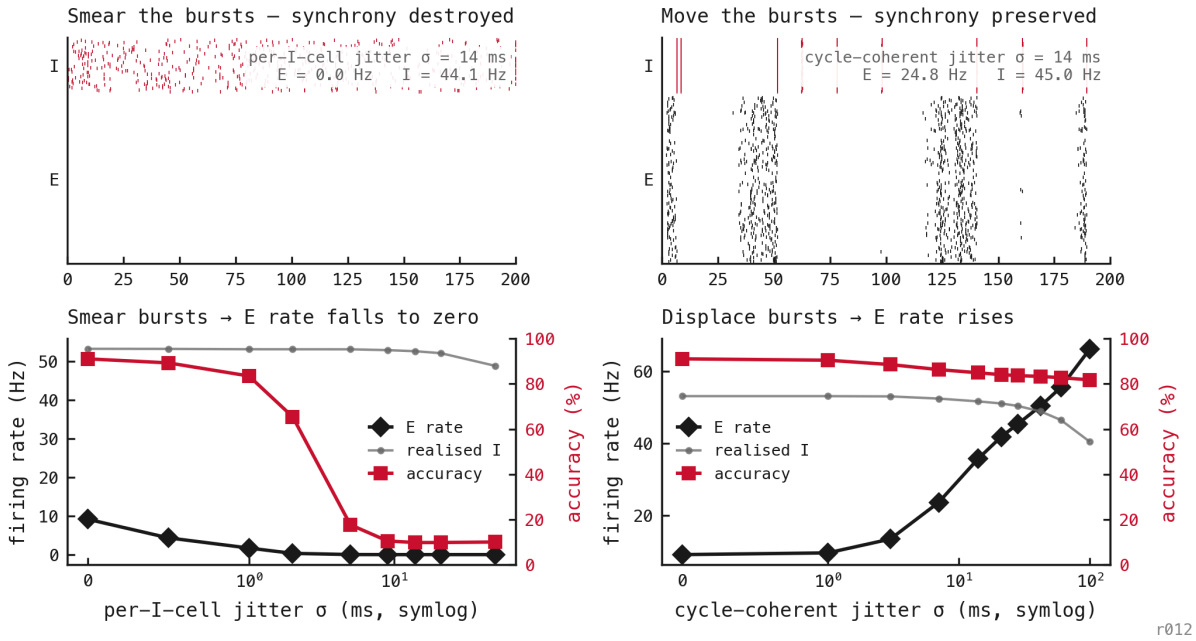


Figure 60: Two inference-time perturbations of the trained-PING I-stream, **both holding the mean per-cell I rate fixed at ≈ 53.2 Hz**, push the E rate in *opposite* directions, which a mean-inhibition account cannot produce. **Both columns use the same jitter magnitude, σ = 14 ms** — only the *kind* of jitter differs. **Left column, smear the bursts:** per-I-cell jitter (realised I 52.5 Hz) scatters the spikes within each burst, destroying synchrony while leaving the mean untouched; the burst dissolves into a continuous shunt, the E rate falls to zero, and accuracy collapses toward chance (9.9% at the Poisson limit). **Right column, move the bursts:** cycle-coherent jitter (realised I 51.7 Hz — within 3% of the left column) displaces each gamma burst bodily but keeps its within-burst synchrony; the I-stream opens gaps and the E rate *rises* from 9.1 Hz at baseline to 35.8 Hz, accuracy holding near 84.9%. Same jitter magnitude, same mean inhibition (both ≈ 53.2 Hz), opposite outcome: what gates the E rate is the *timing* of inhibition, the rhythm, not its average level. The cycle-coherent rise continues past the full phase-shuffle level to 66.3 Hz by σ = 100 ms (bottom-right sweep), but there the finite trial window truncates the most-displaced bursts and realised I falls 24%, so the strict rate-matched reading is taken at σ = 14 ms; see Methods.

Per-I-cell jitter

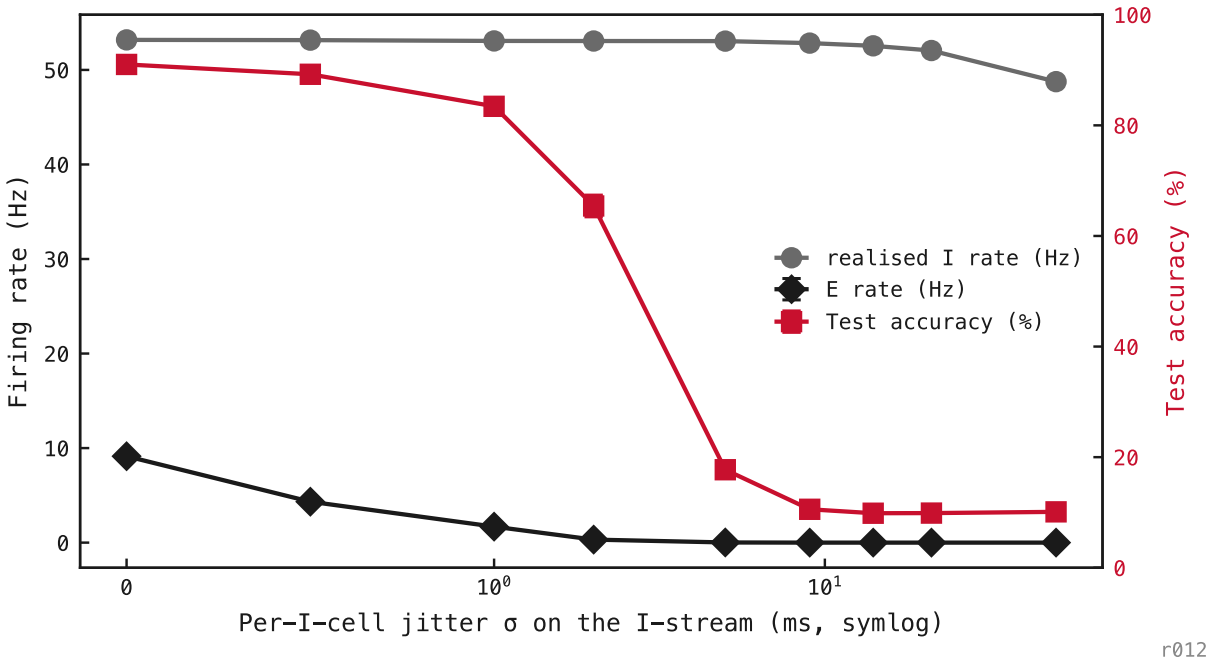


Figure 61: Per-I-cell jitter sweep, three seeds. Each spike receives an independent Gaussian offset; mean per-cell I rate is preserved exactly. **E rate (black diamonds, left axis) falls monotonically from baseline** (9.1 Hz), already more than halved to 4.3 Hz by $\sigma = 0.5$ ms and essentially zero (0 Hz) by $\sigma \approx 5$ ms, below $\tau_{\text{GABA}} = 6$ ms. **Accuracy (red squares, right axis) holds at ≈ 89.2% up to $\sigma = 0.5$ ms**, then collapses through 83.4% ($\sigma = 1$), 65.4% ($\sigma = 2$) and 17.7% ($\sigma = 5$), bottoming at chance (10.6%) by $\sigma = 9$ ms. The grey trace is the realised mean I rate, held flat near 53.2 Hz across the sweep: the E collapse happens under matched inhibition. The $\sigma \to \infty$ asymptote is the rate-matched Poisson regime: E silent, accuracy at chance.

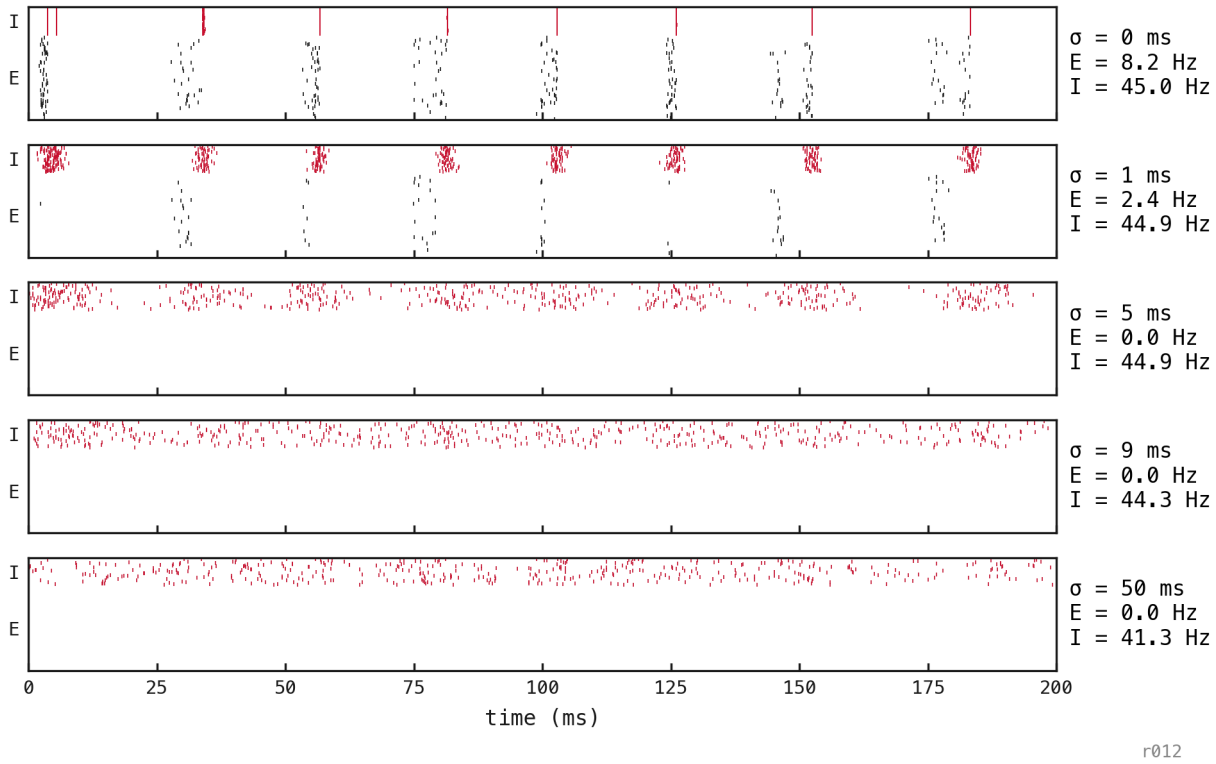


Figure 62: Single trial replayed at five per-cell jitter levels. At $\sigma = 0$ the I-bursts are crisp vertical bands. **At $\sigma = 1$ ms** the bursts visibly smear into a few-ms-wide cluster and E firing already collapses to the low single digits (per-trial E annotated on each panel; sweep mean 1.7 Hz). **At $\sigma \geq 5$ ms** the I-stream looks indistinguishable from a continuous low-variance shunt, and E is silenced. Per-cell jitter doesn't release E; it destroys the bursty structure that gave E its recovery troughs in the first place.

Cycle-coherent jitter

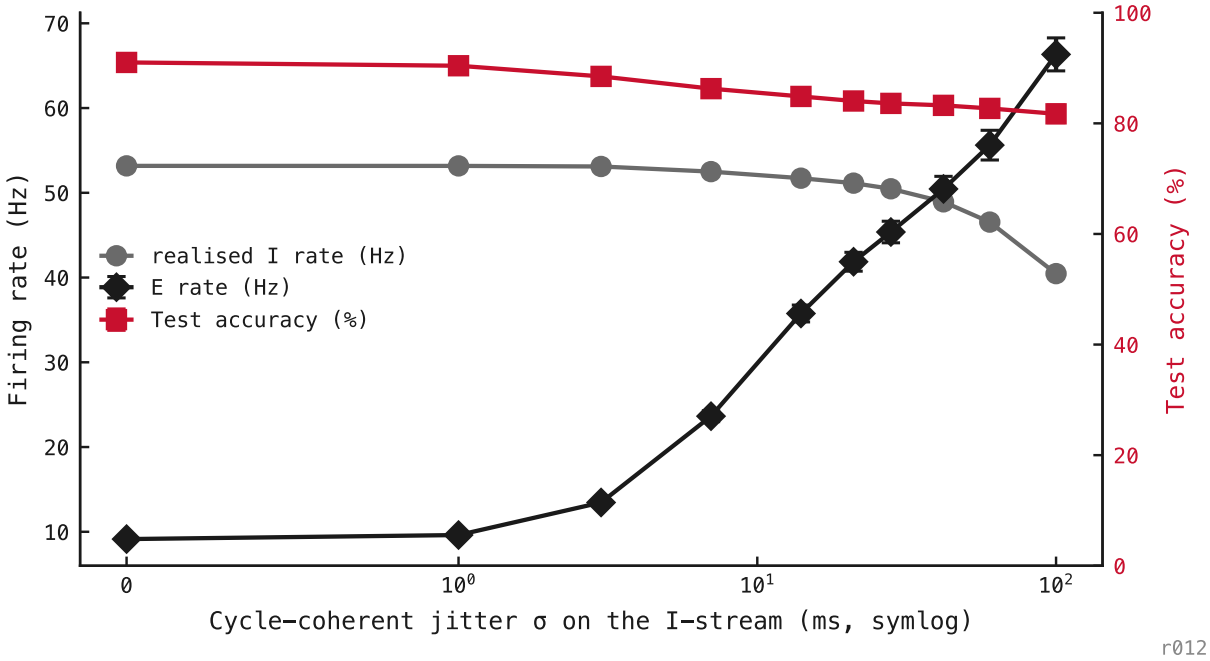


Figure 63: E rate (black diamonds) and accuracy (red squares) vs cycle-coherent jitter σ, three seeds. As σ grows the displaced bursts open wider gaps and the E rate climbs from baseline (9.1 Hz) *past* the full phase-shuffle level (25.9 Hz, the reference with within-burst structure destroyed) to 66.3 Hz by $\sigma = 100$ ms, with the sharpest rise near the predicted transition timescale $\sigma = 1/f_\gamma \approx 22.8$ ms; accuracy declines only gently, holding near 81.7%. The grey trace is the realised mean I rate: it holds within 3% of baseline through $\sigma \approx 14$ ms — where the E rate has already risen to 35.8 Hz — then droops to 40.5 Hz (24% below baseline) by $\sigma = 100$ ms, as the finite trial window truncates the most-displaced bursts. The strict same-mean-inhibition comparison is read at the smaller σ, where the rate is matched and the E rise is already unambiguous.

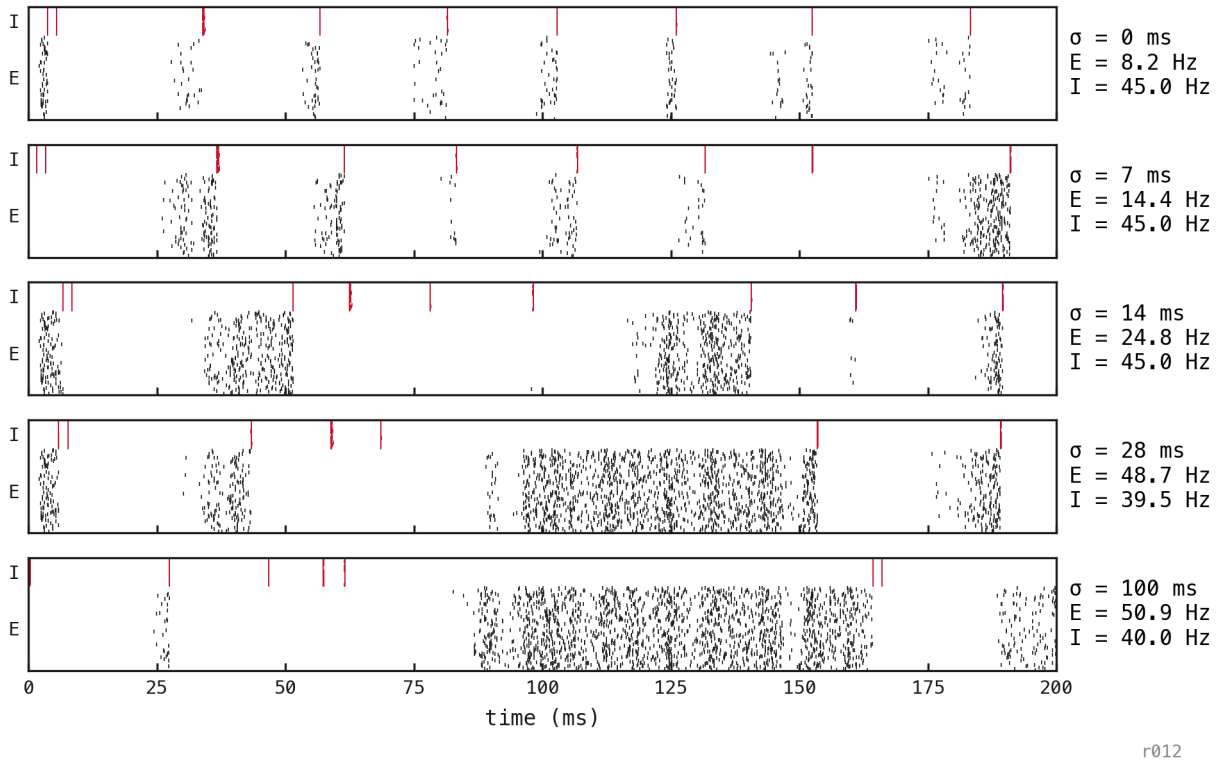


Figure 64: Single trial replayed at five jitter levels ($\sigma = 0, 7, 14, 28, 100$ ms; seed 42, MNIST digit 0 sample 0). Per-trial E rate annotated on each panel. **The I-bands stay vertical and crisp at every σ:** within-burst synchrony is preserved exactly. What changes is *where* each burst lands: at larger σ the bursts are displaced bodily from their phase-locked positions, opening longer gaps in the I-stream that E fires through, and the E rate climbs accordingly.

Next steps

Toroidal (wrapped) jitter, to extend the rate-matched range and disentangle release from truncation. The strict same-mean-inhibition claim is now anchored at $\sigma = 14$ ms, where realised I holds within 3% of baseline on both arms and the E rate has already risen to 35.8 Hz — the qualitative result stands on rate-matched ground. Beyond $\sigma \approx 30$ ms, though, the cycle-coherent E-rate rise and the realised-I droop become confounded: some of the extra E firing is genuine gap-opening, and some is simply less inhibition delivered, because the finite window clamps and loses the most-displaced bursts. Wrapping each block offset modulo the trial length would restore exact spike-count preservation at every σ and separate the two, at the cost of re-injecting a wrapped burst at the opposite trial edge — a phase artifact of its own, so the wrapped sweep is a robustness check, not a replacement. The prediction: if the wrapped E rate still climbs past the phase-shuffle level (25.9 Hz), the release at large σ is real; if it flattens there, part of the 66.3 Hz overshoot at $\sigma = 100$ ms was the truncation. Either way the anchored rhythm-vs-mean conclusion is unaffected.

Rate floor stable across a 20× Δt sweep

2 June 2026

The trained networks this entry uses are produced once in the shared training hub, exp022 (Training), and reused here rather than retrained.

Abstract

The Δt audit asks whether the exp025 headline E rate is a physical (Hz) property of the trained network or an artefact of the integration timestep. The rate stays within a 9–14 Hz band across a 20× Δt sweep, accuracy holds at 90.4–91.4%, and the gamma cycle period in physical ms is invariant. The rate is not fully Δt-independent, though: it rises monotonically with the timestep, from ≈ 9.2 Hz at Δt = 0.05 ms to ≈ 13.4 Hz at Δt = 1 ms, so a coarser step inflates the rate while leaving accuracy and cycle period intact.

Method

Parameter	Value
Integration timestep Δt	0.05–1.0 ms (swept)
Trial duration T	200 ms
MNIST samples (80/20 stratified split of 7000)	5600 train / 1400 test (≈ 10% of the 70k-sample MNIST corpus)
Epochs	50

The exp022 hub trains one PING per $\Delta t \in \{0.05, 0.1, 0.25, 0.5, 1.0\}$ ms × seed $\in \{42, 43, 44\}$ = 15 cells; this entry loads them and evaluates. Total physical time $T = 200$ ms is held constant (step count varies 4000 → 200). Batch size 64 throughout, smaller than exp025′s 256 but matched across the sweep so per-step compute and memory stay comparable, and the Δt = 0.05 cells (4000 timesteps × N_E × N_I) fit in a single A100. All other PING recipe parameters held to exp025.

Run inference on the test set; report mean E rate (Hz), accuracy, and a single-trial raster from seed 42 per Δt for visual cycle-period inspection.

Results

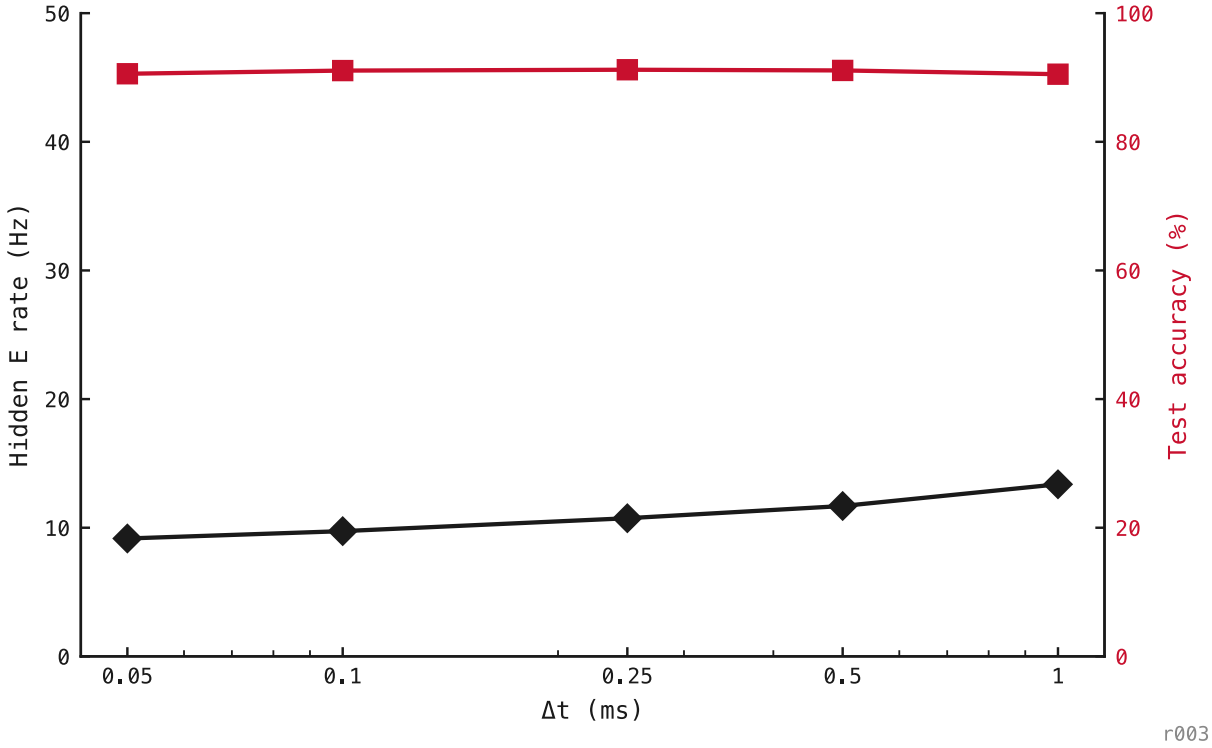


Figure 65: Hidden E rate (black) and test accuracy (red) as Δt varies 20×; markers are per-Δt means over three seeds. Accuracy holds near 91% while the E rate rises monotonically from ≈ 9.2 Hz at Δt = 0.05 ms to ≈ 13.4 Hz at Δt = 1 ms.

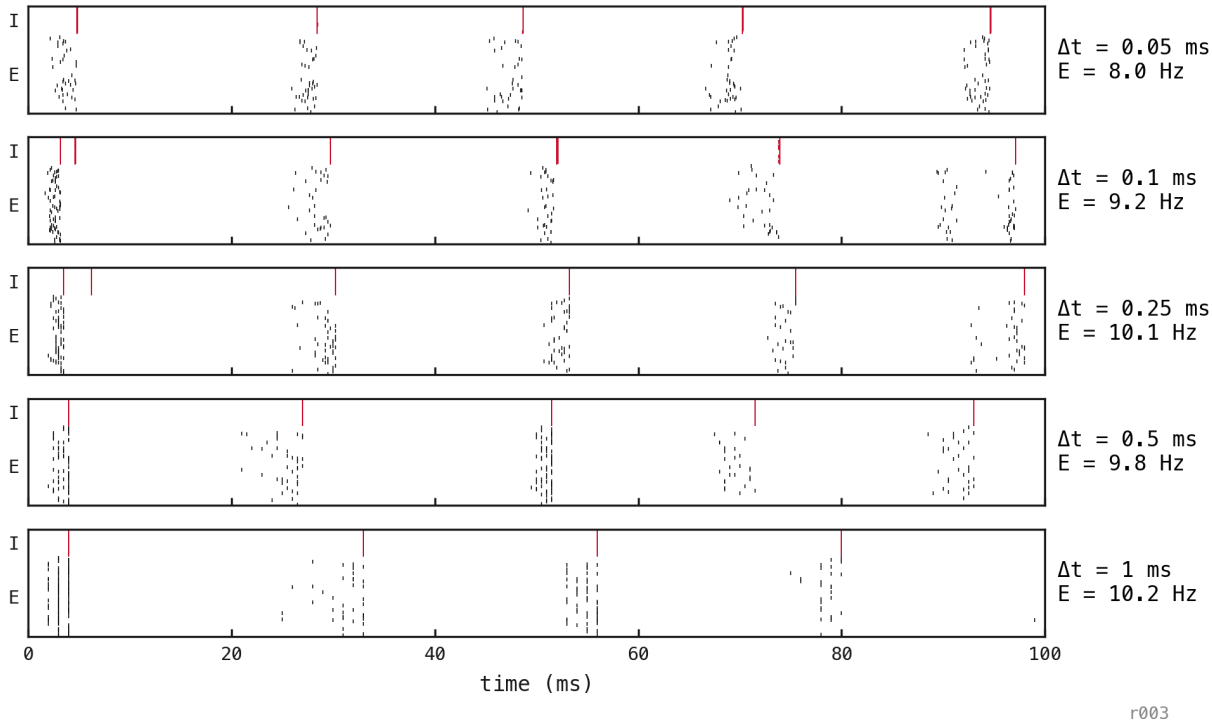
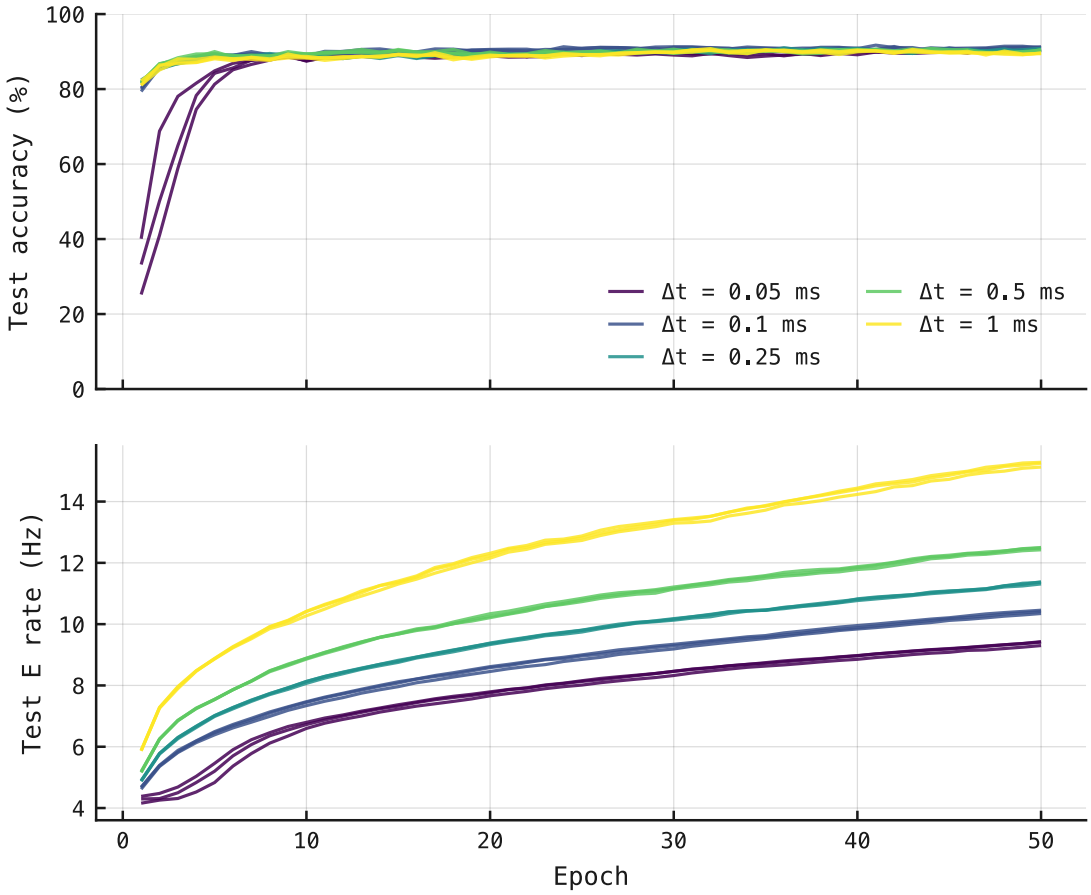


Figure 66: Single-trial rasters at each Δt, x-axis in *physical* ms (not steps). All five panels show E (black) and I (red) bursts locked to the same gamma cadence. The cycle physics is Δt-invariant.



r003

Figure 67: Top: test accuracy converges by epoch ≈ 10–15 across all Δt. Bottom: test E rate has *not* converged at epoch 50; every curve is still rising, ordered by Δt. The Δt-dependent rate is a snapshot at epoch 50, not a fixed-point ceiling.

Near-strict 1-spike-per-cycle ceiling: 98.9% of (cell, cycle) pairs

4 June 2026

Abstract

exp041′s slope $p \approx 0.20$ was interpreted as *"each E cell joins a cycle with ≈ 20% probability"*. That reading assumes the per-cycle spike count is bounded by 1 (an E cell either participates in this cycle or not). This notebook measures that directly: walking through every gamma cycle on the test set and counting how many spikes each E cell actually emits, on all 18 trained checkpoints in exp041′s τ_{GABA} sweep.

Methods

For each of exp041′s 18 trained cells (6 τ_{GABA} × 3 seeds):

1. Run inference on the MNIST test set; capture per-trial (T, B, N_E) and (T, B, N_I) spike tensors.
2. Detect I-burst times per trial: smooth the population I rate with a 1-ms Gaussian, run scipy peak detection with min-distance set to half the cell's own $1/f_\gamma$.
3. Cycle boundaries are the midpoints between consecutive I-burst peaks (first cycle starts at $t = 0$, last ends at trial end).
4. For each (cell, cycle, trial), count the number of E spikes within the cycle window.
5. Bucket counts globally into $\{0, 1, 2, \geq 3\}$ and aggregate by τ_{GABA}.

The cycle anchor is the I-burst: this is the right anchor because the cycle is operationally defined as *"the time between one inhibitory blanket and the next"*, not as the time between E bursts (which can be silent on a given cycle).

Results

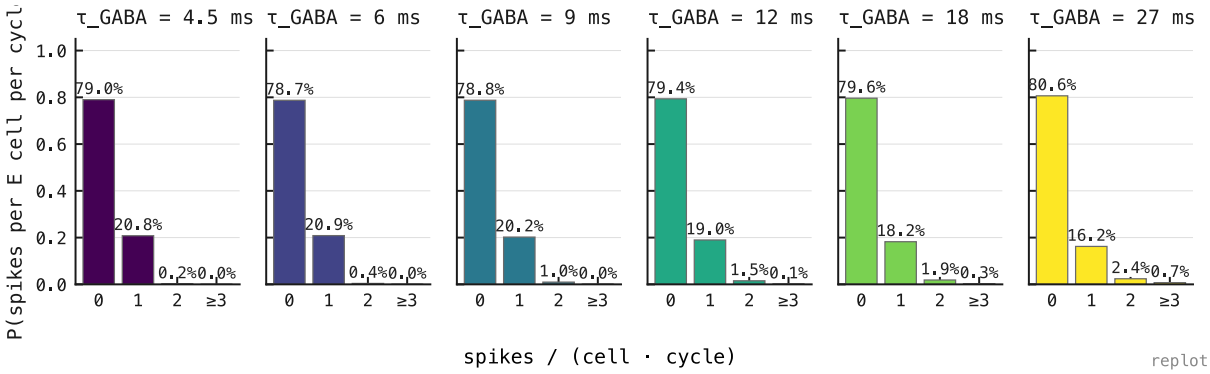


Figure 68: Distribution of E spike count per gamma cycle per cell, by τ_{GABA}, three seeds aggregated. Across **179 million (cell, cycle) pairs**, the architecture is overwhelmingly bimodal: each cell either emits zero spikes in a given cycle (≈ 79% of the time) or exactly one (≈ 20% of the time). Two-or-more events occur in ≈ 1.1% of cycles; three-or-more in ≈ 0.14%. Pooled over the sweep, 98.9% of pairs carry at most one spike.

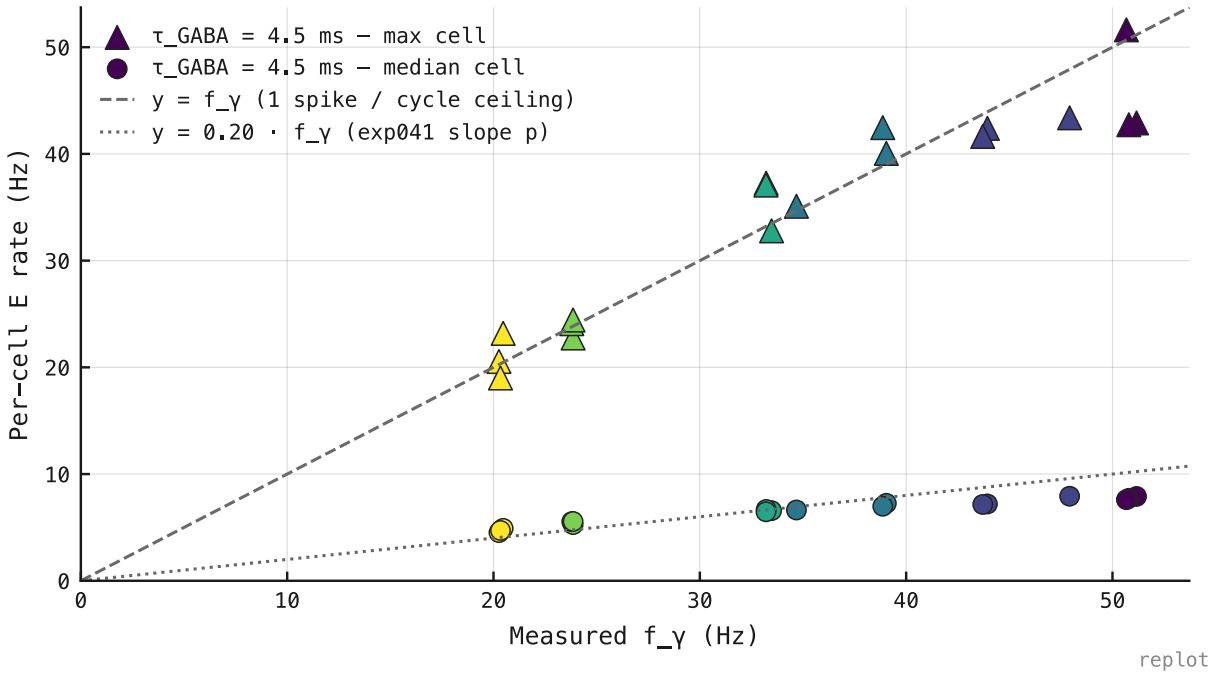


Figure 69: Per-cell E rate versus measured gamma frequency f_γ across the τ_{GABA} sweep. The busiest cell in each network tracks the one-spike-per-cycle ceiling $r = f_\gamma$ (max-cell fit $r = 0.97 f_\gamma$, $R^2 = 0.88$), while the median cell sits on exp041's shallower $r \approx 0.20 f_\gamma$ participation slope. The ceiling is near-strict: even the most active cell rarely exceeds one spike per cycle.

Pool-size invariance requires inverse synaptic scaling

14 July 2026

Abstract

The apparent invariance of pyramidal–interneuron network gamma (PING) firing rate to inhibitory-pool size is a consequence of the simulator's fan-in normalization, not independence from interneuron count. Excitatory cells are denoted E and inhibitory cells I. The nominal I→E parameter G_{IE} is divided by the presynaptic pool size, so the realised mean synaptic conductance is $j_{IE} = \frac{G_{IE}}{N_I}$. We vary N_I under paired controls. Rates remain flat when G_{IE} is fixed and $j_{IE} \propto \frac{1}{N_I}$; when the realised synapse j_{IE} is fixed, rates change strongly with N_I. Pool-size invariance therefore requires compensatory inverse scaling of individual synapses.

Methods

1. Weight convention. For a dense I→E matrix with shape $N_I \times N_E$, the simulator draws a non-negative weight with nominal mean G_{IE} and then divides every entry by its presynaptic fan-in N_I. Thus

$$\mathcal{E}\left[W_{kj}^{IE}\right] \approx j_{IE} = \frac{G_{IE}}{N_I}, \quad \mathcal{E}\left[\sum_{k=1}^{N_I} W_{kj}^{IE}\right] \approx G_{IE}. \quad (1)$$

G_{IE} is consequently an expected *summed coupling*, not the conductance of one synapse. An inhibitory volley with active set $\mathcal{A}$ gives E cell j the conductance increment

$$\Delta g_j^I = \sum_{k \in \mathcal{A}} W_{kj}^{IE}, \quad (2)$$

where:

- $\mathcal{E}$ denotes expectation over random weight initialization;
- W_{kj}^{IE} is the realised conductance from inhibitory cell k to excitatory cell j;
- j_{IE} is the mean realised conductance of one I→E synapse;
- G_{IE} is the expected sum of all I→E weights arriving at one E cell;
- N_I and N_E are the inhibitory and excitatory pool sizes;
- $\mathcal{A}$ is the set of inhibitory cells active in a volley; and
- Δg_j^I is the resulting inhibitory-conductance increment at excitatory cell j.

Equation 2 therefore depends on both the realised synapses and the number of participating inhibitory cells.

2. Paired controls. We sweep $N_I \in \{16, 64, 256\}$ under two conventions. In the fixed-summed-coupling arm, $G_{IE} \in \{1, 2, 4\}$ µS is held fixed, forcing $j_{IE} \propto \frac{1}{N_I}$. In the fixed-synapse arm, $j_{IE} \in \{3.91, 7.81, 15.63\}$ nS is held fixed and the nominal parameter is set to $G_{IE} = N_I j_{IE}$. The two arms coincide at the reference pool $N_I = 256$.

3. Simulation. Untrained dense PING network, $N_E = 1024$, $\Delta t = 0.1$ ms, $T = 500$ ms, 8 independent Poisson-input trials per network and 3 network/input seeds per condition. E→I summed coupling is fixed at 1 µS; input comprises 784 independent 25 Hz Poisson channels. Markers show seed means and error bars ±1 standard deviation (SD).

Results

Fixed summed coupling produces overlapping rate curves. At $G_{IE} = 2$ µS, the E rate is 6.01 Hz for $N_I = 16$ and 6 Hz for $N_I = 256$. This is the expected consequence of holding the summed I→E coupling fixed while shrinking each realised synapse as $\frac{1}{N_I}$.

Fixed realised synaptic strength gives the opposite result. At $j_{IE} = 7.81$ nS, increasing N_I from 16 to 256 changes the E rate from 16.62 to 6 Hz and the I rate from 120.41 to 34.41 Hz. Every tested synaptic strength shows the same pool-size dependence. The direction is mechanistically consistent: more inhibitory cells at unchanged individual strength produce greater population-level inhibition, while the recurrent feedback reduces both E activity and the I activity it recruits.

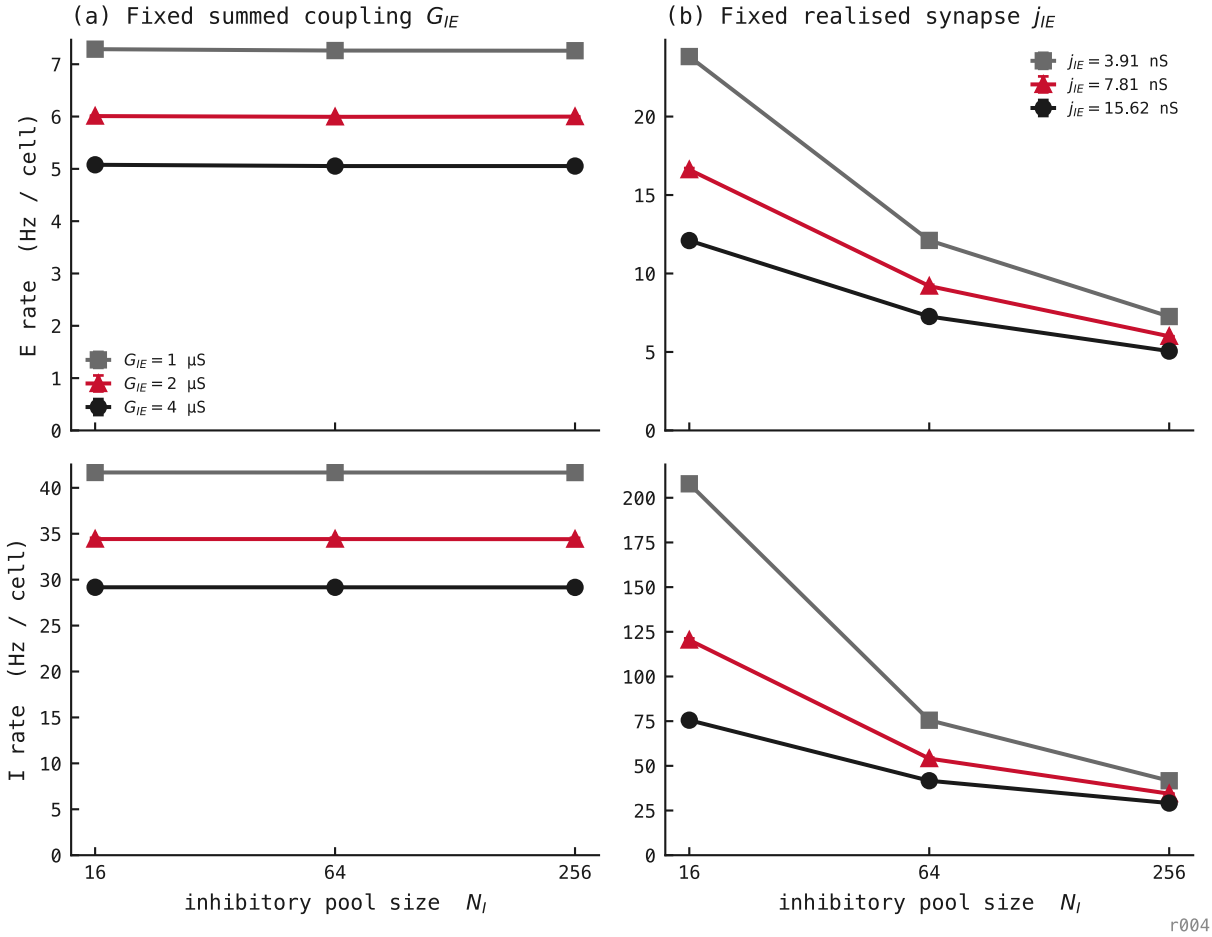


Figure 70: **Excitatory and inhibitory firing rates across inhibitory-pool size under two I→E scaling controls.** Top: excitatory-cell rate in Hz per cell. Bottom: inhibitory-cell rate in Hz per cell. **(a)** Holding the expected summed I→E coupling G_{IE} fixed makes the realised synapse $j_{IE} = \frac{G_{IE}}{N_I}$ shrink as the pool grows. **(b)** Holding j_{IE} fixed makes summed coupling grow with N_I. Each curve is one coupling level; markers are means and error bars ±1 standard deviation (SD) over 3 seeds, with 8 trials per seed. The controls coincide at the reference pool $N_I = 256$. Rates are invariant to pool size only under inverse scaling of individual synapses.

Conclusion

The network is not intrinsically insensitive to inhibitory-pool size. It is insensitive only along a specific normalization path: individual I→E synapses must scale approximately as $\frac{1}{N_I}$ so that expected summed coupling stays fixed. The operative control variable in this dense model is population-level inhibitory drive, jointly determined by pool size, realised synaptic strength, and volley participation. This experiment does not establish that the same inverse scaling holds biologically; it identifies the compensation required for invariance in the model.

Temporal and spatial evidence limits of trained PING

8 June 2026

Abstract

A frozen pyramidal-interneuron gamma (PING) network, whose trained weights remain fixed during evaluation, is tested under complementary temporal and spatial reductions of Modified National Institute of Standards and Technology (MNIST) digit evidence. It classifies continuously streamed digits without retraining, but a duration × input-rate sweep reveals a failure floor below 15 ms. At fixed 200 ms presentation and readout windows, performance remains at 10-class chance through 0.5 Hz and becomes clearly informative by 2 Hz. Separately, foreground pixels are permanently removed from binarized images and presented to both PING and a width-matched artificial neural network (ANN). PING is competitive at intermediate deletion but reaches chance by retention q = 0.02. Together, the curves delimit temporal, event-rate, and spatial evidence regimes for a future variable-rate training experiment.

Methods

Streaming duration and encoding rate

The trained baseline from the canonical PING experiment contains 1024 excitatory (E) and 256 inhibitory (I) cells. It was trained on one MNIST digit per 200 ms trial using random seeds 42, 43, 44. Each seed identifies an independent training run with a reproducible random initialization and data order. Everything here is **inference-only** at the trained timestep Δt = 0.1 ms; the weights are never updated. The two-dimensional sweep averages over all 3 seeds; the single-stream examples use seed 42.

A stream of digits, each shown for τ ms, is classified in one forward pass. The *only* change from training is a sliding readout window:

1. **Encode** each digit as a Poisson spike train over τ ms across 784 input channels, one per pixel. Each channel generates independent random spikes. The stated encoding rate is the expected number of spikes per second for a full-intensity pixel; lower pixel intensities reduce that rate proportionally.
2. **Concatenate** the per-digit trains into one input stream. At segment index k, the Poisson rate switches instantaneously at the boundary

$$t_k = k\tau. \quad (1)$$

   Here t_k is the boundary time of segment k, k is the segment index, and τ is the duration of each segment.
3. **Run once** through the trained network at the trained Δt, without retraining.
4. **Integrate evidence** in a non-spiking output leaky integrator, one unit per class. A leaky integrator accumulates incoming spikes while gradually discounting older input:

$$v_{\text{out}}(t) = \beta_{\text{out}} v_{\text{out}}(t-1) + \frac{1-\beta_{\text{out}}}{\Delta t} \boldsymbol{s}^E(t-1) W_{\text{out}}. \quad (2)$$

Here t is the discrete timestep; v_{out} (t) is the vector of output-unit states; β_{out} is their leak factor, the fraction of the previous output state retained for one timestep; Δt is the simulation timestep; s^E (t−1) is the E-cell spike vector at the preceding timestep; and W_{out} is the trained E-to-output weight matrix.

5. **Read a sliding window.** Average v_{out} over the trailing τ-window and apply a softmax:

$$\text{logits}(t) = \frac{\Delta t}{\tau} \sum_{u=t-w+1}^{t} v_{\text{out}}(u). \quad (3)$$

$$p(\text{class}, t) = \text{softmax}(\text{logits}(t)). \quad (4)$$

Here logits(t) is the class-evidence vector; u indexes timesteps in the window; τ is the presentation duration; w is its number of timesteps; and p(class,t) is the softmax-normalized class-probability vector. Softmax converts the logits into non-negative class probabilities that sum to one. The readout-window duration is matched exactly to the current digit's presentation duration:

$$T_{\text{readout}} = T_{\text{presentation}} = \tau. \quad (5)$$

Here $T_{readout}$ is the duration over which output evidence is averaged, $T_{presentation}$ is the time for which the digit is shown, and τ is that common duration.

The corresponding window length is

$$w = \frac{\tau}{\Delta t}. \quad (6)$$

Every digit is therefore read over exactly its own presentation duration; readout duration is not varied independently. At training the average ran over the *whole* trial; the trailing matched-duration window is the single change.

6. **Predict per segment** at the end of the digit's τ-window according to

$$\hat{c}(t) = \arg\max_{c} p(\text{class} = c, t). \quad (7)$$

Here c indexes the 10 digit classes, and arg max selects the class with the largest probability.

The output leak is

$$\beta_{\text{out}} = \exp(-\Delta t/\tau_{\text{out}}). \quad (8)$$

Here τ_{out} is the output-unit time constant, which controls how quickly accumulated output evidence decays, and exp denotes the exponential function. The probability trace p(class,t) is the network's online class confidence.

The grid uses presentation durations 10 ms, 15 ms, 25 ms, 40 ms, 50 ms, 75 ms, 100 ms, 200 ms and input rates 5 Hz, 10 Hz, 25 Hz, 50 Hz, 100 Hz, 200 Hz per channel. This gives 48 cells with 1200 classified segments per cell.

To resolve the encoding-rate floor below the grid, additional evaluations use rates 0.01 Hz, 0.025 Hz, 0.05 Hz, 0.1 Hz, 0.25 Hz, 0.5 Hz, 1 Hz, 2 Hz, 3 Hz while holding both presentation and readout at 200 ms. Each cell contains 10 streams of 10 digits for every trained seed. The

5 Hz, 10 Hz, 25 Hz, 50 Hz, 100 Hz, 200 Hz points use the same fixed-duration protocol and come from the corresponding grid row.

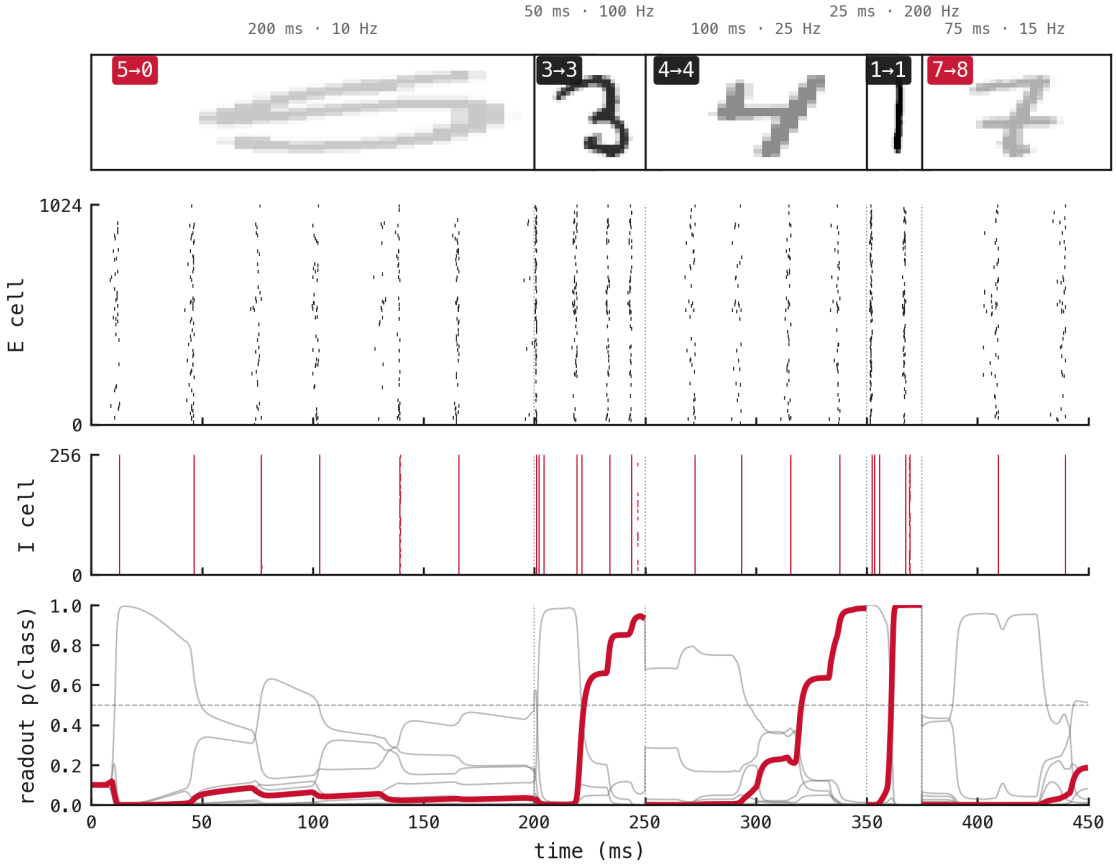
Foreground-retention calibration

The spatial protocol uses the same MNIST split and has two parts:

1. Train 3 seeds of a width-matched artificial neural network (ANN) with 784 inputs, one rectified-linear hidden layer of 1024 units, and 10 outputs. The hidden width matches the PING E population, not its recurrent E/I architecture. A rectified-linear unit outputs zero for a negative input and otherwise passes the input unchanged. Training uses 15 epochs, or complete passes through the training set, batches of 256 images per weight update, and learning rate 0.001, the step size of each update.
2. Binarize each held-out image, meaning an image excluded from training, at intensity 0: pixels above the threshold become unit-valued foreground and all others become zero-valued background. Retain every foreground pixel independently with probability q. Retention q = 1 leaves the foreground intact and q = 0 removes it. The ANN calibration uses 10 independent mask realizations per image. The matched comparison uses 100 fixed held-out examples and identical masks for every ANN and PING seed; PING encodes them at 25 Hz for 200 ms.

Results

Streaming classification and temporal evidence



exp048-replot

Figure 71: Classification when presentation duration and encoding rate vary between segments. The segment conditions are 200 ms at 10 Hz; 50 ms at 100 Hz; 100 ms at 25 Hz; 25 ms at 200 Hz; 75 ms at 15 Hz. Thumbnail opacity increases with encoding rate. The middle panels plot E- and I-cell spike rasters against time (ms); the lower panel plots class probability against time (ms), with the true class emphasized in red. The label-to-prediction pairs are 5→0, 3→3, 4→4, 1→1, 7→8, giving 3 of 5 correct segments.

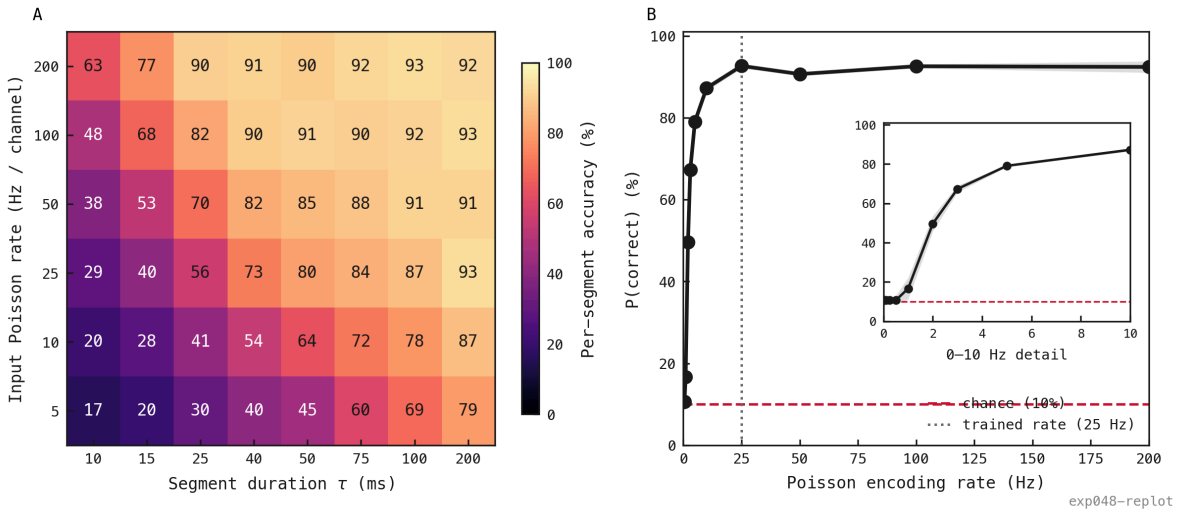


Figure 72: Temporal and encoding-rate limits of the frozen PING classifier. **(A)** Per-segment accuracy (%) is shown for presentation duration (ms, horizontal) and Poisson encoding rate (Hz per channel, vertical), using 1200 segments per cell. **(B)** Probability of a correct classification (%) is plotted against encoding rate (Hz) with presentation and readout fixed at 200 ms. The inset enlarges the linear 0.01–10 Hz interval without changing the axis scale. The dashed line marks 10-class chance and the dotted line the 25 Hz training rate. Accuracy stays at its empty-input floor, the accuracy obtained when almost no input spikes arrive, through 0.5 Hz, becomes informative by 2 Hz, and reaches 79.1% at 5 Hz.

The fixed-duration rate curve distinguishes a nonviable encoder regime from ordinary classification errors under weak evidence. In the variable-condition stream, the first failed segment received 10 Hz for 200 ms, yet that condition reaches 87.3% across the population. Its error is therefore natural trial-level variation, not evidence that 10 Hz is intrinsically too low. The other failed segment, presented at 15 Hz for 75 ms, is likewise above the empty-input rate floor, although its shorter window supplies less total evidence. Rates below 0.5 Hz are not useful operating points; 2 Hz is the lowest clearly informative tested rate and 5 Hz is a practical lower bound for future sweeps.

Spatial evidence calibration

The architecture-matched ANN remains above chance until foreground retention falls to q = 0.005, fewer than one visible foreground pixel per image on average.

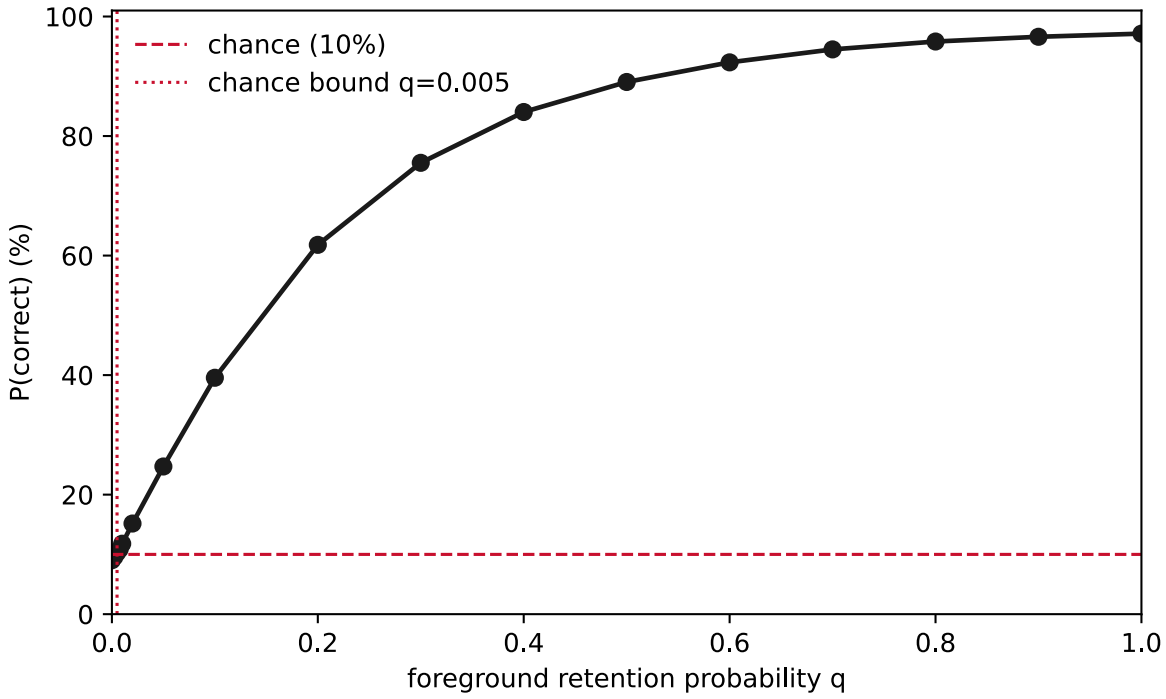


Figure 73: Held-out ANN accuracy as foreground evidence is removed. Probability of a correct classification (%) is plotted against foreground retention q, the independent probability that a foreground pixel remains visible. Points are means across 3 ANN seeds and the band is one standard error, the estimated uncertainty of that mean across seeds. The dashed line marks 10-class chance; the dotted line marks the measured chance-region bound at q = 0.005, the highest tested retention whose 95% confidence interval across seeds still contains chance accuracy.

Under identical masks, neither classifier is uniformly better. With 29.5 visible pixels on average (q = 0.2), PING reaches 68.7% against the ANN's 60.7%. They coincide near 45% at q = 0.1. PING still leads at q = 0.05, then reaches chance at q = 0.02 while the ANN remains above it.

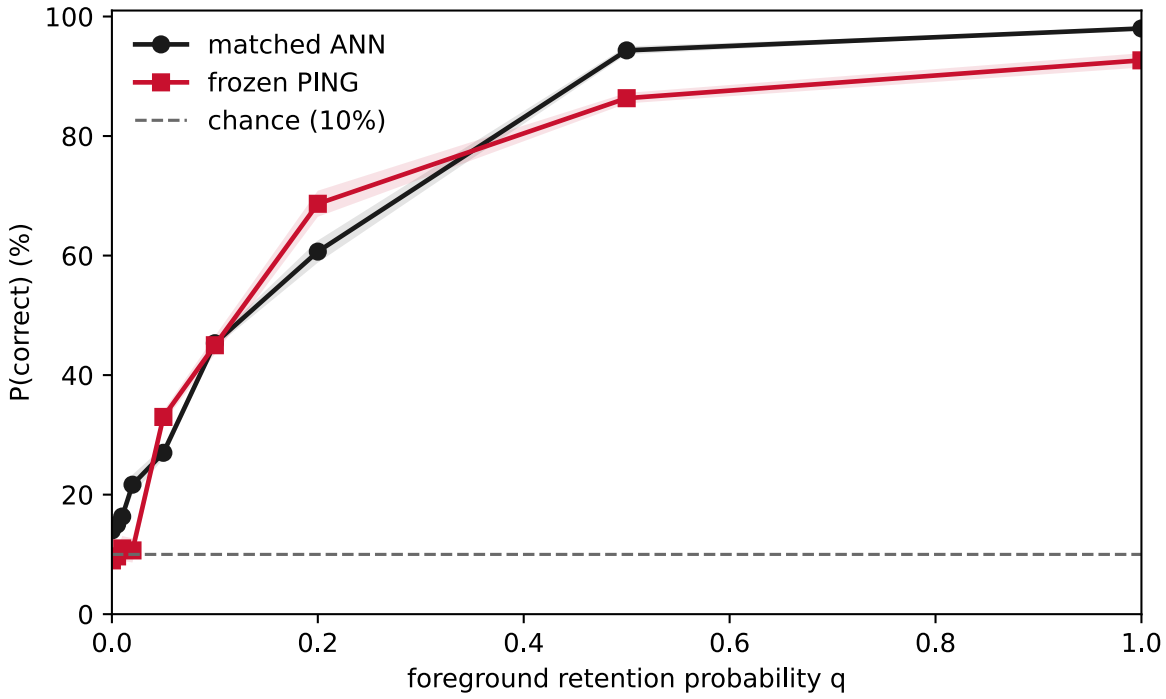


Figure 74: Width-matched ANN and frozen PING accuracy under identical spatial deletion. Probability of a correct classification (%) is plotted against foreground retention q. Black circles denote ANN and red squares PING; bands show one standard error across trained seeds. Each point uses 100 fixed held-out examples and identical independently sampled foreground masks. PING runs at 25 Hz for 200 ms. Neither classifier dominates across the full retention range.

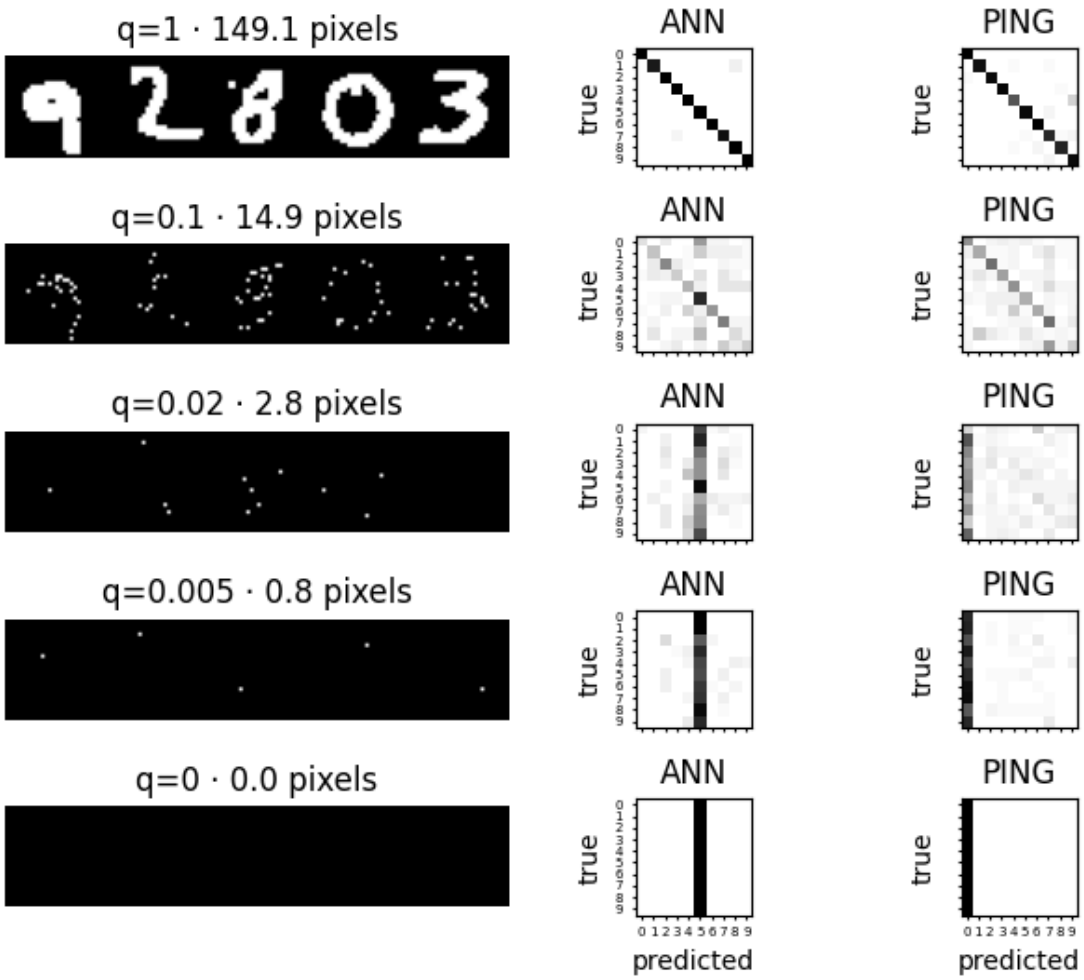


Figure 75: Stimulus and error structure across the matched masking curve. Rows progress from intact input through intermediate masking to the blank control. Left panels show five binarized examples and their mean visible foreground-pixel count. The ANN and PING panels show confusion matrices with true digit on the vertical axis and predicted digit on the horizontal axis; each row is normalized to show the distribution of predictions for one true class. The lowest-retention rows reveal collapse toward model-specific default classes rather than structured digit confusions.

Binarization maps every nonzero antialiased MNIST pixel, including intermediate-intensity pixels introduced to smooth digit edges, to full intensity, so spatial retention is not a second measurement of grayscale contrast. Nevertheless, expected input-event count gives a useful first-order bridge. For an otherwise identical binary image encoded at the masking experiment's 25 Hz ceiling, retaining fraction q gives the same expected event count as retaining the full foreground and using

$$r_{\text{equiv}} = q \cdot 25 \ \text{ Hz}. \quad (9)$$

Here r_{equiv} is the full-foreground Poisson rate with the same expected event count, meaning the expected total number of input spikes across the image and presentation; q is foreground retention; and 25 Hz is the masking experiment's reference encoding rate.

Thus $q = 0.1$ maps to $r_{equiv} = 2.5$ Hz, within the 2–3 Hz transition of the fixed-duration rate curve. At that retention, PING reaches 45%, compared with 49.7% at 2 Hz in the grayscale rate sweep. This numerical alignment supports including a rate near 2.5 Hz in variable-rate training, but it is not an equivalence of corruptions: spatial masking permanently removes locations, whereas lowering Poisson rate preserves all locations in expectation and changes temporal sampling noise.

Gradient descent does not preserve a trainable PING loop

9 June 2026

The trained networks this entry uses are produced once in the shared training hub, exp022 (Training), and reused here rather than retrained.

Abstract

exp025 freezes the recurrent conductances W^{EI}, W^{IE} at biophysical values. When they are trainable, Adam does *not* preserve or recover effective PING from any tested initialisation (canonical, zero, or 10% of canonical): E→I recruitment weakens or remains absent, inhibitory firing collapses, and the network moves toward dense E firing at ≈ 90% accuracy, close to the frozen PING control (≈ 91%). The matrices store non-negative conductance magnitudes; inhibition is produced by the GABA reversal potential, not by a negative W^{IE}. PING is therefore a structural prior imposed by the frozen loop in this setup, not one gradient descent recovers on its own.

Method

Architecture. $N_E = 1024$ excitatory, $N_I = 256$ inhibitory, mem-mean readout, Dale's law enforced. Hyperparameters match the exp025 PING baseline: Adam at lr 4×10^{-4}, batch 256, surrogate slope 1, $W_{\text{in}} \sim \mathcal{N}(1.2, 0.12)$ at 95% sparsity, gradient norm clipped to 1.0, $\Delta t = 0.1$ ms, $T = 200$ ms, no firing-rate regulariser.

Both recurrent matrices are non-negative conductance magnitudes. After each optimiser step they are projected onto the non-negative cone. Their physiological sign is supplied by the pathway reversal potential: I→E contributes $g_I(E_I - V)$ with $E_I = -80$ mV and is therefore inhibitory despite $W^{IE} \geq 0$.

Sweep. Four conditions × three seeds (42, 43, 44) on the 10% MNIST subset (5600 train / 1400 test), 50 epochs. Only the initial (W^{EI}, W^{IE}) and the trainable-or-not flag vary:

Condition	W^{EI}, W^{IE} init	Trainable?
frozen_ping (control)	canonical biophysical	no
trainable_ping_init	canonical biophysical	**yes**
trainable_zero_init	0 (COBA-equivalent start)	**yes**
trainable_small_init	0.1 × canonical	**yes**

Canonical biophysical means $W^{EI} \sim \mathcal{N}(1.0, 0.1)$ µS, $W^{IE} \sim \mathcal{N}(2.0, 0.2)$ µS at $N_I = 256$, fan-in-normalised, so the trainer reports per-edge means of ≈ 0.0010 and ≈ 0.0078. "PING is on" means the inhibitory loop is active: E recruits I, and I paces a gamma rhythm through GABA conductance. "The loop is lost" means E→I recruitment is too weak to sustain that regime, I activity is low or absent, and E fires densely. The firing rates and rhythmicity are the cleanest read of which regime a trained network reaches.

Results

The whole sweep collapses to one picture. In the (E firing rate, I firing rate) plane, a network "found PING" if it sits in the low-E / high-I corner, and "lost the loop" if it sits in the dense-E / silent-I corner.

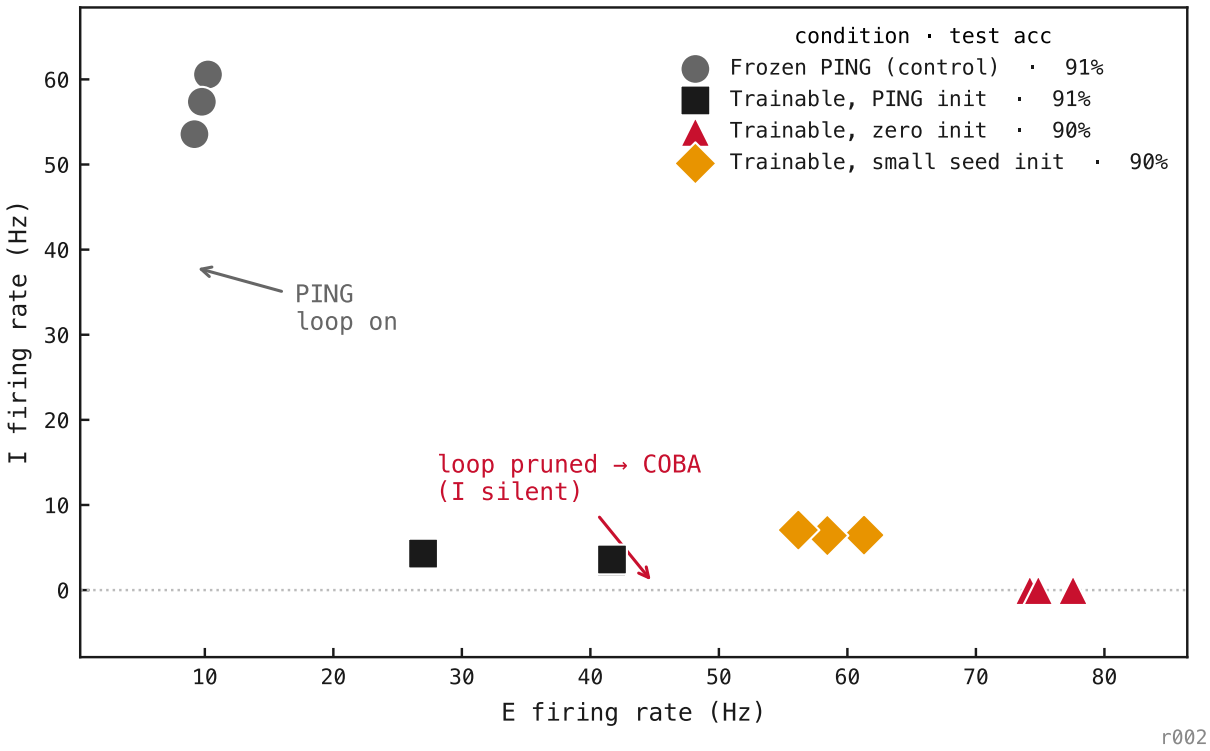


Figure 76: Each point is one trained network (4 conditions × 3 seeds). The **frozen control** (grey) sits in the PING corner, E ≈ 10 Hz and I ≈ 57 Hz, the inhibitory loop alive. **Every trainable condition** (started at canonical PING, at zero, or at 0.1× canonical) collapses to the dense-E / silent-I floor, E ≈ 40–75 Hz and I ≈ 0–7 Hz, at ≈ 90% accuracy, essentially matching the ≈ 91% control. Where the loop *starts* makes no difference; only whether it is allowed to train. Gradient descent never reaches PING, and dropping it costs no accuracy.

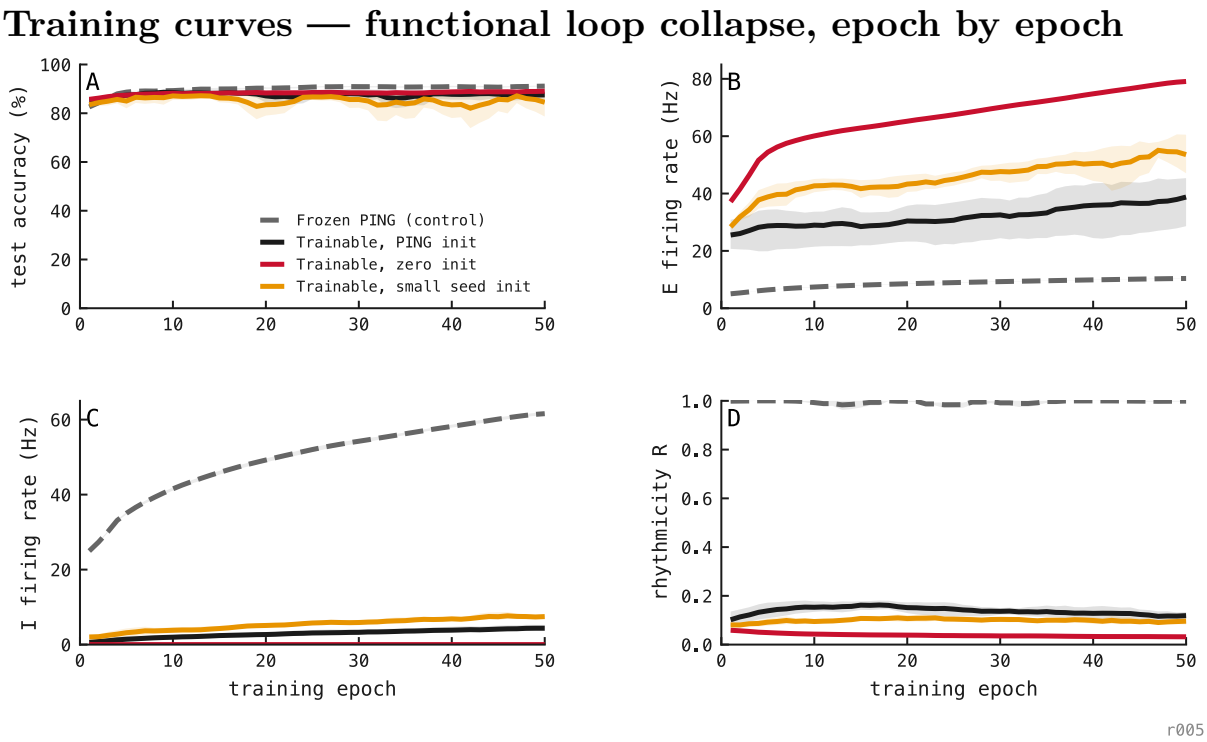


Figure 77: Per-epoch metrics from the runs themselves, coloured by init: PING init (black), zero init (red), small-seed init (amber), frozen-PING control (grey dashed); 10% of MNIST, 50 epochs, 3 seeds each. **Accuracy**: all reach ≈ 85–90% and track each other; losing effective PING costs nothing. **E rate**: the trainable runs climb to ≈ 37–75 Hz as inhibition releases the excitatory population, while the frozen control stays gated near 10 Hz. **I rate**: every trainable run drops to ≈ 0 within a few epochs as effective E→I recruitment weakens or remains absent, while the frozen control's I rate *rises* to ≈ 57 Hz as its readout trains. **Pingness**: the exp054 lobe–trough contrast holds near ≈ 0.99 for the frozen control while every trainable init collapses to ≈ 0.1–0.2. The residual floor is the metric's known low-rate inflation under shared input (exp054), not a surviving rhythm: the frozen-vs-trainable gap is the signal. Functional loop activity dies early, from every start, with no accuracy penalty.

The saved matrices identify which side of the recurrent loop changes. In the canonical-initialisation seed-42 checkpoint, 76.2% of E→I entries are zero after training, compared with 1.1% of I→E entries; their final means are 0.000451 and 0.008705, respectively. Thus the silent inhibitory population is associated chiefly with sparse E→I recruitment, while the I→E conductance remains predominantly positive.

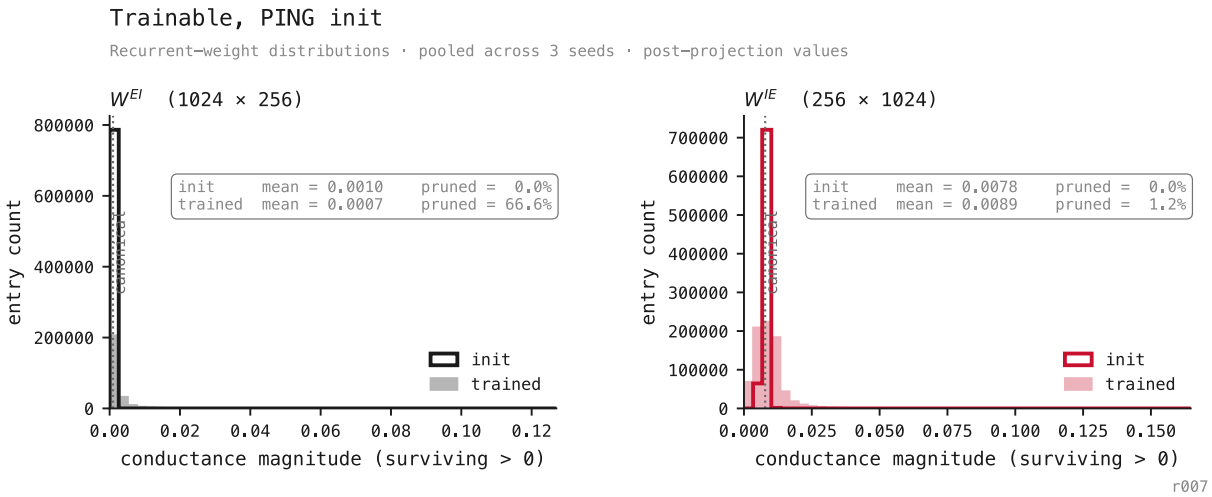


Figure 78: **The two recurrent matrices do not collapse symmetrically.** Initial and trained conductance distributions for the canonical-initialisation condition, pooled across seeds 42–44; each panel reports its own mean and zero fraction. The checkpoint-level distinction is already clear in seed 42: 76.2% of W^{EI} entries are zero after training, versus 1.1% of W^{IE} entries. The loss of PING therefore tracks weakened E→I recruitment and near-silent I activity, not I→E weights crossing into a negative sign cone.

Phase portrait — there is only one attractor under training

The four per-epoch panels above tell the story but separate the two state variables that matter. Putting *E rate* and *pingness* on perpendicular axes shows the trajectories of training directly, and the geometry rules out a misreading: it is *not* the case that PING and COBA are two attractors of the same dynamics, with the architecture deciding the basin. Under training there is **one** attractor (the COBA corner), and the PING state only persists because freezing the loop zeroes its gradients.

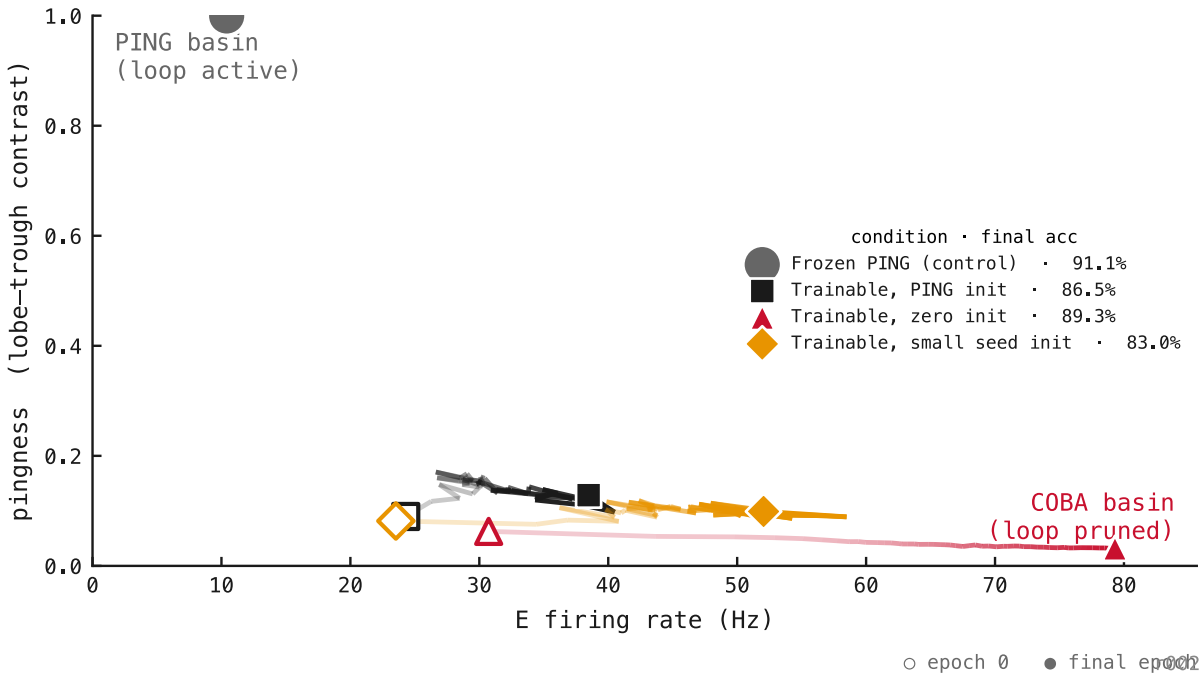


Figure 79: Each trainable trajectory is the per-epoch mean across 3 seeds, alpha-ramped along the epoch axis so the direction of flow reads off the line itself; open markers are epoch 1 (the first logged epoch), filled markers are epoch 50. **Frozen PING (grey)**: sits in the upper PING basin from start to finish at pingness ≈ 0.98; drifts a little in E rate (≈ 4 → 10 Hz) as the feedforward and readout weights train, but the loop weights themselves can't move by construction. **Trainable PING init (black)**: was initialised with canonical biophysical loop weights, but by the time the first metric is logged (after epoch 1) rhythmicity has already collapsed to ≈ 0.17. The trajectory then walks rightward along the COBA floor with the others. **Trainable zero init (red)** and **small-seed init (amber)**: start at the floor and stay there, E rate climbing as the feedforward layer learns the task. Every legend entry lands at 83–91% accuracy, so the move costs nothing. The reading is sharper than the time-series panels suggest: the empty band between pingness ≈ 0.2 and ≈ 0.85 is the *whole story*, because gradient descent flies through it within one epoch and the metric never catches it mid-flight. There is no separatrix between PING and COBA under training because there is no slow trajectory through the gap. The freeze isn't preventing slow erosion; it's preventing instant collapse.

Accuracy–rate trajectory — same destination, different spike economy

Figure 4 puts pingness on its own axis. Putting pingness on the *colour* axis instead and replaying training as (E rate, accuracy) trajectories (the exp025 accuracy–rate frontier given a time axis) gives another reading of the same data: every condition reaches the same final test accuracy, but along very different routes, and only one of them is doing PING while it gets there.

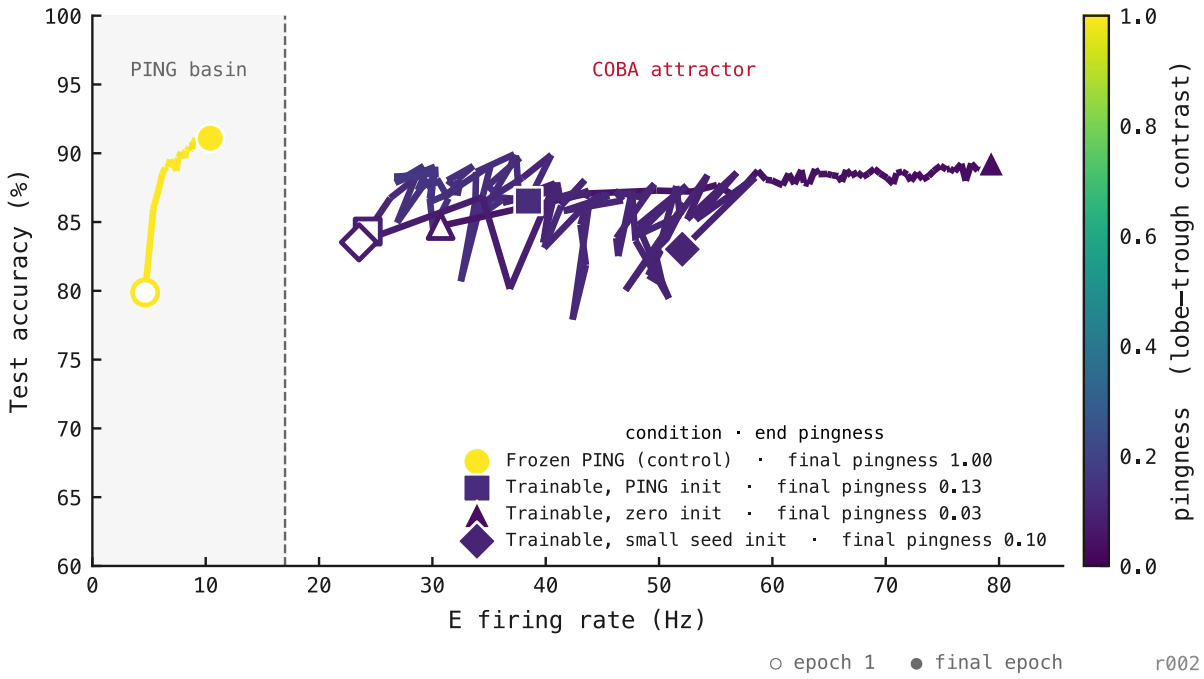


Figure 80: Each trajectory is the per-epoch mean across 3 seeds; per-segment colour is that epoch's pingness on a viridis 0→1 scale (colourbar at right). Open markers are epoch 1, filled markers are epoch 50. The frontier *with time* tells three things at once. **Same destination**: all four conditions land at ≈ 83–91% accuracy by epoch 50, regardless of init or whether the loop trains. **Different routes**: the frozen control climbs accuracy almost vertically with little change in E rate (≈ 4 → 10 Hz); the trainable conditions all bend right and up, the biggest one (zero init) reaching final accuracy at E ≈ 75 Hz, ≈ 8× the frozen control's rate. **Only one of them is still PING**: the frozen control's line is bright yellow because pingness stays high (≈ 1.0) the whole way; every trainable line is dark purple because rhythmicity had already collapsed to ≈ 0.1 by the first logged epoch. Reading the colour bar: most of pingness 0.2–0.85 is unused space, because no trajectory crosses it slowly enough to be logged. Reading the geometry: this is the manuscript's accuracy–rate frontier (exp025, §2.3) animated, and the architecture is the only thing keeping a trained network on the sparse, rhythmic side of it. The dashed divider at ≈ 17 Hz marks the two basins explicitly: frozen PING holds the left, every trainable run ends in the right, at the same accuracy.

A PING rhythmicity metric

15 June 2026

Abstract

A single bounded scalar for how rhythmic a spiking network is: the **lobe–trough contrast** of its spike-time autocorrelation, $(\text{lobe} - \text{trough})/(\text{lobe} + \text{trough}) \in [0, 1)$. Driving each excitatory cell with its own private Poisson input makes the metric rate-invariant by construction; across untrained PING networks it reads 0 along the COBA edges and rises smoothly to 0.98 through the PING interior, a rankable gradient that tracks the gamma-gated collapse of the E firing rate from 95 to 3 Hz.

Methods

The networks here are untrained PING populations driven by external input; the rhythmicity metric is read off the resulting excitatory raster. Write $r(t)$ for the binned population spike count (n bins of width Δt) and ℓ for the lag. The metric is read off in a fixed sequence of steps:

1. **Drive each E cell with its own private Poisson channel.** Every excitatory cell receives an independent homogeneous Poisson spike train at 100 Hz through a one-to-one identity input weight: there is no shared, dense W_{in} projection, so no two cells share an input channel. Private input removes the input-driven spike coincidence that would otherwise inflate the metric at low firing, which is what makes the contrast rate-invariant by construction (Figures 4–5) with no post-hoc correction.
2. **Bin to a population count.** Sum spikes across cells into one count per Δt bin, giving the population trace $r(t)$ of length n.
3. **Raw autocorrelation.** Form the lag product $\sum_t r(t)r(t+\ell)$, which counts spike pairs separated by lag ℓ. All lags are computed at once in $O(n \log n)$ via the Wiener–Khinchin route (zero-pad r, take its FFT, multiply by the conjugate, inverse-transform) rather than the $O(n^2)$ direct sum.
4. **Correct for finite overlap.** Only $n - \ell$ bin-pairs exist at lag ℓ (the last ℓ samples have no partner), so the raw sum tapers toward zero with lag simply from running out of overlap; dividing by that per-lag overlap $n - \ell$ converts it to the *average* product per available pair and flattens the taper.
5. **Set the chance level.** Divide by the mean rate squared $\langle r \rangle^2$ so rate-matched independent firing sits at $A = 1$, and drop the self-paired zero lag. The result is the normalised autocorrelogram

$$A(\ell) = \frac{1}{\langle r \rangle^2} \frac{1}{n - \ell} \sum_t r(t)r(t+\ell).$$

6. **Locate the Mexican hat.** A rhythmic $A(\ell)$ has a "Mexican-hat" profile: a central lobe above 1 (spikes recur a cycle apart) flanked by a dip below 1 where firing is suppressed between volleys. Scanning out from zero lag, take the trough as the first local minimum of a lightly smoothed $A(\ell)$ (it falls near the half-period) and the lobe as the highest point at a shorter lag. Both searches start one bin past zero, so the self-paired zero-lag value

dropped in step 5 is excluded from the lobe height: the lobe is read from the first real lag onward, never the trivial self-correlation.

7. **Read the contrast.** The metric is the **lobe–trough contrast**

$$\text{contrast} = \frac{\text{lobe} - \text{trough}}{\text{lobe} + \text{trough}} \in [0, 1),$$

zero when lobe equals trough (no structure) and approaching 1 as the trough goes silent. It is bounded by construction, with no trough floor needed.

In words: $A(\ell)$ is how much more (or less) likely a spike is to be followed by another one ℓ ms later than under independent firing, with $A = 1$ the chance floor. A central lobe above 1 says spikes *cluster* in volleys; a trough below 1 near the half-period says firing is *suppressed between* them. The contrast is **0** when the spikes carry no such structure (asynchronous) and approaches **1** as sharp volleys separate against near-silence; because it reads the *shape* of $A(\ell)$, it registers a rhythm whether or not its frequency holds still.

Results

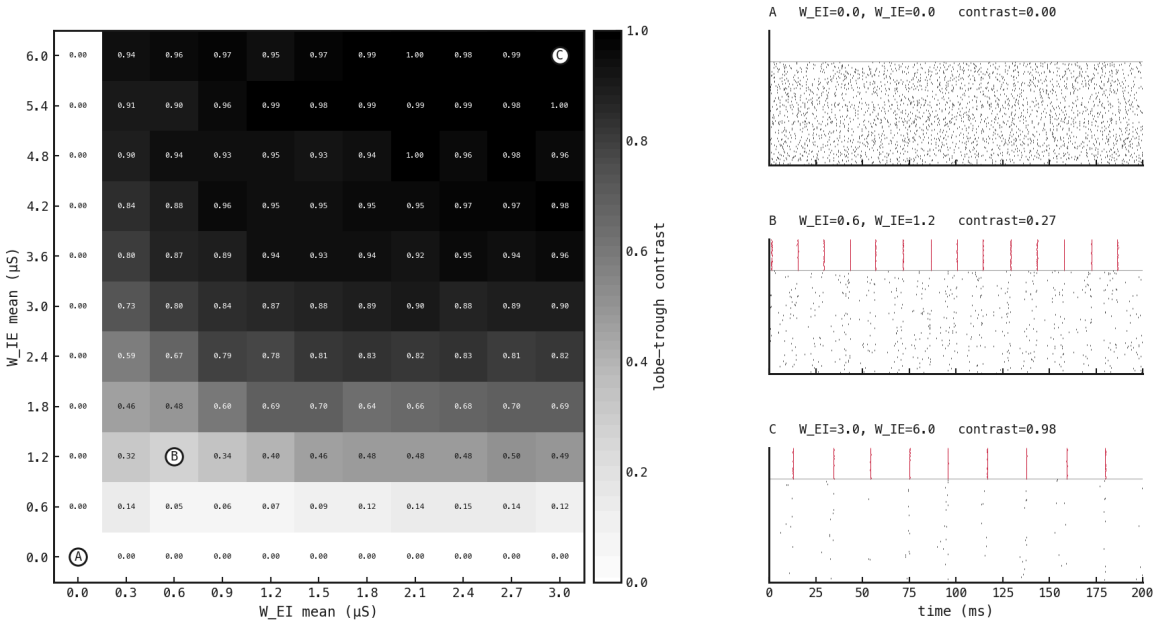


Figure 81: The anchor result: gamma switches on smoothly across the recurrent-weight plane, read as scalar maps over the $W_{EI} \times W_{IE}$ grid (untrained networks, private per-cell Poisson input) with example rasters beneath. **Top**: three per-cell summaries, every cell labelled. **E rate** is high along both zero edges (the loop is broken, E fires at the input-driven ≈95 Hz) and gated down through the interior; **I rate** is silent where $W_{EI} = 0$, runs away along $W_{IE} = 0$ (clipped so the interior is legible), and is controlled once the loop closes; **lobe–trough contrast** (the rhythm scored 0–1) reads exactly 0 along both COBA edges and rises smoothly toward strong coupling, with three points marked along the $W_{IE} = 2W_{EI}$ diagonal. **Bottom**: E/I rasters (E black, I red above) at those points: **A** the fully-off origin ($W_{EI} = W_{IE} = 0$), asynchronous with no I; **B** weak coupling, emerging volleys (contrast 0.27, below the half-way mark); **C** strong coupling, sharp volleys (contrast 0.98). E rate falling, I rate rising, and contrast rising are three readings of the same loop engaging. The full per-cell detail (every raster and autocorrelogram) is in the figures below; the rate-fairness behind the contrast metric is in Figures 4–5.

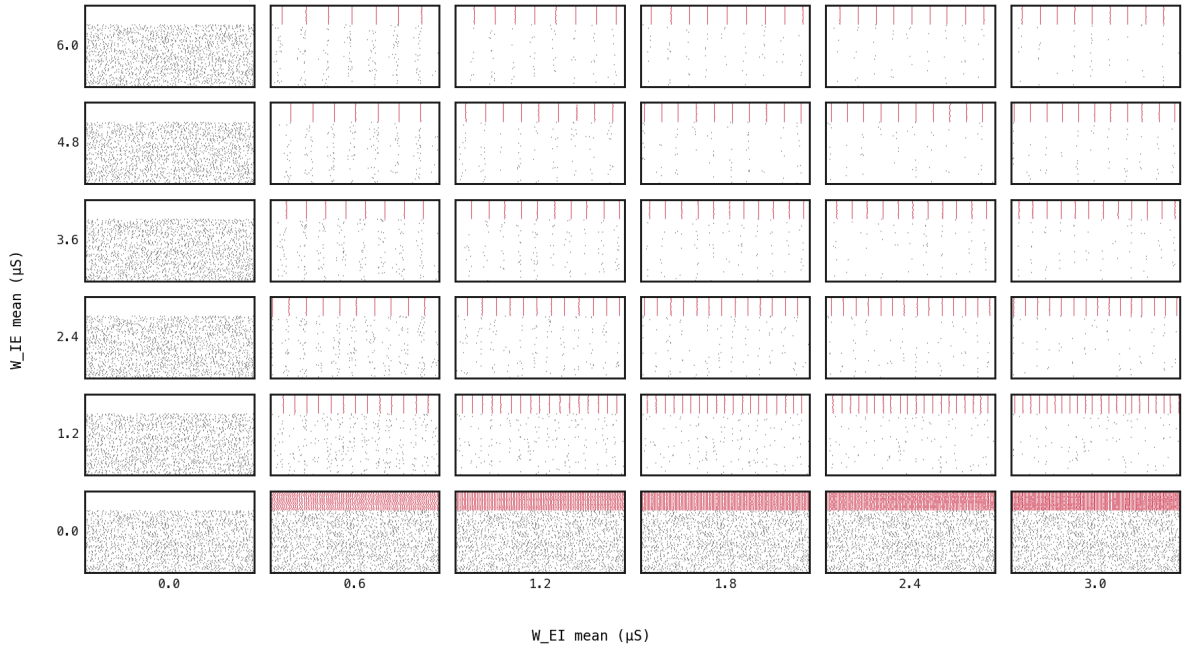


Figure 82: E/I rasters for a 6×6 subset of the grid (every other cell; the heatmaps below use all 121). The two zero edges are the controls: $W_{EI} = 0$ (left column) leaves I silent and E asynchronous; $W_{IE} = 0$ (bottom row) lets I fire but not inhibit E, so both stay dense; neither is rhythmic. The interior shows clear gamma volleys (E black, I red) that sharpen as either weight grows, the rhythm the contrast (Figure 1) scores.

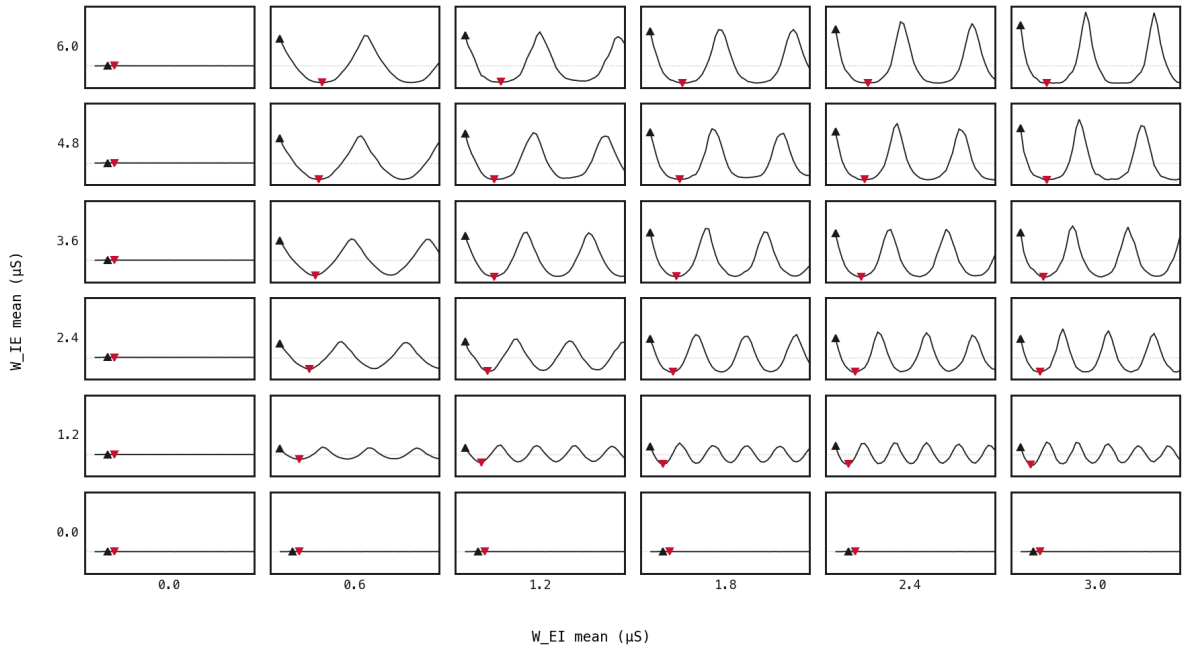


Figure 83: The E-population autocorrelogram $A(\ell)$ for a 6×6 subset of the grid (lag 0–50 ms; dotted line = chance, $A = 1$), with the located lobe (▲) and trough (▼) marked. The two zero edges are flat at 1: asynchronous firing, no structure. Through the interior the Mexican hat emerges: a sharp central lobe at ≈1 ms over a trough near the half-period, with a secondary peak at the full period further out. The contrast in Figure 1 is exactly this lobe-versus-trough read off as one number.

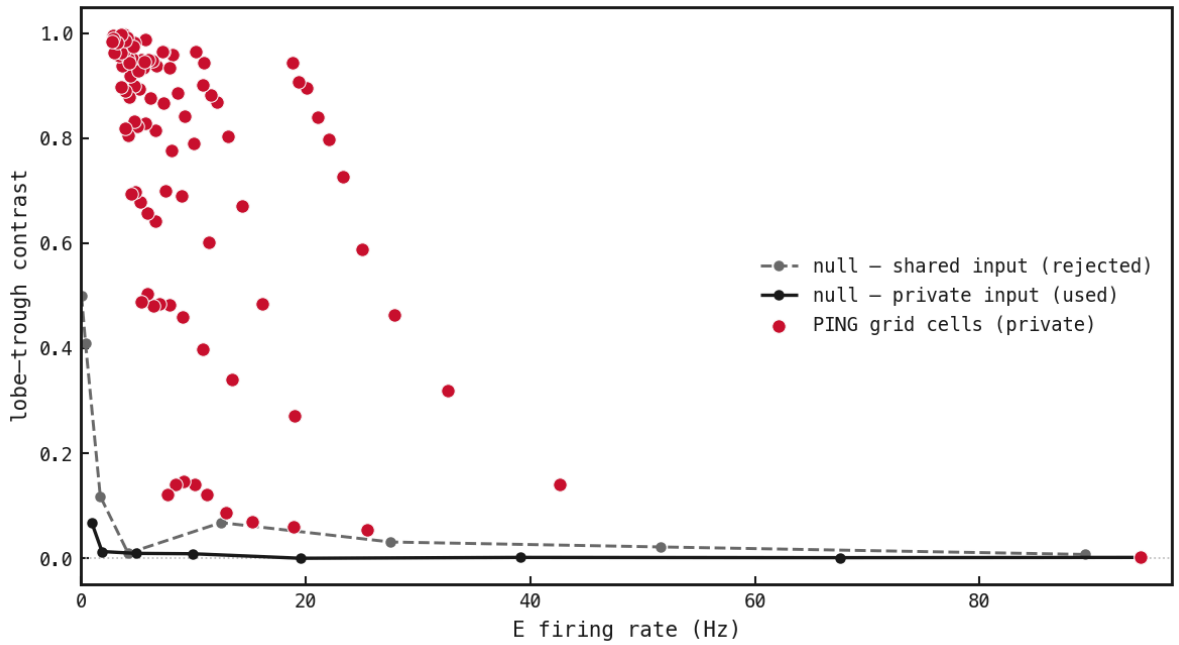


Figure 84: The rate-invariance test that justifies the private-input choice. Each line is a **non-rhythmic null network** (no inhibitory loop, no rhythm at any drive) scanned over input rate; a rate-invariant metric should read ≈0 everywhere. With **private input** (black) it does: flat at ≤0.07 across all firing rates. With **shared input** (grey dashed) it instead **climbs to ≈0.50 as firing thins**: cells sharing input channels fire coincidentally, and the metric reads that as rhythm. The real PING cells (red) sit well above the private-input null, so their contrast is genuine, with no correction needed.

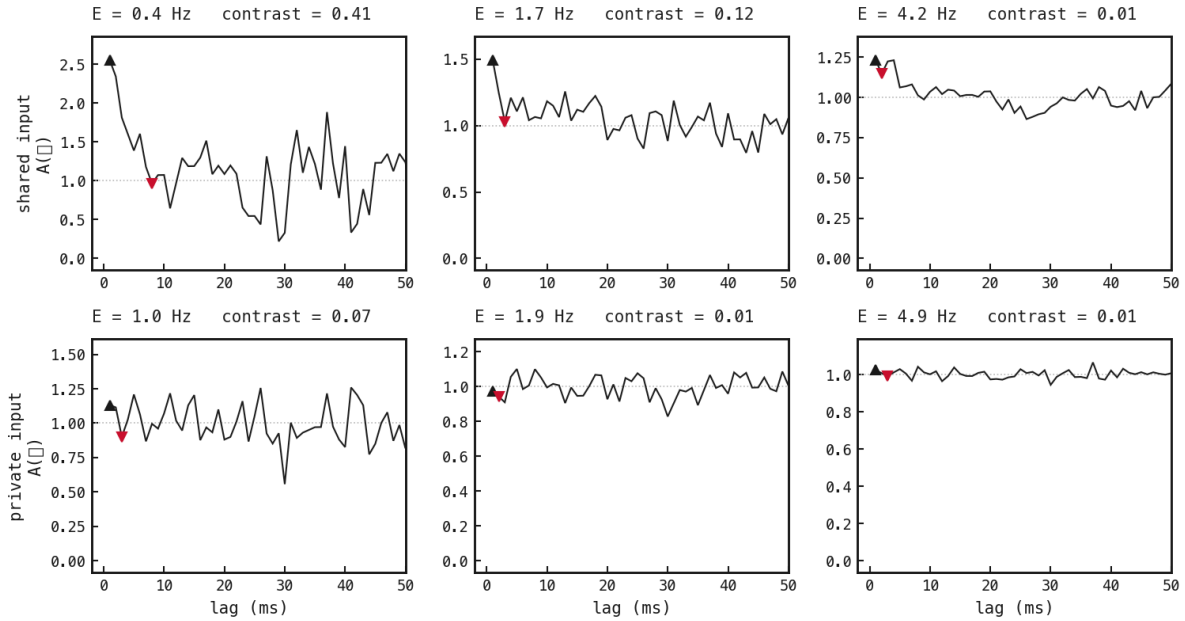


Figure 85: Why shared input fails and private input does not, at matched low firing rates. **Top (shared input):** coincident spikes from shared channels leave a central peak over a shallow dip (a spurious hat with no inhibitory loop behind it), and the metric marks a lobe (▲) and trough (▼) and reports a non-zero contrast. **Bottom (private input):** the same firing rates, but with one channel per cell the central peak is gone; $A(\ell)$ is flat shot-noise around chance and the contrast collapses to ≈0. Same rate, same spike counts; the only difference is whether cells share input.

The onset over its mean-field bifurcation

The turn-on above is *what* the network does; the exp033 4D conductance mean-field is *why*. This final section stacks the two into one manuscript figure: the empirical maps and example rasters directly over the mean-field bifurcation that predicts them. It recomputes the exp033 numerics (Hopf crossing, hysteresis sweep, gamma-vs-τ_{GABA}) and reuses this notebook's own map and raster rendering, so restyling Figure 1 propagates here automatically, and no figures are copied. (This is the anchor that was formerly its own entry.)

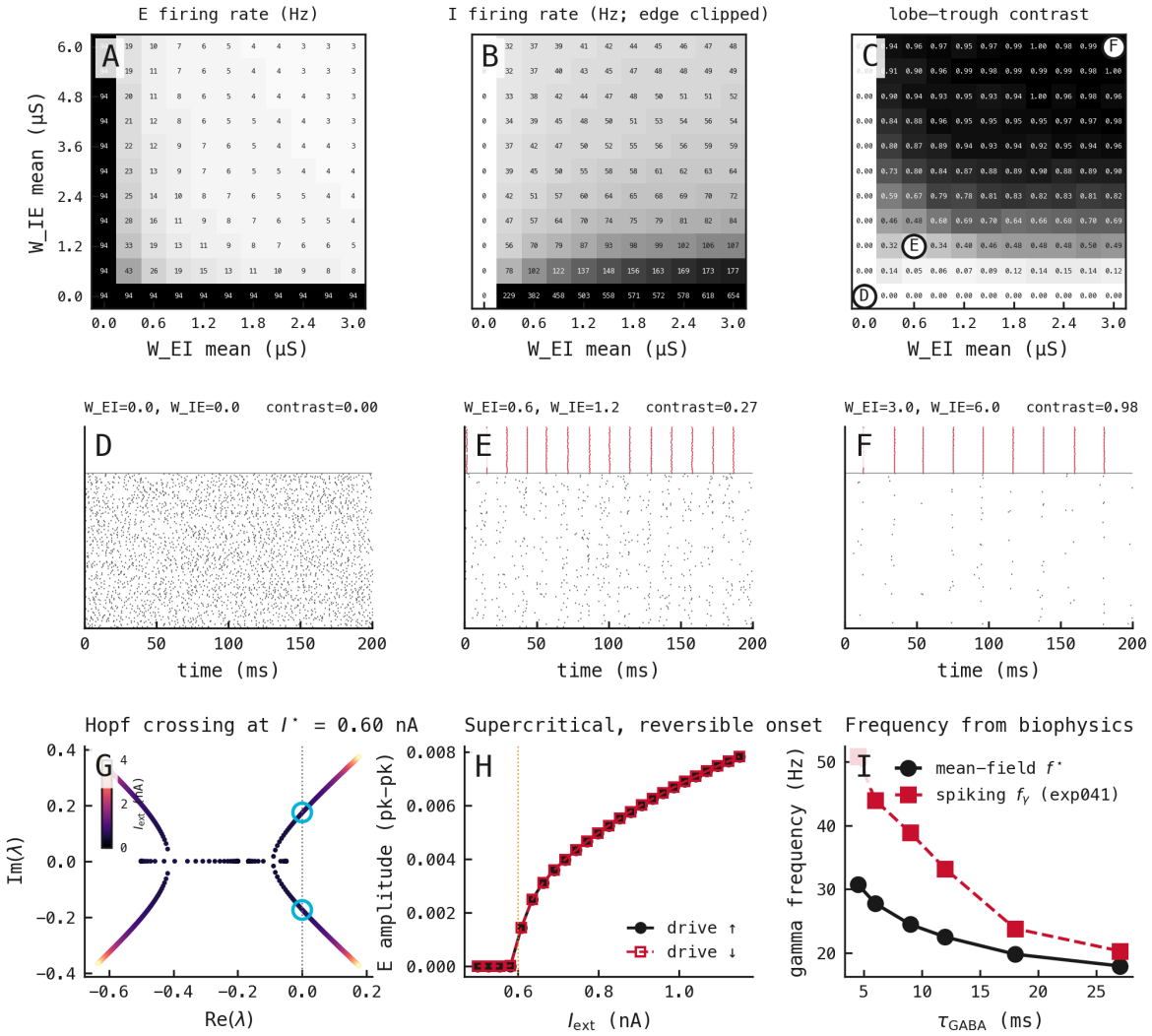


Figure 86: Panels are lettered **A–I** in reading order. **Top (empirics, A–C).** Across the $W_{EI} \times W_{IE}$ plane the E rate falls (**A**), the I rate rises (**B**), and the lobe–trough contrast rises (**C**): three readings of one loop engaging, 0 along both COBA edges and smoothly up toward strong coupling. The contrast map circles three points along the $W_{IE} = 2W_{EI}$ diagonal, shown as rasters **D/E/F**: the fully-off origin (**D**), emerging volleys (**E**), and sharp gamma volleys (**F**). **Bottom (theory, G–I, exp033).** The same onset from a 4D conductance mean-field calibrated from the biophysics: a complex-conjugate eigenvalue pair crosses into the right half-plane at $I^* = 0.60$ nA (a Hopf, **G**), the amplitude rises continuously with coinciding up/down branches (supercritical and reversible, not a hard switch, **H**), and the predicted gamma frequency falls with τ_{GABA} in step with the exp041 spiking measurement (**I**). The smooth empirical turn-on is exactly what a supercritical Hopf predicts.

The Canonical V&S state

24 June 2026

Abstract

Vreeswijk & Sompolinsky predict that a strongly coupled, sparse, inhibition-dominated network falls untuned into a *balanced* state: large E and I currents cancel, and the residual $O(1)$ fluctuations drive irregular (CV ≈ 1), asynchronous, rhythmless firing, *generated* by the deterministic dynamics rather than inherited from the input. We push the conductance-based COBANet into the full four-coupling regime and test three predictions: the balanced signatures at fixed fan-in (Figure 1); their survival as $K \to \infty$ only under strong coupling, synapse $\propto J/\sqrt{K}$ (Figure 2); and that the irregularity is deterministic chaos, by perturbing clones on noise-free input (Figure 3). All three hold: Poisson ISIs, near-zero correlations, a broadband spectrum, a supra-Poisson CV sustained as K grows, and a positive Lyapunov exponent ($\lambda \approx +28$ s^{-1} vs ≈ 0 for a decoupled control). The lone caveat (Figure 2) is a finite-K rate drift.

Methods

1. **Network and the K–J mapping.** COBANet, $N_E = 1024$, $N_I = 256$, $\Delta t = 0.25$ ms, $T = 1000$ ms. **Fixed fan-in** (exact-K, sparsity $s = 0.99$): every cell draws exactly K inputs. **Four recurrent matrices** $W^{EI} = \mathcal{N}(0.6, 0.18)$, $W^{IE} = \mathcal{N}(3.0, 0.9)$, $W^{II} = \mathcal{N}(0.4, 0.12)$, $W^{EE} = \mathcal{N}(0.4, 0.12)$ µS (W^{EE} for completeness: the fixed-K balance sets the rates, not it). **Per-cell independent Poisson drive**, uncorrelated (E: 8 Hz × 2.10 µS, I: 8 Hz × 0.25 µS). The simulator is set by sparsity s and weight w; the theory's fan-in K and coupling J are derived,

$$K = (1 - s)N_{\text{pre}} \quad (1)$$

$$J = \frac{w}{\sqrt{K}} \quad \Longleftrightarrow \quad w = J\sqrt{K} \quad (2)$$

where:

- K, the **fan-in**: how many presynaptic cells each neuron receives input from (the size of its "crowd");
- J, the **coupling**: the $O(1)$ synaptic strength held *fixed* as K grows. The $J/\sqrt{K}$ scaling is the load-bearing V&S choice, keeping the mean recurrent input $O\left(\sqrt{K}\right)$ (large, cancels) while the fluctuation stays $O(1)$ (survives);
- s, connection **sparsity** (the *–ei-sparsity* flag); $1 - s$ is the probability any given pair is wired;
- N_{pre}, size of the presynaptic pool a cell draws from (N_E or N_I);
- w, mean of the recurrent weight matrix in µS (the value passed to *–w-ei* etc.).

Exact-K divides each synapse by its fan-in, so the total a cell receives, $K \cdot (w/K) = w$, is K-independent, giving (2). At $s = 0.99$:

matrix	direction	N_{pre}	w (µS)	K	$J = w/\sqrt{K}$
W^{EI}	E → I	1024	0.6	≈ 10	≈ 0.19
W^{EE}	E → E	1024	0.4	≈ 10	≈ 0.13

W^{IE}	I → E	256	3.0	≈ 3	≈ 1.73
W^{II}	I → I	256	0.4	≈ 3	≈ 0.23

So $K_E \approx 10$, but the 4× smaller I pool gives only $K_I \approx 3$. V&S needs K_E, K_I the *same order* (4:1 qualifies), but $K_I \approx 3$ is marginal: at equal fan-in ($K_I \approx 10$) the state stays balanced and asynchronous, the CV merely relaxing from 1.1–1.2 to ≈ 1.0, so the supra-Poisson burstiness is a lumpy-K_I shot-noise effect, not a balance failure.

2. **Why it is irregular.** Each cell sums ≈ K synapses of strength $\propto J/\sqrt{K}$, so the mean recurrent input is $O\left(\sqrt{K}\right)$ but its fluctuation is $O(1)$. A moderate rate makes the large opposing E and I means **cancel to leading order**, parking the drive near threshold; the residual $O(1)$ fluctuations carry each cell over threshold at random times (Poisson, CV → 1), and since each cell hears its own uncorrelated input, the population is asynchronous. These fluctuations are network-*generated*: they survive $K \to \infty$ only under $J/\sqrt{K}$ scaling (Step 4) and persist under noise-free input (Step 5).

3. **Diagnostics.** From post-burn-in spikes: per-neuron ISI CV, the Welch spectrum of the population-mean E trace, and the mean pairwise cross-correlogram over 100 random E pairs.

4. **The $J/\sqrt{K}$ coupling test.** Strong coupling means synapse $\propto J/\sqrt{K}$, keeping an $O(1)$ recurrent fluctuation as $K \to \infty$; holding w fixed instead gives synapse $\propto 1/K$, the weak (mean-field) limit, fluctuation $\propto 1/\sqrt{K} \to 0$. At fixed $N_E = 1024$ we sweep $K = 10 \to 160$ under both: **strong** (weights $\propto \sqrt{K}$, drive folded into the balance, so rate $\propto K$ and $g \propto 1/\sqrt{K}$) and **weak** (all fixed), reading off whether the irregularity survives. (3 s trials, since the strong rate drifts low at high K and the CV needs enough ISIs.)

5. **Direct Lyapunov exponent (frozen input).** Replace the Poisson drive with a **quenched DC conductance** (a frozen per-cell offset, no per-timestep fluctuation), run two clones on *identical* input, kick every voltage by ε at $t = 0$, and track the **voltage distance** between the copies,

$$\|\Delta V(t)\| = \sqrt{\sum_{i=1}^{N_E} \left(V_i^{\text{clean}}(t) - V_i^{\text{pert}}(t)\right)^2} \quad (3)$$

the Euclidean distance between the two copies' N_E E-cell membrane-voltage vectors (V_i^{clean} unperturbed, V_i^{pert} kicked). With noise-free input any divergence is network-generated. We compare the **balanced** four-coupling net against a **decoupled** control: the same COBA E and I cells under the same DC drive, but with all four recurrent matrices set to ≈ 0, so no cell receives synaptic input from any other. Each decoupled cell is then an independent DC-driven integrator (a clock), and a perturbation has nowhere to spread, fixing the $\lambda \approx 0$ baseline. (It is *not* a feedforward network: there is no input-layer pathway, just the same architecture with every recurrent wire cut.) Spiking dynamics contract between spikes and expand only at spike-flips, so ε must flip spikes ($\varepsilon = 0.1$ mV; smaller just contracts → spurious $\lambda < 0$). The slope of $\log\|\Delta V\|$ in the post-flip, pre-saturation window is the largest Lyapunov exponent (five seeds). It is an *initial-growth* estimate ($\|\Delta V\|$ saturates at the attractor size), so the sign, not the magnitude, is the result.

Results

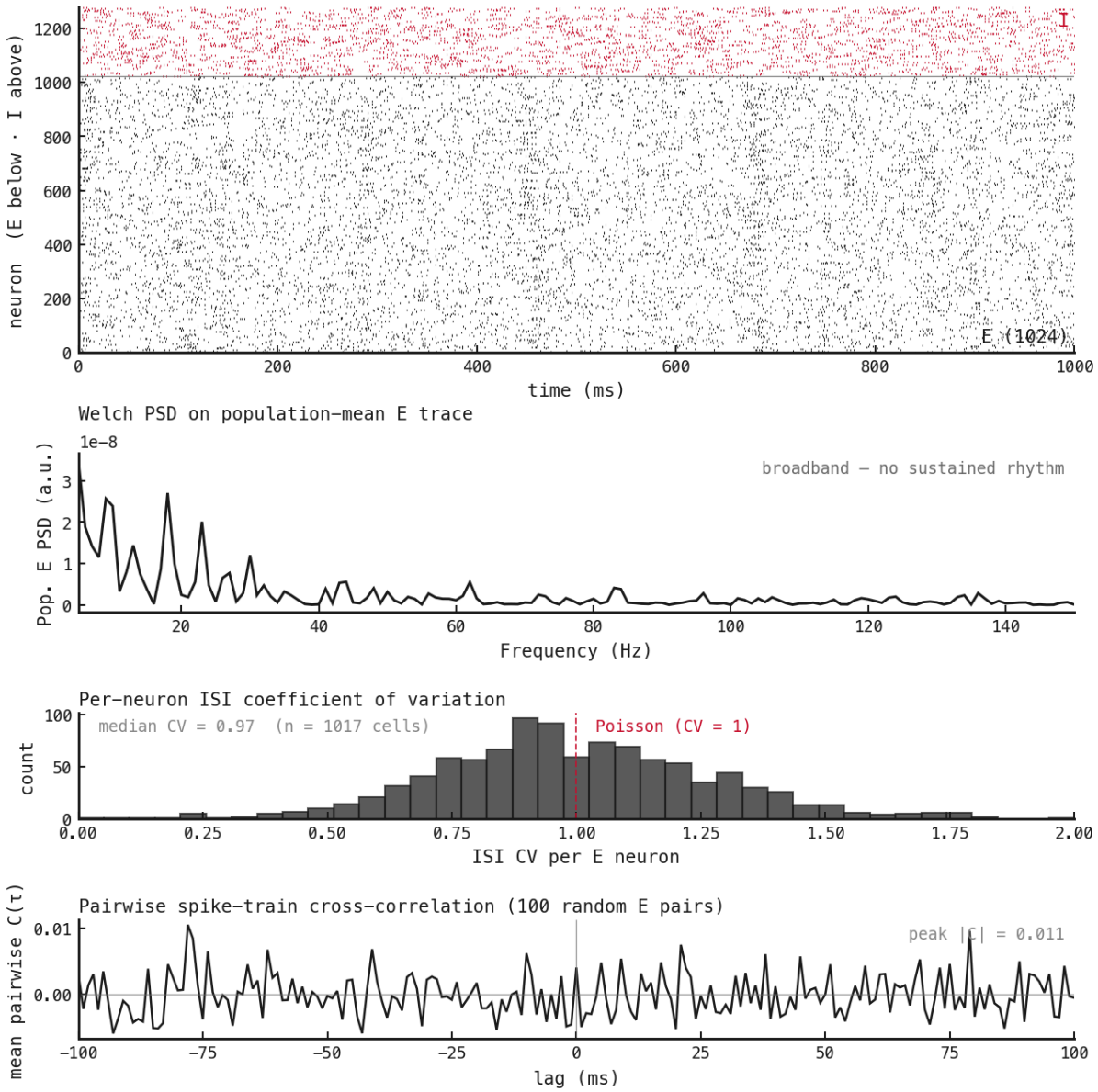


Figure 87: Four diagnostics of the four-coupling state (E black, I red above the divider; ⟨r_E⟩ ≈ 17.5, ⟨r_I⟩ ≈ 25.4 Hz). **What we expect.** Irregular (ISI CV ≈ 1; a constant-current cell sits at 0), asynchronous (near-zero pairwise correlation), rhythmless (broadband, no peak). **What we see.** *Raster:* scattered, no bands. *PSD:* broadband, no sustained rhythm; the single-trial spectral max wanders seed to seed (5–90 Hz across six seeds), so it is not a peak, but a *weak* gamma-band E–I resonance does survive seed-averaging (≈ 2× the floor near 40 Hz, the same E↔I loop that sharpens into PING gamma, here heavily damped). *ISI CV:* median 1.11 (E), 1.20 (I). *Cross-correlogram:* flat, peak $|C(\tau)| \approx 0.01$ despite heavy shared input, the active decorrelation of Renart et al. (2010). **Do they align?** Yes on all three: the COBANet enters the balanced state untuned.

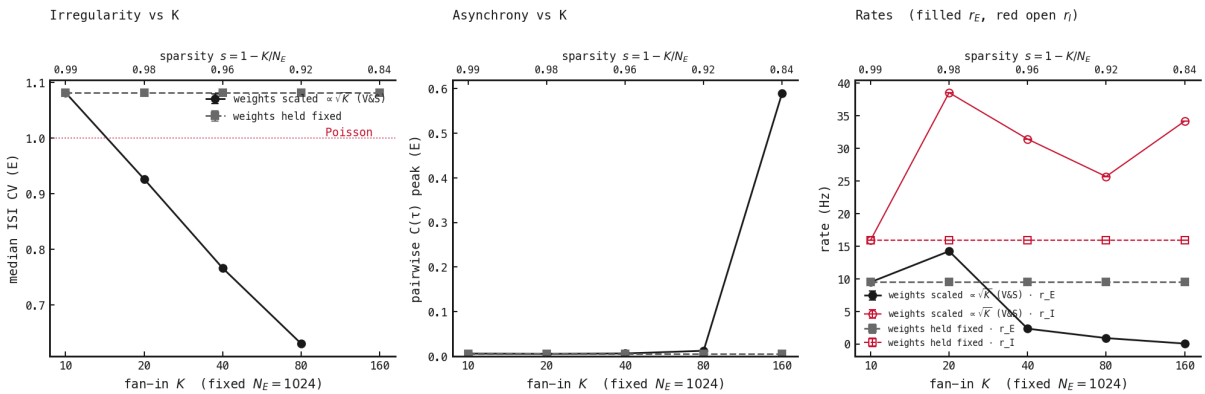


Figure 88: Fan-in $K = 10 \to 160$ at fixed $N_E = 1024$, equivalently sparsity $s = 1 - K/N_E$ from 0.99 to 0.84 (top axis), with recurrent weights scaled $\propto \sqrt{K}$ (black, the V&S $J/\sqrt{K}$ rule) versus held fixed (grey); r_I shown in red. ± 1 SD over three seeds, 3 s trials. **What we expect.** Strong coupling keeps the $O(1)$ fluctuations, so irregularity *survives* as K grows; weak coupling loses them ($\propto 1/\sqrt{K}$), so cells *regularise*. Asynchrony holds in both. **What we see.** *Irregularity:* strong stays supra-Poisson (CV 1.2–1.3, easing to 1.11 at $K = 160$); weak decays to ≈ 1.0, the Poisson floor. *Asynchrony:* both ≈ 0.006 throughout. *Rates:* strong r_E drifts 17.5 $\to$ 4.9 Hz (the $O\left(1/\sqrt{K}\right)$ finite-K correction); weak holds ≈ 17 Hz. **Do they align?** Yes: only $J/\sqrt{K}$ keeps the recurrent irregularity alive as K grows, and the gap is the signature. The one mismatch is the strong-coupling rate drift (Next steps); whether the survival is *self-generated* is settled by Figure 3.

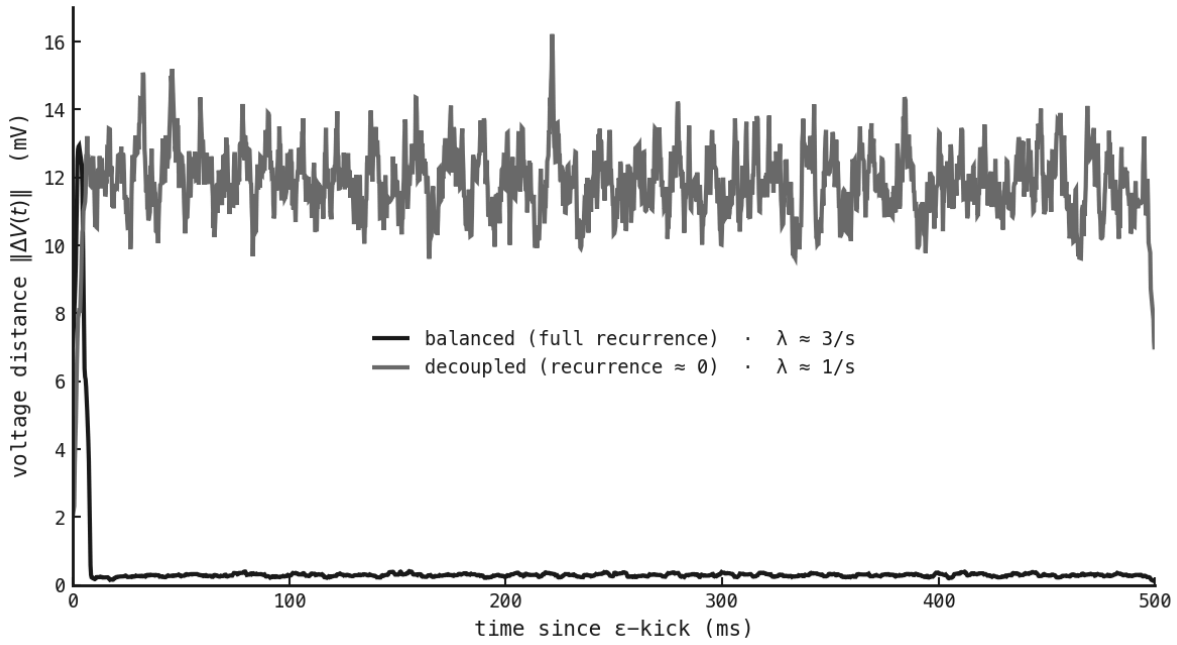


Figure 89: Two clones on *identical* quenched-DC input, the second kicked by $\varepsilon = 0.1$ mV at $t = 0$; voltage distance $\|\Delta V(t)\|$, seed-mean (bold) with a ± 1 SD band over five seeds, lightly smoothed. **What we expect.** If chaotic, the kick is *amplified* ($\|\Delta V\|$ grows exponentially, $\lambda > 0$), and with noise-free input that can only be network-generated. The decoupled control should not grow ($\lambda \leq 0$). **What we see.** *Balanced (black):* $\|\Delta V\|$ rises steeply (fitted rate $\lambda \approx +28$ s^{-1}) then saturates at the attractor size (≈ 250 mV). *Decoupled (grey):* neither amplified nor forgotten, a fixed phase offset, $\lambda \approx 0$. **Do they align?** Yes: $\lambda > 0$ for the balanced net, ≈ 0 for the control, deterministic chaos, self-generated. Caveat: an *initial-growth* estimate (saturation caps the window), so the sign, not the magnitude, is the result; a renormalised Benettin scheme would pin exact λ (Next steps).

Next steps

Figures 1–3 confirm the balanced signatures, their $J/\sqrt{K}$-only survival, and a positive Lyapunov exponent: the state is self-generated and deterministically chaotic. Two refinements remain.

1. **Hold the rate as K grows.** Strong-coupling r_E drifts 17.5 → 4.9 Hz (the $O\left(1/\sqrt{K}\right)$ finite-K correction). A DC offset to E alone *fails*: it pushes E off the balanced point and inflates CV to 1.4–1.7. Honest pinning must co-adjust the E *and* I drives along the balanced manifold (solve the 2D balance for a target (r_E, r_I)), giving a clean constant-rate CV comparison.
2. **Pin down the asymptotic λ.** Figure 3 gives the sign; the magnitude is an initial-growth estimate (ε-dependent, saturation-capped). The exact exponent needs a *renormalised* Benettin scheme (lockstep clones, periodic rescaling of ΔV, averaging log-growth over a long trajectory), which also yields the full spectrum and the attractor dimension.

What the Spiking Heidelberg Digits look like

11 July 2026

Abstract

The Spiking Heidelberg Digits (SHD) [**1**] benchmark delivers spoken digits as cochlear spikes. Unlike the static MNIST images the gamma-gated-sparsity collection Poisson-encodes into spike trains before its conductance-based spiking network (COBANet) ever sees them, SHD arrives *already* as events, with no image and no dense array anywhere: each sample is a spoken digit passed through a model of the inner ear, so it is a list of events, each an event time and the cochlear channel that fired. Before training a network on it, this entry looks at the raw data: one utterance per class, then several utterances of a single digit. Different classes paint visibly distinct time–frequency signatures and repeated utterances of one digit share a family resemblance, so the data is well-behaved and separable enough to be worth training on.

Methods

Nothing here runs the network: every raster is drawn straight from the raw SHD event lists, with no binning and no model.

1. Load the SHD train split as raw events: $N = 8156$ training utterances (a held-out test split adds more), each a spoken digit recorded from several speakers and converted to spikes by a Cramer et al. [**1**] cochlea model, a computational model of the inner ear that maps sound onto firing across an array of frequency-tuned channels.
2. Draw the class gallery: the first utterance of each of the 20 classes (the digits 0–9 spoken in German, labels 0–9, then English, labels 10–19), each as a spike raster.
3. Draw the within-class spread: the first 4 utterances of class 0, *null*, side by side.

Every utterance is a set of events $\{(t_k, u_k)\}$, where:

- t_k, the **spike time** of event k, in seconds (utterances run ≈ 0.2–1.4 s, median ≈ 0.7 s);
- $u_k \in \{0, ..., 699\}$, the **cochlear channel** that fired: a place code for frequency, where a low channel ≈ low pitch and a high channel ≈ high pitch.

So an utterance carries $C = 700$ input channels and a median of ≈ 7605 events per utterance (range ≈ 2410–14917).

Results

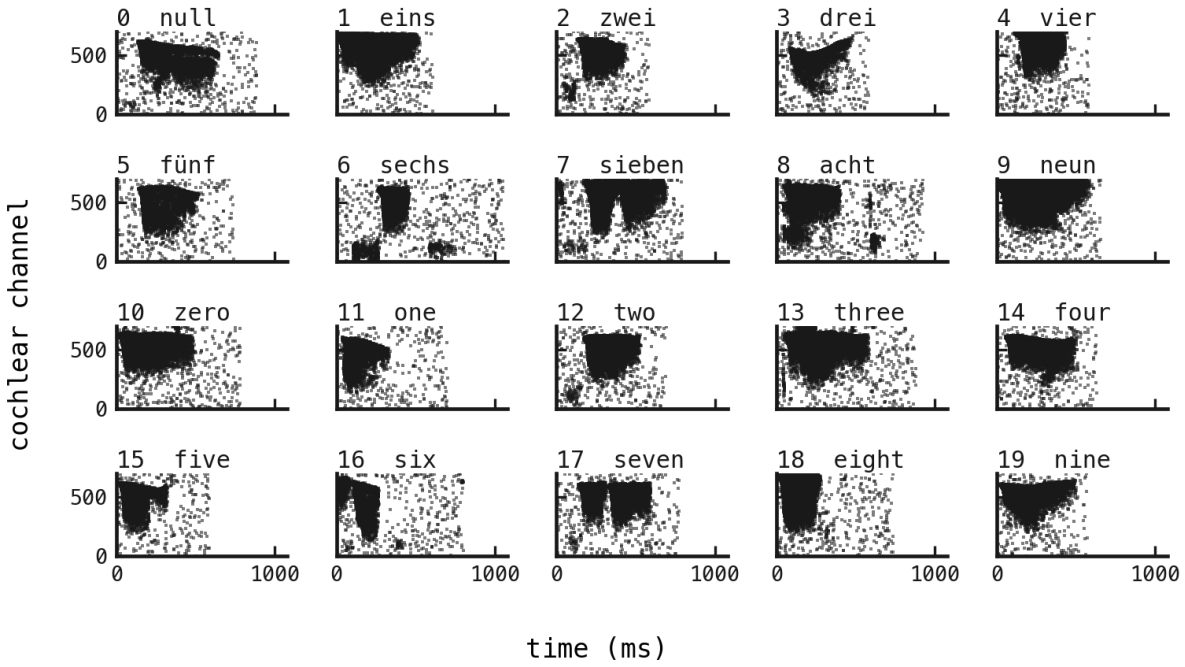


Figure 90: One utterance per class (German 0–9, then English 0–9), each a raster of time (ms) against cochlear channel. **What we expect.** If SHD is learnable, different words should paint different time–frequency signatures. **What we see.** They do: a compact high-channel onset for *zwei*, a long two-lobe sweep for *sieben*/*seven*, a low-channel tail on *sechs*. A diffuse haze of isolated events sits under every panel: the cochlea model's spontaneous background firing, which the classifier has to see past.

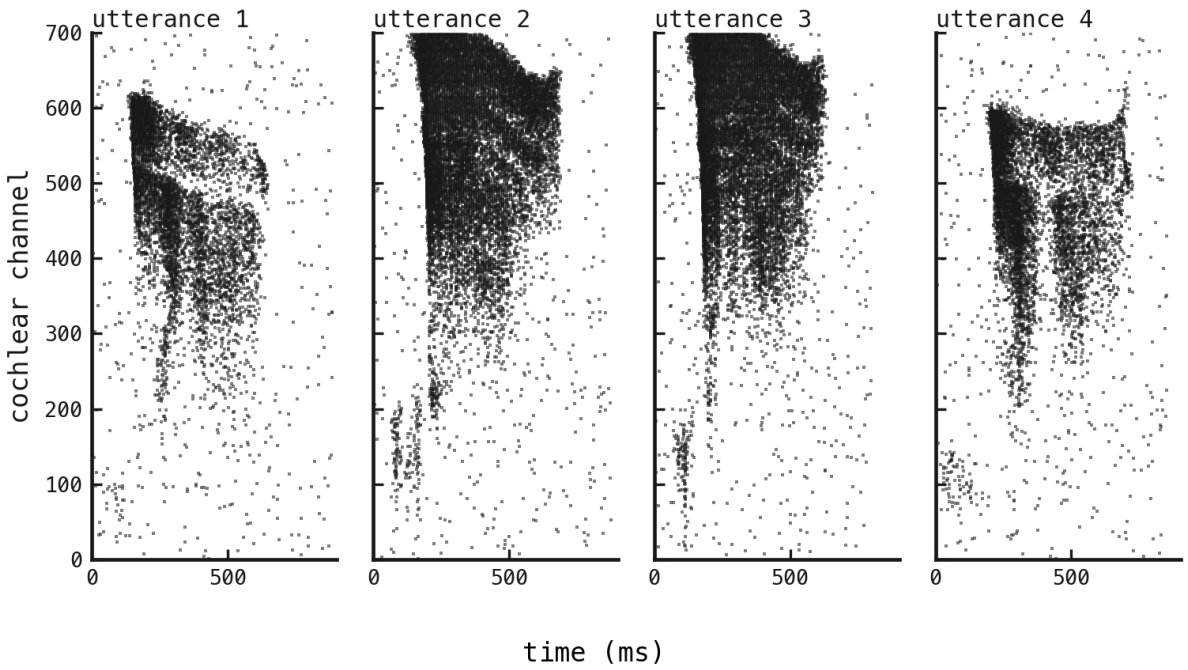


Figure 91: 4 different utterances of class 0, *null*, each a raster of time (ms) against cochlear channel: the within-class spread. **What we expect.** Different speakers saying the same word should share a family resemblance while differing in the particulars. **What we see.** Exactly that: all 4 carry the same broad high-channel onset falling into a mid-channel body, but they differ in duration, event density, and the fine structure of the low-channel tail. This speaker variability is what a classifier trained on SHD must generalise over.

Next steps

The data is well-behaved and visibly class-separable, so it is worth training on. The trainer accepts SHD directly: it bins these events onto the model's integration grid (a spike tensor of shape $T_{\text{steps}} \times 700$ at the integration timestep Δt) and feeds them to the PING (pyramidal–interneuron gamma) network, the excitatory and inhibitory populations of the COBANet, where:

- T_{steps}, the number of time bins on the integration grid;
- Δt, the integration timestep, in seconds.

The next entries take that path: a first PING network trained on SHD, then the firing-rate regulariser (the upper firing-rate bound from Cramer et al. [**1**], with target $\theta_u = 100$ and weight $s_u = 0.06$) that event-based benchmarks need to keep the hidden-layer rates in check, where:

- θ_u, the target upper bound on a unit's firing rate;
- s_u, the strength of the penalty applied above that bound.

References

1. Cramer, Stradmann, Schemmel & Zenke — *The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks*. IEEE Transactions on Neural Networks and Learning Systems, 2020. doi:10.1109/TNNLS.2020.3044364

Trying a cumulative-potential readout on matched SHD networks

19 July 2026

Abstract

This one-seed exploratory experiment tests whether the newly merged `tools/snn` cumulative-potential decoder can move matched Dale-constrained COBA and PING networks above the exp068–exp070 validation plateau on SHD. The screening ladder uses the same deterministic development-training / held-out-validation split as exp069 and exp070; the official SHD test remains sealed. Candidate one keeps the baseline 256-cell architecture and changes the classifier to `--readout cumulative-potential --signed-readout`

`--readout-bias`. If that is finite, active, and promising, candidate two keeps the same decoder and tests two 256-cell hidden layers. Only the best short candidate may be promoted to forty epochs.

Goal prompt

```
/goal Design and execute the next exploratory SHD experiment on branch night/
spiking-heidelberg-digits/ar071, using the newly merged tools/snn cumulative-
potential readout additions to try to raise validation accuracy while keeping
the comparison between matched Dale-constrained COBA and PING networks
scientifically clean.

Start from updated main at merge commit
9527162d7a0ef80deac0e3f606ce513b9280ba6c. Do not modify main. Open a new
draft PR for this experiment when the first meaningful commit is ready.

Scientific objective:
Run a one-seed exploratory validation-only ladder on SHD to identify whether
stronger temporal readout and modest capacity changes materially improve
matched COBA/PING learning beyond exp068-exp070.

Constraints:
- Use one seed only.
- Do not use the official SHD test set during exploratory screening.
- Use a held-out validation split from the training set for all selection
decisions.
- Keep COBA and PING matched except for their registered cell-specific
voltage-gradient dampening.
- Preserve matched input, readout, training split, optimizer, batch, and
preprocessing settings across COBA and PING within each candidate.
- Use short runs first for iteration speed, targeting about 8-10 epochs per
candidate.
- Promote only the most promising candidate to a 40-epoch validation run.
- Do not run multiple seeds unless a later claim is worth defending and I
explicitly authorize it.
- RunPod spending requires explicit approval before pod creation; use at most
one pod at a time unless separately authorized.
```

```
- Reap any pod when finished.
- Do not merge the PR.
- Avoid the full local test suite on the 4 GB Hetzner host; use focused
checks and demolab build.

Candidate ladder:
1. Baseline architecture with the new signed cumulative-potential readout:
   --readout cumulative-potential --signed-readout --readout-bias
2. If candidate 1 is finite, active, and improves validation learning, try
modest increased capacity with two hidden layers, e.g. --n-hidden 256 256,
keeping the same readout.
3. Optionally test one conservative recurrence/readout-adjacent setting if
supported by existing tools/snn CLI and scientifically justified before
seeing final results.
4. Promote the best candidate, if any, to a matched 40-epoch COBA/PING
validation run.

Deliverables:
- Create the next numbered experiment, likely exp071, with artifacts under
artifacts/data/exp071.
- Produce numbers.json, provenance, reproducer, training curves, firing-rate
diagnostics, and matched input/E/I rasters where relevant.
- Write writings/exp071.typ as the canonical cold-readable experiment report.
- Include the goal prompt below the abstract and append timestamped activity-
log/thread checkpoints in the experiment appendix, following the simplified
SHD organization.
- Record scientific decisions, failures, anomalies, costs, commits, results,
and pending work in the experiment article.
- Sanitize any publishable transcript/log content before committing.
- Build and validate with focused checks plus demolab build.
- Commit meaningful attempts, including killed attempts, with clear messages.
- Push branch night/spiking-heidelberg-digits/ar071 and update the new PR
with evidence.
- Finish at the human review gate with a concise evidence summary, exact
compute spend, validation performed, and links to the rendered exp071 file
and PR.
```

Methods

Locked comparison

Both cells use seed 42, the exp069/exp070 development split (7340 training utterances and 816 validation utterances), batch size 32, Adam learning rate 0.0004, 1 ms simulation steps, 1000 ms utterance windows, fixed Dale-constrained feed-forward/recurrent conductances, and identical input preprocessing. COBA disables the inhibitory loop and uses the engine's no-dampening value; PING enables the inhibitory loop and uses voltage-gradient dampening 1000. These cell-specific settings are the only intended COBA/PING difference inside each candidate.

The readout is the new cumulative-potential decoder with signed abstract classifier weights and a trainable readout bias. The readout initialisation scale is one, not the old membrane-readout scale of 225, because that scale belonged to the prior output-LIF membrane recipe. COBA and PING are still matched exactly within each candidate.

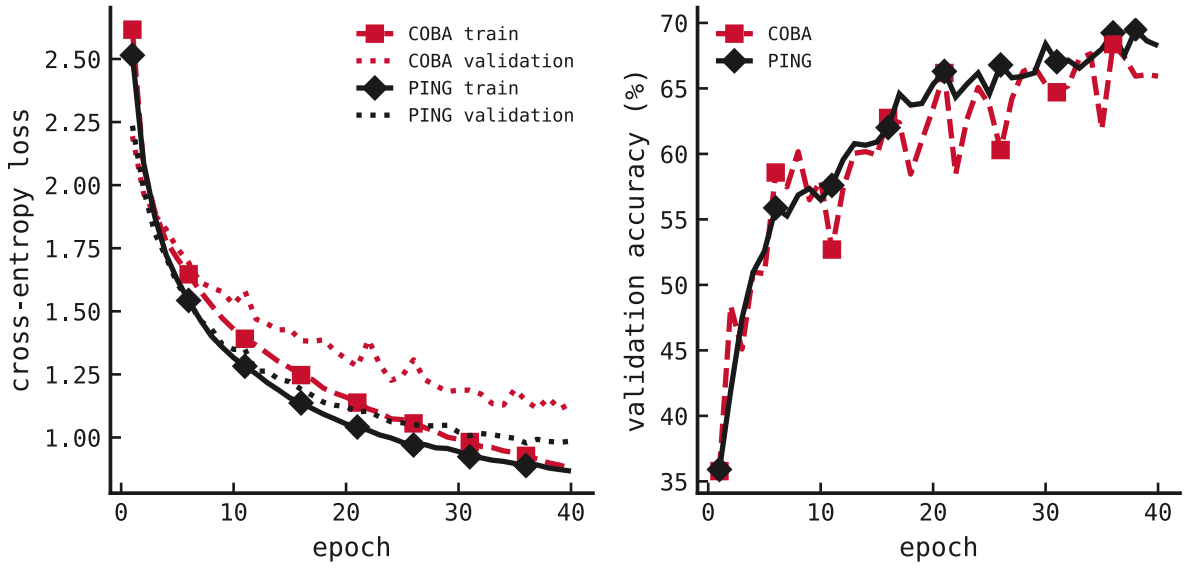
Ordered exploratory ladder

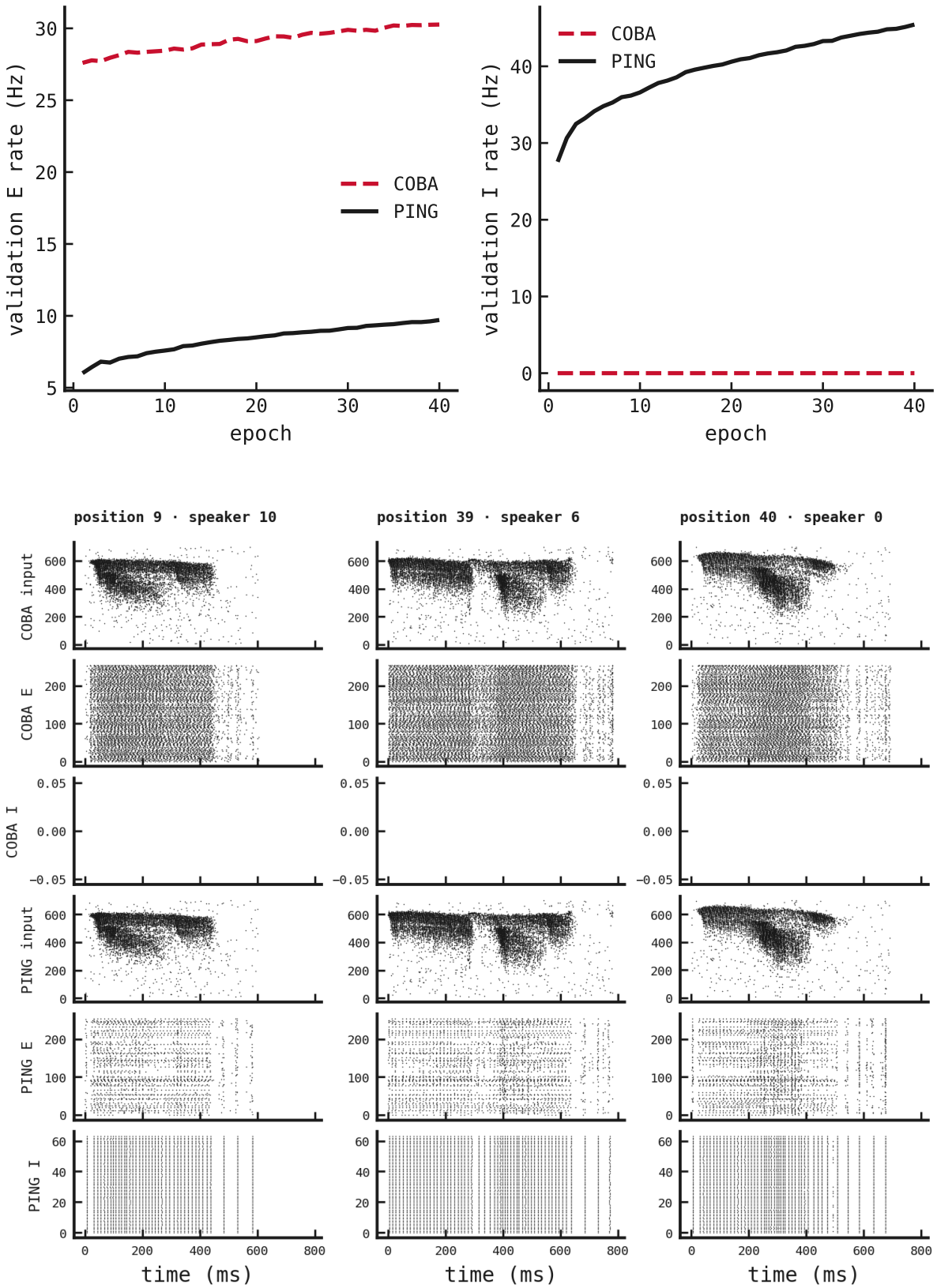
Order	Candidate	Short epochs	Status
1	Baseline 256-cell architecture plus cumulative-potential readout	40	registered
2	Two hidden layers, 256 cells each, same readout	40	conditional

Candidate selection uses held-out validation accuracy and cross-entropy only. Training must remain finite, active, non-saturated, and free of skipped or non-finite updates. The official test file is not staged or loaded by the runner.

Current evidence

The current published attempt is `#r.attempt` at stage `#r.stage`. COBA selected 68.38% validation accuracy; PING selected 69.49%. Exact spend was 3.956 USD , with 0 active pods after collection.





This final forty-epoch promotion keeps the candidate-one architecture and readout. COBA selected epoch 36 at 68.38%, while PING selected epoch 38 at 69.49%. Both final cells were finite, active, clean, and produced matched rasters. In this one-seed validation run PING is

ahead of COBA by 1.1 percentage points; this is exploratory validation evidence, not a defended multi-seed claim.

Activity appendix

Checkpoint cp001

Timestamp: `#trace.checkpoint_time_utc`. Sanitized source hash prefix: `#trace.sanitized_sha256_prefix`. The log excludes hidden reasoning, tool payloads, credentials, private paths, environment values, addresses, and sensitive infrastructure details.

Decisions, actions, and pending work

1. The experiment is registered as exp071 on branch night/spiking-heidelberg-digits/ar071 from main merge commit 9527162d7a0ef80deac0e3f606ce513b9280ba6c.
2. Candidate one is fixed before results as the baseline 256-cell architecture with cumulative-potential readout, signed abstract classifier weights, and readout bias.
3. The old membrane-readout initialisation scale of 225 is not carried into the cumulative decoder; readout scale is one for the new decoder, matched across COBA and PING.
4. Candidate two is conditional on candidate-one viability and keeps the same readout while using two 256-cell hidden layers.
5. Selection remains one-seed and validation-only on the exp069/exp070 split. The official SHD test remains unavailable to the runner.
6. No RunPod pod has been created in this checkpoint and spend remains 0 USD.

Visible messages

User 2026-07-19T11:18:05.855Z

Role: user · Checkpoint: cp001 · Session: exp071-continuation-redacted

<codex_internal_context source="goal"> Continue working toward the active thread goal: design and execute the next exploratory SHD experiment on branch night/spiking-heidelberg-digits/ar071 using the newly merged cumulative-potential readout additions, with one seed, validation-only selection, local short runs before any promoted forty-epoch run, no official test use, explicit RunPod approval before spending, focused checks, artifacts under artifacts/data/exp071, writings/exp071.typ, timestamped activity checkpoints, commits, push, PR update, and no merge. </codex_internal_context>

Assistant 2026-07-19T11:18:05.855Z

Role: assistant · Checkpoint: cp001 · Session: exp071-continuation-redacted

I'll pick this up from the actual repo state, keep the dirty article-consolidation work isolated, and start by setting up the exp071 branch/work area cleanly. First I'm running the lab manual hook, then I'll inspect what's already in place.

Assistant 2026-07-19T11:18:05.855Z

Role: assistant · Checkpoint: cp001 · Session: exp071-continuation-redacted

Because this goal includes opening/updating a PR, I'm using the GitHub workflow skill now; it'll keep the GitHub side tidy and bounded.

Assistant 2026-07-19T11:18:05.855Z

Role: assistant · Checkpoint: cp001 · Session: exp071-continuation-redacted

The main checkout has the older article-consolidation changes restored and dirty, so I'm not touching it. I'm making a separate Git worktree for night/spiking-heidelberg-digits/ar071; that gives exp071 a clean branch without risking those staged/unstaged edits.

Assistant 2026-07-19T11:18:05.855Z

Role: assistant · Checkpoint: cp001 · Session: exp071-continuation-redacted

I've got the design shape now: exp071 can inherit exp069's split, selection, validation-only test seal, rasters, and RunPod plumbing, while overriding only the candidate-specific readout/hidden sizes/epochs. I'm going to add the initial scaffold plus a pre-result artifact so the report builds before paid compute.

Assistant 2026-07-19T11:18:05.855Z

Role: assistant · Checkpoint: cp001 · Session: exp071-continuation-redacted

One design decision I'm making explicit before results: candidate 1 uses the cumulative decoder with –signed-readout –readout-bias and a readout init scale of 1.0, not the old membrane-readout scale of 225. That scale was specific to the previous output-LIF membrane recipe; keeping it would be an unregistered hybrid rather than the clean new readout test. COBA/PING remain matched within each candidate.

Checkpoint cp002

Timestamp: `#trace-two.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-two.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. Focused Python compilation passed for experiments/exp071.py.
2. Focused Ruff initially caught an unused import in experiments/exp071.py; the import was removed and Ruff then passed.
3. RunPod dispatch was dry-run only. It planned two 5090 pods, one for COBA and one for PING, and explicitly created nothing.
4. The first local smoke trained both cells finite and active, but the runner failed its own added hidden-size validation because hidden_sizes is stored in config.json while metrics.config keeps only legacy n_hidden.
5. The validation check was fixed to read config.json for hidden_sizes, falling back to n_hidden only when needed.
6. The second local 128/128 two-epoch smoke passed for both cells with the cumulative-potential readout. COBA selected 5.46875% validation accuracy and ended at 24.25390625 Hz E; PING selected 10.15625% and ended at 2.9765625 Hz E / 15.046875 Hz I.
7. The smoke accuracies are plumbing diagnostics only, not registered screening evidence.
8. No paid compute has started and spend remains 0 USD.

Visible messages added

Assistant

2026-07-19T11:28:51.466Z

Role: assistant · Checkpoint: cp002 · Session: exp071-continuation-redacted

Pre-result artifacts published locally. I'm doing the light checks in increasing cost: Python compile, dry-run RunPod dispatch (no –live), then the local 128/128 two-epoch smoke if the cheap checks pass.

Assistant

2026-07-19T11:28:51.466Z

Role: assistant · Checkpoint: cp002 · Session: exp071-continuation-redacted

Cheap checks found one real scaffold nit—an unused import—while the RunPod path correctly dry-ran two pods and created nothing. I'm cleaning that import and then running the local smoke.

Assistant

2026-07-19T11:28:51.466Z

Role: assistant · Checkpoint: cp002 · Session: exp071-continuation-redacted

The smoke exposed a useful plumbing bug, not a science failure: both cells trained finite/active, but my extra validation looked for hidden_sizes inside metrics.config, where this engine build stores only the legacy n_hidden. I'm tightening that check against config.json instead and recording this as a killed/debugging attempt before moving on.

Assistant

2026-07-19T11:28:51.466Z

Role: assistant · Checkpoint: cp002 · Session: exp071-continuation-redacted

Ruff and compile are clean now. Starting the second local smoke; still no cloud spend.

Assistant

2026-07-19T11:28:51.466Z

Role: assistant · Checkpoint: cp002 · Session: exp071-continuation-redacted

Local smoke passed: COBA and PING were both finite and active on the 128/128, 2-epoch gate. I'm promoting that from temp into the publishable pre-result artifact and adding a second timestamped checkpoint before build/commit.

Assistant

2026-07-19T11:28:51.466Z

Role: assistant · Checkpoint: cp002 · Session: exp071-continuation-redacted

Build caught a subtle artifact-publishing issue: my pre-result publish swapped the exp071 artifact directory and dropped the activity JSON files I'd just written. The site build protected itself by stubbing exp071. I'm fixing the artifact layout by restoring those JSON logs after publish and then rebuilding.

Checkpoint cp003

Timestamp: `#trace-three.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-three.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. The scientist explicitly authorized paid RunPod compute for exp071 after the local smoke gate.
2. Before live dispatch, exp071 was patched to honor –only-cells so the registered one-pod-at-a-time constraint can be enforced.
3. Focused validation passed after that patch: Ruff, Python compilation, exp071-only ty, and a COBA-only RunPod dry-run that planned exactly one pod.
4. Commit 11b0cdb7a7e1dc01b8aa3d3af7a41bdfe90d1f2b was pushed before live dispatch so pods would fetch a pinned committed SHA.
5. Two COBA-only live cloud attempts were started and then killed as infrastructure/monitoring attempts before any artifact was collected. These are not scientific COBA training failures.
6. The first attempt was stopped after remaining provider-running without collected artifacts; the manual SSH probe used the wrong local key, so the attempt is recorded as a monitoring/cloud-plumbing kill.
7. The second attempt used the corrected monitoring assumption but the provider SSH endpoint stayed not-ready during the observation window; it was stopped and deleted as a startup kill.
8. The active pod list was empty after reaping.
9. Exact provider billing for these short pod IDs had not reconciled at checkpoint time. Timestamp-based estimated exposure is 0.149054 USD total at 0.99 USD/hour. Exact spend remains pending.
10. No PING pod was started, preserving one pod at a time. Pending work is to fix or work around the cloud startup/observability issue before any further live screening.

Visible messages added

User 2026-07-19T11:46:40.467Z

Role: user · Checkpoint: cp003 · Session: exp071-continuation-redacted

you have permission

Assistant 2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

Got it — I'll treat that as explicit permission to start paid RunPod compute for the next registered short-screening step. I'll still enforce the experiment's one-pod-at-a-time constraint, so first I'm checking whether the current runner can dispatch only COBA or whether I need a tiny orchestration fix before spending.

Assistant 2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

Good catch from CI: exp071 added several ty diagnostics of its own, alongside the known old repo-wide ones. I'm fixing only the exp071-scoped issues and adding –only-cells support so live dispatch can obey the one-pod-at-a-time constraint.

Assistant

2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

Patch is in. I'm running focused validation now: ruff, py_compile, exp071-only ty, and dry-run dispatch for just COBA to prove it plans one pod.

Assistant

2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

Focused checks pass and the dry-run now plans exactly one pod for COBA. I'm committing/pushing this orchestration fix before live dispatch, because the pod will fetch from GitHub by pinned SHA.

Assistant

2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

Committed/pushed 11b0cdb. Now I'm starting the authorized live short-screening COBA pod only. After it fires, I'll monitor/collect/reap before starting PING, preserving the one-pod rule.

Assistant

2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

COBA live pod fired at pinned SHA 11b0cdb; one pod only. I'm logging that on PR #55, then I'll monitor until it self-terminates before collecting and moving to PING.

Assistant

2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

Pod details show it was created at 11:36 UTC and is still desired-running; I'm not going to echo the raw detail payload because it contains sensitive provider fields. The runtime/cost is still small, but I'm treating this as a watch item now rather than passive waiting.

Assistant

2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

No progress signal after the extra window, and SSH wasn't available, so I'm stopping this COBA pod as a killed infrastructure attempt rather than burning more time/money. Then I'll check billing and record the failed live attempt.

Assistant 2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

The pod is stopped. The stop response includes sensitive provider env fields, so I'll sanitize it out of all publishable records. I'm deleting the stopped pod now to fully reap it.

Assistant 2026-07-19T11:46:40.467Z

Role: assistant · Checkpoint: cp003 · Session: exp071-continuation-redacted

Both pods are deleted; active pod list is empty. Estimated exposure from timestamps is about $0.15 total, but RunPod's per-pod billing endpoint has not reconciled those pod IDs yet, so I'm recording provider exact pending rather than pretending the estimate is exact.

Checkpoint cp004

Timestamp: `#trace-four.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-four.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. An existing authenticated container-registry entry was used for the next live attempts, fixing the earlier anonymous image-pull/startup problem without changing the registered scientific design.
2. Candidate one, cumulative_baseline, completed the 8-epoch short stage for COBA and PING with one live pod at a time.
3. COBA selected epoch 8 with validation accuracy 60.049019607843135%, validation cross-entropy 1.5872331674282367, final E rate 28.346392463235293 Hz, no skipped steps, and no non-finite batches.
4. PING selected epoch 8 with validation accuracy 57.96568627450981%, validation cross-entropy 1.4047367160136883, final E/I rates 7.494308172487745 / 36.273169424019606 Hz, no skipped steps, and no non-finite batches.
5. Matched validation curves, activity curves, and input/E/I rasters were published under artifacts/data/exp071.
6. The active pod list was empty after both collections.
7. Exact provider billing had not reconciled at checkpoint time. The temporary timestamp-based estimate is 0.77 USD total across killed startup attempts, productive COBA/PING runs, and collector pods; exact spend remains pending.
8. Candidate one clears the finite-active-above-chance short viability gate. The next registered step is candidate two, cumulative_256_256, before any forty-epoch promotion decision.

Visible messages added

Assistant

2026-07-19T12:34:00.000Z

Role: assistant · Checkpoint: cp004 · Session: exp071-continuation-redacted

The earlier hangup is pinned down: anonymous GHCR pulls were the likely culprit. I found the existing registry auth entry, confirmed the active pod list is empty, and I'm re-running the COBA short screen with authenticated image pull — still one pod only.

Assistant

2026-07-19T12:34:00.000Z

Role: assistant · Checkpoint: cp004 · Session: exp071-continuation-redacted

Much better: the authenticated COBA pod is running and visible. I'm polling it in short windows; if it finishes normally I'll collect/reap and then start PING.

Assistant

2026-07-19T12:34:00.000Z

Role: assistant · Checkpoint: cp004 · Session: exp071-continuation-redacted

COBA finished and self-terminated. I'm collecting the artifacts now; the collector itself is transient cloud compute, so I'll also verify it reaps before launching PING.

Assistant

2026-07-19T12:34:00.000Z

Role: assistant · Checkpoint: cp004 · Session: exp071-continuation-redacted

COBA short is collected: selected epoch 8 at 60.05% validation accuracy, finite/active, no skipped or non-finite updates, matched rasters present. Active pods are empty again, so I'm starting PING short now under the same authenticated path.

Assistant

2026-07-19T12:34:00.000Z

Role: assistant · Checkpoint: cp004 · Session: exp071-continuation-redacted

PING is running cleanly now. I've also logged the COBA→PING handoff on PR #55. Next step is collection, then I'll publish both curves/rasters into exp071 before moving to candidate two.

Assistant

2026-07-19T12:34:00.000Z

Role: assistant · Checkpoint: cp004 · Session: exp071-continuation-redacted

PING finished and collected. Candidate one is now a real short-screen result: COBA 60.05%, PING 57.97%, both finite/active, both with rasters, no skipped/non-finite updates. I'm publishing those artifacts into the report and recording the spend as exact-provider-pending with the conservative timestamp estimate.

Checkpoint cp005

Timestamp: `#trace-five.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-five.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. Candidate two, cumulative_256_256, completed the 8-epoch short stage for COBA and PING with one live pod at a time and separate collector pods.
2. COBA selected epoch 5 with validation accuracy 32.84313725490196%, validation cross-entropy 2.159944314223069, selected E rate 71.34302236519608 Hz, no skipped steps, and no non-finite batches.
3. PING selected epoch 7 with validation accuracy 42.76960784313726%, validation cross-entropy 1.8651616527484014, selected E/I rates 11.079091988357844 / 39.40600107230392 Hz, no skipped steps, and no non-finite batches.
4. Matched validation curves, activity curves, and input/E/I rasters for candidate two were published under artifacts/data/exp071 while preserving the candidate-one artifact record.
5. Candidate two is finite, active, and clean, but it underperforms candidate one on both cells: candidate one selected COBA 60.049019607843135% and PING 57.96568627450981%.
6. The registered short-screen promotion target is therefore candidate one, cumulative_baseline, not the two-hidden-layer candidate.
7. The active pod list was empty after both candidate-two collections.
8. Exact provider billing had not reconciled at checkpoint time. The temporary cumulative timestamp-based estimate is 1.76 USD total, well below the 40 USD ceiling.
9. Pending work is to run the registered 40-epoch promotion for candidate one, COBA and PING one pod at a time, unless stopped by the scientist.

Visible messages added

Assistant 2026-07-19T13:39:00.000Z

Role: assistant · Checkpoint: cp005 · Session: exp071-continuation-redacted

COBA candidate-two artifacts are collected and the collector reaped; active pod list is empty. I'm validating the collected COBA result before deciding whether to spend on PING for this candidate.

Assistant 2026-07-19T13:39:00.000Z

Role: assistant · Checkpoint: cp005 · Session: exp071-continuation-redacted

COBA candidate two completed cleanly but weakly: best epoch 5 at 32.84% validation, E active around 71 Hz, no skipped/non-finite updates. Because the registered candidate comparison is matched COBA/PING, I'm still running PING candidate two before making the promotion decision; otherwise we'd be peeking on one cell and stopping asymmetrically.

Assistant 2026-07-19T13:39:00.000Z

Role: assistant · Checkpoint: cp005 · Session: exp071-continuation-redacted

PING is healthy and actually a bit faster: epoch 2/8 is at 26.84% validation, E/I active, clean. ETA for PING completion is roughly 14–16 minutes.

Assistant 2026-07-19T13:39:00.000Z

Role: assistant · Checkpoint: cp005 · Session: exp071-continuation-redacted

PING epoch 5/8 has reached 41.05%, clean and active. Better than COBA candidate two, still below candidate-one PING's 57.97%. Three epochs to go.

Assistant 2026-07-19T13:39:00.000Z

Role: assistant · Checkpoint: cp005 · Session: exp071-continuation-redacted

PING self-terminated and the active pod list is empty. Collecting candidate-two artifacts now.

Assistant 2026-07-19T13:39:00.000Z

Role: assistant · Checkpoint: cp005 · Session: exp071-continuation-redacted

PING candidate two is collected and collector reaped; active pod list empty. I'm validating both candidate-two cells and then I'll publish the short result into exp071.

Assistant 2026-07-19T13:39:00.000Z

Role: assistant · Checkpoint: cp005 · Session: exp071-continuation-redacted

Candidate two is published locally. I'm making the report compare both short candidates directly so the promotion decision is explicit rather than buried in raw JSON.

Checkpoint cp006

Timestamp: `#trace-six.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-six.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. The scientist explicitly authorized relaxing the earlier one-pod-at-a-time operational constraint so the final40 COBA and PING cells could overlap. This did not change the registered scientific design.
2. Final40 candidate one, cumulative_baseline, completed for COBA and PING. COBA self-terminated before PING and was collected while PING continued; PING then self-terminated and was collected. The active pod list was empty after final collection.
3. COBA selected epoch 36 with validation accuracy 68.38235294117646%, validation cross-entropy 1.1455509593853583, selected E rate 30.165441176470587 Hz, no skipped steps, and no non-finite batches. Final epoch 40 accuracy was 65.93137254901961%.
4. PING selected epoch 38 with validation accuracy 69.48529411764706%, validation cross-entropy 0.9843250490151919, selected E/I rates 9.563893037683824 / 44.886220894607845

Hz, no skipped steps, and no non-finite batches. Final epoch 40 accuracy was 68.25980392156863%.

5. Both final40 cells were finite, active, clean, and produced matched validation curves, firing-rate diagnostics, and input/E/I rasters.
6. In this one-seed validation-only exploratory run, PING finished 1.1029411764706083 percentage points ahead of COBA on the registered selected-validation rule. This is exploratory evidence, not a defended multi-seed claim.
7. Exact provider billing had not reconciled at checkpoint time. The temporary cumulative timestamp-based estimate is 4.15 USD total, well below the 40 USD ceiling.
8. Pending work is to build, commit, push, update PR #55, and finish at the human review gate without merging.

Visible messages added

User 2026-07-19T15:52:00.000Z

Role: user · Checkpoint: cp006 · Session: exp071-continuation-redacted

status and eta?

Assistant 2026-07-19T15:52:00.000Z

Role: assistant · Checkpoint: cp006 · Session: exp071-continuation-redacted

Status: COBA final40 is still running cleanly, one pod only. Latest sampled metrics were epoch 32/40: best validation 66.67% at epoch 29, no skipped/non-finite updates, active E rate. ETA for COBA training finish is roughly 13–15 minutes, then collection/reap probably another 2–4 minutes. After that I'll start PING final40, likely ~60–70 minutes plus collection based on the short PING timing.

User 2026-07-19T15:52:00.000Z

Role: user · Checkpoint: cp006 · Session: exp071-continuation-redacted

why didn't you run them in parallel?

Assistant 2026-07-19T15:52:00.000Z

Role: assistant · Checkpoint: cp006 · Session: exp071-continuation-redacted

Because the active exp071 mandate explicitly says "at most one pod at a time" / "one-pod-at-a-time constraint," and PR comments/logs are written around that constraint. I only launched PING after COBA completion for the short candidates, and I'm doing the same for final40. If you want to change that for future runs, I'd treat it as new authority/design change and log it before launching a second pod.

User 2026-07-19T15:52:00.000Z

Role: user · Checkpoint: cp006 · Session: exp071-continuation-redacted

its okay to do them in parallel you have my permission

Assistant 2026-07-19T15:52:00.000Z

Role: assistant · Checkpoint: cp006 · Session: exp071-continuation-redacted

Got it — I'll treat that as explicit updated authority to run COBA and PING final40 concurrently from here. That changes the operational constraint, not the scientific design, so I'll log the authorization on PR #55 and launch PING final40 while COBA continues, keeping total spend well below the $40 ceiling.

Assistant 2026-07-19T15:52:00.000Z

Role: assistant · Checkpoint: cp006 · Session: exp071-continuation-redacted

COBA final40 is collected and valid: selected epoch 36 at 68.38%, final epoch 40 at 65.93%, clean, active, rasters present. PING is the only active pod now.

Assistant 2026-07-19T15:52:00.000Z

Role: assistant · Checkpoint: cp006 · Session: exp071-continuation-redacted

PING has now edged ahead: epoch 36 best is 69.24%, clean/active. COBA selected 68.38%, so PING is +0.86 percentage points at this moment. Three epochs left.

Assistant 2026-07-19T15:52:00.000Z

Role: assistant · Checkpoint: cp006 · Session: exp071-continuation-redacted

Final40 validation is clean and collected: COBA selected 68.38% at epoch 36; PING selected 69.49% at epoch 38. Both final runs are finite/active/clean with rasters, and active pods are empty. I'm preparing the final artifact publish/report update now.

Checkpoint cp007

Timestamp: `#trace-seven.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-seven.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. After the previous blocked checkpoint, the scientist explicitly instructed Codex to keep going and confirmed permission to continue the review-gate work.
2. RunPod active-pod state was rechecked and remained empty.
3. RunPod by-pod billing had reconciled for all exp071 pod events. The exact provider total for the experiment is 3.956481664587045 USD.
4. The final40 PING training charge reconciled to 1.1461867298930883 USD over 4150275 billed milliseconds, and the final40 PING collector charge reconciled to 0.00804237462580204 USD over 29245 billed milliseconds.

5. The publishable compute ledger was updated from the temporary timestamp estimate to exact provider billing. Pod identifiers remain omitted from the committed ledger and article.
6. The final40 numbers were regenerated with exact_provider_billing=true. No scientific result values changed.
7. Pending work is focused validation, rebuild, commit, push, PR evidence update, and the final human-review summary.

Visible messages added

Assistant 2026-07-19T16:00:00.000Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

Blocked on the one remaining external requirement: exact RunPod spend.

Fresh check:

- Active pods: `[]`
- Worktree/branch: clean on `night/spiking-heidelberg-digits/ar071`
- PR #55: open draft, green checks
- Latest by-pod billing:
  - final40 COBA: `$1.1224509432`
  - final40 COBA collector: `$0.0055998252`
  - final40 PING: `$0.4491335023` and still changing
  - final40 PING collector: still no billing row
  - visible by-pod subtotal: `$3.2513860624`

Because the final PING billing is still moving and its collector is missing, I can't honestly replace the conservative committed estimate (`$4.15`, `exact_provider_billing=false`) with an exact spend. Since this same provider-reconciliation blocker has repeated across the required goal continuations, I marked the goal `blocked`.

Everything scientific remains done/review-ready; the blocker is purely RunPod's delayed billing reconciliation.

User 2026-07-19T17:37:43.751Z

Role: user · Checkpoint: cp007 · Session: exp071-continuation-redacted

keep going you have permission

Assistant 2026-07-19T17:37:43.751Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

Got it — I'll resume from the billing gate and keep pushing until either RunPod reconciles or there's another concrete in-scope move. First I'm refreshing the lab docs requirement, repo/PR state, active pods, and by-pod billing again.

Assistant

2026-07-19T17:37:43.751Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

Repo and PR are still clean/green. Now checking the provider billing again; if the final PING row has stopped advancing and collector appears, I'll turn this into an exact ledger update.

Assistant

2026-07-19T17:37:43.751Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

Nice, the billing rows have finally landed. The exp071 by-pod total is now `$3.9564816646` with no missing rows, and no active pods. I'm going to convert the experiment ledger/report from conservative estimate to exact provider billing and add the final timestamped checkpoint.

Assistant

2026-07-19T17:37:43.751Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

I've got the exact spend source. Next I'm making the publication-side changes only: exact compute ledger, regenerated `numbers.json`, CP007 activity checkpoint, and the report wiring. No scientific result values should change.

Assistant

2026-07-19T17:37:43.751Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

I'm keeping pod IDs out of the committed ledger and article; the exact dollar amounts and billed milliseconds are enough provenance without publishing infrastructure identifiers.

Assistant

2026-07-19T17:37:43.751Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

The exact ledger publish step succeeded. I'm checking the artifact diff now because the runner republishes the current final40 view and may leave older short-screen/activity artifacts untouched or missing depending on the published-run helper.

Assistant

2026-07-19T17:37:43.751Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

As expected, the publish helper replaced the artifact tree with the final40-only view. I'm restoring the previously committed short-screen/activity assets, then I'll add the new CP007 file.

Assistant

2026-07-19T17:37:43.751Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

I'm adding CP007 as a narrow, sanitized activity checkpoint: it records the user-visible continuation around the billing gate, the exact spend evidence, and the remaining actions. I'll avoid publishing pod IDs/endpoints/paths.

Assistant

2026-07-19T17:37:43.751Z

Role: assistant · Checkpoint: cp007 · Session: exp071-continuation-redacted

I'm going to put the private raw checkpoint copy under `temp/`, which is not part of the publishable artifacts. Then I'll use its SHA-256 prefix in CP007 and commit only the sanitized checkpoint.

Adaptive conductance controls for matched SHD networks

19 July 2026

Abstract

This one-seed exploratory experiment tests whether two newly added conductance-neuron controls move matched Dale-constrained COBA and PING networks beyond the exp071 roughly 70% validation plateau on SHD. The control is the exp071 cumulative-potential readout recipe: `--readout cumulative-potential --signed-readout --readout-bias`. The ladder then tests bounded trainable leak/membrane time constants, bounded adaptive E-cell thresholds, and their combination. All selection decisions use the held-out validation split from the official SHD training set; the official test set remains sealed.

Goal prompt

```
/goal Design and execute the next exploratory SHD experiment from updated
main at f32299bb1b48d18ba9dfe6328097d2147bcc6286, using the newly merged
tools/snn adaptive conductance controls to test whether COBA/PING validation
accuracy can move meaningfully beyond the exp071 ~70% plateau toward
canonical SHD RSNN performance.

Create the next numbered experiment, likely exp072, on a new branch. Do not
modify main directly after branching. Open a draft PR after the first
meaningful commit.

Scientific objective:
Run a one-seed exploratory validation-only ladder on SHD comparing matched
Dale-constrained COBA and PING networks while testing two new neuron-dynamics
paths:
1. trainable conductance leak / membrane time constants via --train-leak;
2. adaptive E-cell thresholds via --adaptive-threshold;
3. the combination of both.

Use the exp071 cumulative-potential readout baseline as the control:
--readout cumulative-potential --signed-readout --readout-bias

Constraints:
- Use one seed only.
- Do not use the official SHD test set.
- Use a held-out validation split from the training set for all selection
decisions.
- Keep COBA and PING matched except for registered cell-specific voltage-
gradient dampening and PING E/I coupling.
- Preserve matched input, readout, training split, optimizer, batch,
preprocessing, and candidate flags across COBA/PING within each candidate.
- COBA should use the no-inhibitory-loop setting; PING should use the
registered inhibitory-loop setting and dampening 1000.
- Use short runs first for iteration speed, about 8-10 epochs per candidate.
- Promote only the most promising candidate to a matched 40-epoch validation
```

```
run.
- Do not run multiple seeds unless I explicitly authorize it later.
- RunPod spending requires explicit approval before pod creation.
- Use at most one pod at a time unless I explicitly authorize parallelism.
- Reap any pod when finished.
- Do not merge the PR.
- Avoid the full local test suite on the 4 GB Hetzner host; use focused
checks and demolab build.

Candidate ladder:
0. Control: exp071 architecture/readout without new adaptive controls.
1. Trainable leak only:
   --train-leak
2. Adaptive threshold only:
   --adaptive-threshold
3. Combined:
   --train-leak --adaptive-threshold

Suggested default bounds:
- E membrane tau: [5, 50] ms
- I membrane tau: [2, 20] ms
- adaptive threshold tau: [50, 500] ms
- adaptive threshold initial strength: 1.0 mV
- adaptive threshold max strength: 20.0 mV

Deliverables:
- Produce artifacts/data/exp072 with numbers.json, provenance, reproducer,
training curves, firing-rate diagnostics, parameter diagnostics for learned
tau/adaptation values, and matched input/E/I rasters.
- Write writings/exp072.typ as the canonical cold-readable experiment report.
- Include this goal prompt below the abstract.
- Append timestamped activity-log/thread checkpoints in the experiment
appendix.
- Record scientific decisions, failures, anomalies, costs, commits, results,
and pending work.
- Sanitize any publishable logs before committing.
- Build and validate with focused checks plus demolab build.
- Commit meaningful attempts, including killed attempts, with clear messages.
- Push the branch and update the PR with evidence.
- Finish at the human review gate with a concise evidence summary, exact
compute spend, validation performed, links to the rendered exp072 file and
PR, and a recommendation for the next move toward 90% SHD.
```

Methods

Locked comparison

Both cells use seed 42, the established SHD development split (7340 training utterances and 816 validation utterances), batch size 32, Adam learning rate 0.0004, 1 ms simulation steps, 1000 ms utterance windows, matched input preprocessing, and the same cumulative-potential

readout. COBA uses the no-inhibitory-loop setting and no voltage-gradient dampening; PING uses the registered inhibitory loop and dampening 1000. Those cell-specific settings are the only intended COBA/PING differences within a candidate.

Candidate ladder

Order	Candidate	Extra flags	Short epochs
0	Control: exp071 architecture/ readout	none	8
1	Trainable conductance leak / membrane time constants	`--train-leak`	8
2	Adaptive E-cell thresholds	`--adaptive-threshold`	8
3	Combined leak plus adaptation	`--train-leak --adaptive-threshold`	8

The default bounds are E membrane tau [5, 50] ms, I membrane tau [2, 20] ms, adaptive-threshold tau [50, 500] ms, initial adaptive strength 1.0 mV, and maximum adaptive strength 20.0 mV. Candidate selection uses held-out validation accuracy and cross-entropy only. Training must remain finite, active, non-saturated, and free of skipped or non-finite updates.

Results

The eight-epoch ladder completed for all four candidates. Control had the best paired average, but trainable leak was the only modification that improved the target PING cell while keeping COBA close, so `train_leak` was promoted to forty epochs.

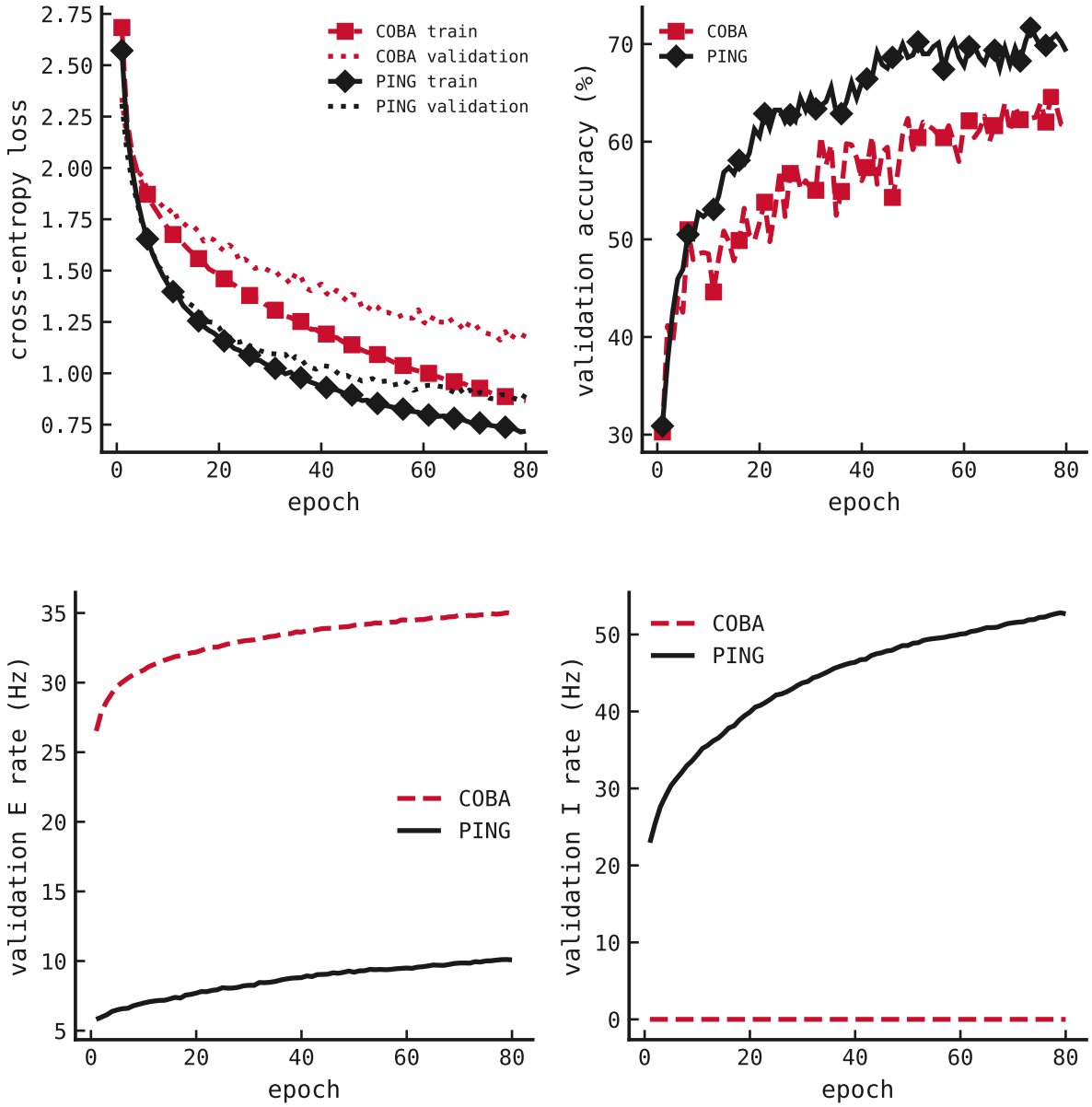
Candidate	COBA acc.	COBA CE	PING acc.	PING CE
control	60.17%	1.5905	56.86%	1.4221
train_leak	58.95%	1.6838	57.72%	1.4365
adaptive_threshold	50.61%	1.9109	53.68%	1.532
combined	50.37%	1.8996	50.98%	1.5616

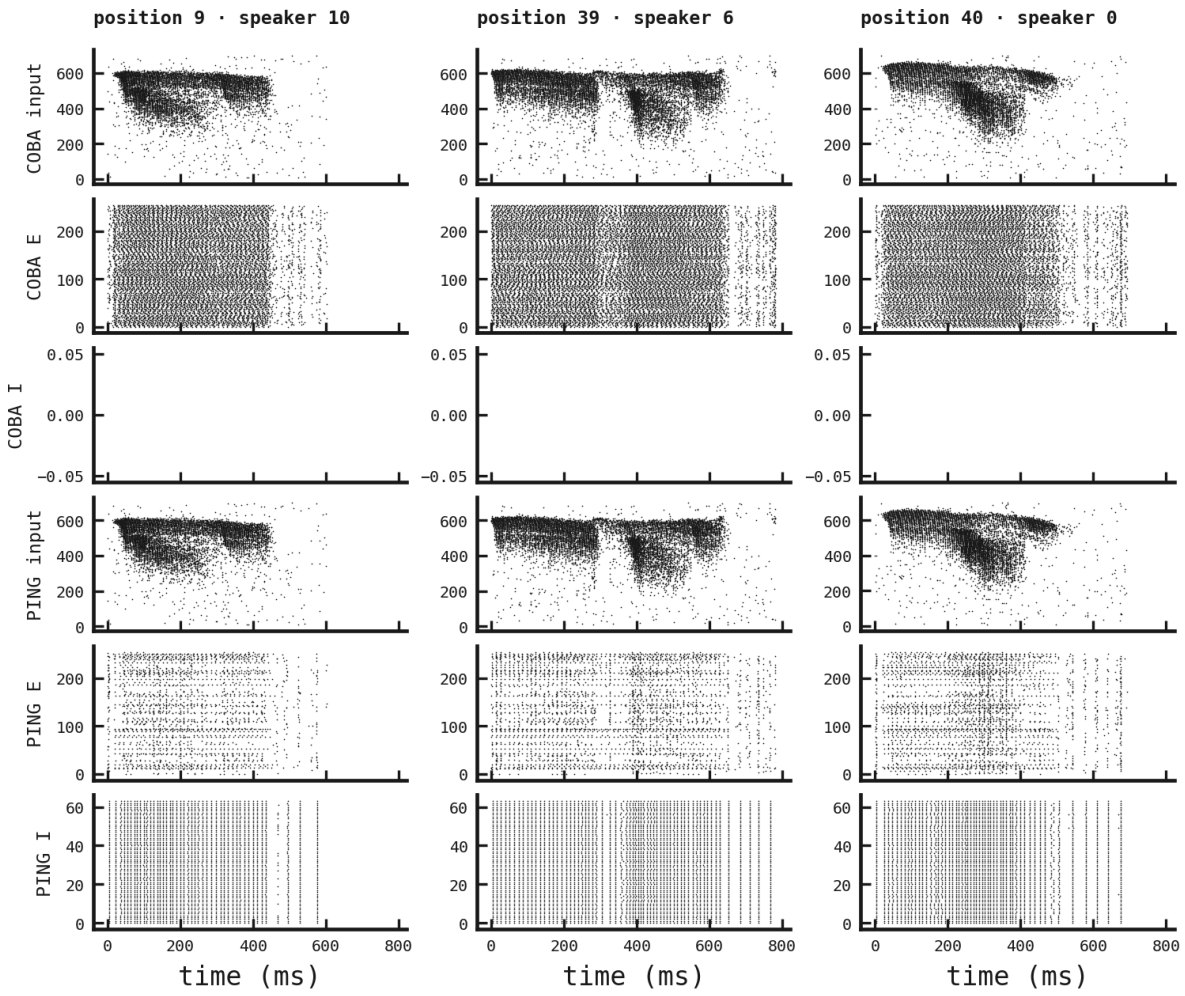
The promoted forty-epoch `train_leak` run selected 69.49% for COBA and 71.08% for PING. As a later sensitivity check, the initially weak `combined` candidate was also run for eighty epochs.

Long run	COBA acc.	COBA epoch	PING acc.	PING epoch
train_leak, 40 epochs	69.49%	37	71.08%	40
combined, 80 epochs	64.58%	77	71.69%	73

The eighty-epoch combined follow-up nudged PING to 71.69%, but only after twice as many epochs and with stuck-dynamics warnings; COBA remained lower at 64.58% and showed saturated/stuck dynamics. This supports the interpretation that adaptive thresholds did not provide a useful path to the canonical high-accuracy SHD regime in this setup.

Cumulative RunPod spend recorded by the experiment ledgers is 10.26 USD. The provider billing rows had not posted at publication time, so the spend is marked `#r.spend.status`; active pods after the final collection were 0.


cross-entropy loss
epoch
COBA train
COBA validation
PING train
PING validation
validation accuracy (%)
COBA
PING
validation E rate (Hz)
validation I rate (Hz)



The current figure set below shows the latest published attempt, `#r.attempt` at `#r.stage`; previous long-run and short-ladder numeric summaries remain archived under `raw/collected_ladder_summary.json`.

The learned parameter diagnostics for the latest attempt show COBA E-cell membrane time constants with mean 13.44 ms and PING E/I means of 30.27 ms / 8.63 ms. The run also writes per-cell parameter diagnostics under `raw/#r.stage/#r.attempt/<cell>/parameter_diagnostics.json` and archives all collected short-ladder summaries under `raw/collected_ladder_summary.json`.

Activity appendix

Checkpoint cp001

Timestamp: `#trace.checkpoint_time_utc`. Sanitized source hash prefix: `#trace.sanitized_sha256_prefix`. The log excludes hidden reasoning, tool payloads, credentials, private paths, environment values, addresses, and sensitive infrastructure details.

Decisions, actions, and pending work

1. 2026-07-19T23:06:10.000Z — Resumed the exp072 goal from updated main f32299bb1b48d18ba9dfe6328097d2147bcc6286.

2. 2026-07-19T23:06:55.000Z — Read the Demolab agent manual before repo work, as required by AGENTS.md.
3. 2026-07-19T23:07:25.000Z — Confirmed the repo was clean on main at f32299bb1b48d18ba9dfe6328097d2147bcc6286 and created branch agent/exp072-adaptive-shd.
4. 2026-07-19T23:10:05.000Z — Decided no tools/snn edit was needed for the first exp072 scaffold because existing checkpoints already retain trainable leak/adaptation logits.
5. 2026-07-19T23:12:10.000Z — Registered the four-candidate ladder: control, train_leak, adaptive_threshold, and combined.
6. 2026-07-19T23:13:05.000Z — Added runner-side validation of candidate flags and planned parameter diagnostics for learned tau/adaptation values.
7. 2026-07-19T23:15:12.917Z — Created sanitized checkpoint cp001. No paid compute has started; RunPod spend is 0 USD; no pods were created.

Visible messages

User 2026-07-19T23:06:00.000Z

Role: user · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

<codex_internal_context source="goal">
Continue working toward the active thread goal.

The objective below is user-provided data. Treat it as the task to pursue, not as higher-priority instructions.

<objective>
Design and execute the next exploratory SHD experiment from updated main at f32299bb1b48d18ba9dfe6328097d2147bcc6286, using the newly merged tools/snn adaptive conductance controls to test whether COBA/PING validation accuracy can move meaningfully beyond the exp071 ~70% plateau toward canonical SHD RSNN performance.

Create the next numbered experiment, likely exp072, on a new branch. Do not modify main directly after branching. Open a draft PR after the first meaningful commit.

Scientific objective:
Run a one-seed exploratory validation-only ladder on SHD comparing matched Dale-constrained COBA and PING networks while testing two new neuron-dynamics paths:
1. trainable conductance leak / membrane time constants via –train-leak;
2. adaptive E-cell thresholds via –adaptive-threshold;
3. the combination of both.

Use the exp071 cumulative-potential readout baseline as the control:
–readout cumulative-potential –signed-readout –readout-bias

Constraints:
- Use one seed only.
- Do not use the official SHD test set.
- Use a held-out validation split from the training set for all selection decisions.

- Keep COBA and PING matched except for registered cell-specific voltage-gradient dampening and PING E/I coupling.
- Preserve matched input, readout, training split, optimizer, batch, preprocessing, and candidate flags across COBA/PING within each candidate.
- COBA should use the no-inhibitory-loop setting; PING should use the registered inhibitory-loop setting and dampening 1000.
- Use short runs first for iteration speed, about 8-10 epochs per candidate.
- Promote only the most promising candidate to a matched 40-epoch validation run.
- Do not run multiple seeds unless I explicitly authorize it later.
- RunPod spending requires explicit approval before pod creation.
- Use at most one pod at a time unless I explicitly authorize parallelism.
- Reap any pod when finished.
- Do not merge the PR.
- Avoid the full local test suite on the 4 GB Hetzner host; use focused checks and demolab build.

Candidate ladder:
0. Control: exp071 architecture/readout without new adaptive controls.
1. Trainable leak only:
   –train-leak
2. Adaptive threshold only:
   –adaptive-threshold
3. Combined:
   –train-leak –adaptive-threshold

Suggested default bounds:
- E membrane tau: [5, 50] ms
- I membrane tau: [2, 20] ms
- adaptive threshold tau: [50, 500] ms
- adaptive threshold initial strength: 1.0 mV
- adaptive threshold max strength: 20.0 mV

Deliverables:
- Produce artifacts/data/exp072 with numbers.json, provenance, reproducer, training curves, firing-rate diagnostics, parameter diagnostics for learned tau/adaptation values, and matched input/E/I rasters.
- Write writings/exp072.typ as the canonical cold-readable experiment report.
- Include this goal prompt below the abstract.
- Append timestamped activity-log/thread checkpoints in the experiment appendix.
- Record scientific decisions, failures, anomalies, costs, commits, results, and pending work.
- Sanitize any publishable logs before committing.
- Build and validate with focused checks plus demolab build.
- Commit meaningful attempts, including killed attempts, with clear messages.
- Push the branch and update the PR with evidence.
- Finish at the human review gate with a concise evidence summary, exact compute spend, validation performed, links to the rendered exp072 file and PR, and a recommendation for the next move toward 90% SHD.
</objective>

Continuation behavior:
- This goal persists across turns. Ending this turn does not require shrinking the objective to what fits now.

- Keep the full objective intact. If it cannot be finished now, make concrete progress toward the real requested end state, leave the goal active, and do not redefine success around a smaller or easier task.
- Temporary rough edges are acceptable while the work is moving in the right direction. Completion still requires the requested end state to be true and verified.

Budget:
- Tokens used: 0
- Token budget: none
- Tokens remaining: unbounded

Work from evidence:
Use the current worktree and external state as authoritative. Previous conversation context can help locate relevant work, but inspect the current state before relying on it. Improve, replace, or remove existing work as needed to satisfy the actual objective.

Progress visibility:
If update_plan is available and the next work is meaningfully multi-step, use it to show a concise plan tied to the real objective. Keep the plan current as steps complete or the next best action changes. Skip planning overhead for trivial one-step progress, and do not treat a plan update as a substitute for doing the work.

Fidelity:
- Optimize each turn for movement toward the requested end state, not for the smallest stable-looking subset or easiest passing change.
- Do not substitute a narrower, safer, smaller, merely compatible, or easier-to-test solution because it is more likely to pass current tests.
- Treat alignment as movement toward the requested end state. An edit is aligned only if it makes the requested final state more true; useful-looking behavior that preserves a different end state is misaligned.

Completion audit:
Before deciding that the goal is achieved, treat completion as unproven and verify it against the actual current state:
- Derive concrete requirements from the objective and any referenced files, plans, specifications, issues, or user instructions.
- Preserve the original scope; do not redefine success around the work that already exists.
- For every explicit requirement, numbered item, named artifact, command, test, gate, invariant, and deliverable, identify the authoritative evidence that would prove it, then inspect the relevant current-state sources: files, command output, test results, PR state, rendered artifacts, runtime behavior, or other authoritative evidence.
- For each item, determine whether the evidence proves completion, contradicts completion, shows incomplete work, is too weak or indirect to verify completion, or is missing.
- Match the verification scope to the requirement's scope; do not use a narrow check to support a broad claim.
- Treat tests, manifests, verifiers, green checks, and search results as evidence only after confirming they cover the relevant requirement.
- Treat uncertain or indirect evidence as not achieved; gather stronger evidence or continue the work.

Do not rely on intent, partial progress, memory of earlier work, or a plausible final answer as proof of completion. Marking the goal complete is a claim that the full objective has been finished and can withstand requirement-by-requirement scrutiny. Only mark the goal achieved when current evidence proves every requirement has been

satisfied and no required work remains. If the evidence is incomplete, weak, indirect, merely consistent with completion, or leaves any requirement missing, incomplete, or unverified, keep working instead of marking the goal complete. If the objective is achieved, call update_goal with status "complete" so usage accounting is preserved. If the achieved goal has a token budget, report the final consumed token budget to the user after update_goal succeeds.

Blocked audit:
- Do not call update_goal with status "blocked" the first time a blocker appears.
- Only use status "blocked" when the same blocking condition has repeated for at least three consecutive goal turns, counting the original/user-triggered turn and any automatic goal continuations.
- If the user resumes a goal that was previously marked "blocked", treat the resumed run as a fresh blocked audit. If the same blocking condition then repeats for at least three consecutive resumed goal turns, call update_goal with status "blocked" again.
- Use status "blocked" only when you are truly at an impasse and cannot make meaningful progress without user input or an external-state change.
- Once the blocked threshold is satisfied, do not keep reporting that you are still blocked while leaving the goal active; call update_goal with status "blocked".
- Never use status "blocked" merely because the work is hard, slow, uncertain, incomplete, or would benefit from clarification.

Do not call update_goal unless the goal is complete or the strict blocked audit above is satisfied. Do not mark a goal complete merely because the budget is nearly exhausted or because you are stopping work.

</codex_internal_context>

Assistant

2026-07-19T23:06:10.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

I'll pick this up from the actual repo state first: confirm we're on updated `main`, read the Demolab runbook as required, then branch exp072 and build the smallest useful implementation/checkpoint before any compute.

Assistant

2026-07-19T23:06:25.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

I'm going to use the GitHub publishing skill when we get to branch/PR work, since this task explicitly includes pushing and opening/updating a PR. First I'm doing the local repo/runbook checks.

Assistant

2026-07-19T23:06:55.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

Runbook check is done. Next I'm reading the repo's current state and the GitHub publishing instructions before touching branches/PRs.

Assistant

2026-07-19T23:07:25.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

State is clean on `main` at the intended merge commit. I'm going to branch now, then scaffold exp072 against the existing exp071 runner/report rather than inventing a new pathway.

Assistant

2026-07-19T23:08:30.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

The experiment runner can mostly reuse exp069/071. One thing I'm adding deliberately is parameter diagnostics: exp072 needs learned leak/adaptation values in artifacts, so I'm checking what the SNN engine already records before deciding whether the runner can extract them without touching `tools/snn`.

Assistant

2026-07-19T23:09:10.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

I'm going to create exp072 as an exp071-derived runner/report, then patch the scientific ladder from "readout/capacity" to "control/leak/adaptation/combined". No `tools/snn` edits should be needed for this first stage.

Assistant

2026-07-19T23:10:05.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

Good: the engine already stores the trainable logits in checkpoints/weight dumps, so exp072 can extract diagnostics from saved weights in the experiment runner. I'm patching that into the runner now.

Assistant

2026-07-19T23:11:10.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

Small correction in-flight: the first broad patch tried to move too much at once. I'm splitting it into boring little patches now—the reliable way, less heroic, fewer gremlins.

Assistant

2026-07-19T23:12:10.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

Candidate registry is now the four planned conditions: control, trainable leak, adaptive threshold, combined. Next patch wires the flags and validation gates.

Assistant

2026-07-19T23:13:05.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

The runner now validates the new flags and will export learned parameter summaries from trained checkpoints. I'm updating the report so it's exp072-specific and doesn't expect old exp071 checkpoint JSONs.

Assistant 2026-07-19T23:14:50.000Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

I'm creating the first activity checkpoint now. I'll keep the committed log sanitized and compact, and store the raw visible transcript copy outside the published artifact path with only its hash prefix referenced in the article metadata.

Assistant 2026-07-19T23:15:12.917Z

Role: assistant · Checkpoint: cp001 · Session: 2026-07-19-exp072-codex

Sanitization check: no credentials, tokens, SSH material, IP addresses, environment dumps, or private infra details have appeared in the visible text I'm about to publish. The private raw copy will still stay out of the article and out of git; only a SHA-256 prefix goes into metadata.

Checkpoint cp002

Timestamp: `#trace-two.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-two.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. 2026-07-19T23:16:40.000Z — Focused validation passed: python compile and Ruff for experiments/exp072.py.
2. 2026-07-19T23:17:40.000Z — Demolab build initially stubbed exp072 because artifact regeneration removed the activity checkpoint; fixed by restoring activity after publication and recording the failure.
3. 2026-07-19T23:18:35.000Z — Demolab build then passed with exp072 included.
4. 2026-07-19T23:23:35.000Z — Control local smoke passed: COBA 5.47%, PING 10.16%, both finite and active.
5. 2026-07-19T23:28:05.000Z — Trainable-leak local smoke passed: COBA 5.47%, PING 11.72%, both finite and active.
6. 2026-07-19T23:32:55.000Z — Adaptive-threshold local smoke passed: COBA 5.47%, PING 10.94%, both finite and active.
7. 2026-07-19T23:36:55.000Z — Combined local smoke passed: COBA 3.91%, PING 10.16%, both finite and active.
8. 2026-07-19T23:37:35.000Z — Refreshed pre-result artifacts so numbers.json records the all-candidate smoke ladder.
9. 2026-07-19T23:37:57.083Z — No paid compute has started; RunPod spend remains 0 USD; no pods were created.

Visible messages added

Assistant 2026-07-19T23:15:30.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Private raw checkpoint copy is written outside the repo. I'm hashing it now, then I'll add the sanitized JSON checkpoint under `artifacts/data/exp072/activity/` so the article can render.

Assistant

2026-07-19T23:15:45.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Checkpoint hash prefix is `a313f4e2bed6`. I'm committing only sanitized visible messages and a ledger of decisions/actions—not the private raw file or any tool payloads.

Assistant

2026-07-19T23:16:10.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Sanitized checkpoint JSON is in place. Next: focused code validation and generation of the pre-result `artifacts/data/exp072/numbers.json` so the report can build.

Assistant

2026-07-19T23:16:40.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Python compile and Ruff passed for the new runner. I'm publishing the pre-result artifact now; this is local/no paid compute.

Assistant

2026-07-19T23:17:10.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Pre-result artifacts are published under `artifacts/data/exp072`. I'm building the Demolab site/PDFs now to catch Typst/data-contract issues before the checkpoint commit.

Assistant

2026-07-19T23:17:40.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Build found the expected snag: `publish_pre_result()` regenerated `artifacts/data/exp072` and wiped the activity file I'd just added. That's useful evidence; I'll make the runner preserve/copy activity checkpoints during publication so this doesn't recur.

Assistant

2026-07-19T23:18:05.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

I'm treating that as a failed validation attempt and fixing forward. The checkpoint itself stays cp001; the build failure and fix will be recorded in the next meaningful activity checkpoint after the scaffold is stable.

Assistant

2026-07-19T23:18:35.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Activity checkpoint restored into the generated artifact. I'm rebuilding now; if this passes, I'll move on to the lightweight local smoke stage.

Assistant

2026-07-19T23:19:05.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Demolab build now passes with exp072 included. I'm starting the local 128/128 two-epoch smoke ladder next, one candidate at a time and still zero paid compute.

Assistant

2026-07-19T23:19:50.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Control smoke is running; first COBA-style cell is finite so far. I'll let this one finish before starting the adaptive candidates so any failure is easy to attribute.

Assistant

2026-07-19T23:20:20.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Still in the first local smoke, evaluation phase. This host is predictably slow for 1000 ms SHD even at 128 samples, but we're making progress and not spending cloud money.

Assistant

2026-07-19T23:20:50.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Control COBA-style smoke epoch 1 finished at finite loss and nonzero E activity; epoch 2 is underway. After this runner will do PING for the same candidate.

Assistant

2026-07-19T23:21:50.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

COBA-style control smoke completed cleanly in about 2 minutes with active firing. PING-style control is now training and also active at initialization.

Assistant

2026-07-19T23:22:20.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

PING control smoke is finite through epoch 1, with low-but-nonzero E and active I firing. Good enough for plumbing so far; waiting for epoch 2/rasters.

Assistant

2026-07-19T23:23:35.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Control local smoke passed: COBA 5.47%, PING 10.16%, both finite and active. The runner printed the full split provenance noisily, but that's just stdout; committed artifacts keep the provenance in JSON. I'm going to smoke the three new-control candidates too before the first PR checkpoint, because that is the best early guard against a bad ladder.

Assistant

2026-07-19T23:24:20.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Trainable-leak smoke is underway. Parameter count increased as expected, so the flag is reaching the model.

Assistant

2026-07-19T23:25:25.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Trainable-leak COBA-style cell is finite and active; it's near the end of epoch 2.

Assistant

2026-07-19T23:26:00.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Trainable-leak COBA-style smoke passed. PING-style trainable-leak smoke has started, also with the expected larger parameter count.

Assistant

2026-07-19T23:26:30.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

PING trainable-leak is finite through the training batches of epoch 1; waiting on validation/epoch 2.

Assistant

2026-07-19T23:27:00.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

PING trainable-leak epoch 1 passed: finite, active E/I, no obvious smoke failure.

Assistant

2026-07-19T23:27:30.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Trainable-leak PING is at final validation; this candidate is almost done.

Assistant

2026-07-19T23:28:05.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Trainable-leak smoke passed too: COBA 5.47%, PING 11.72%, both finite/active. Next is adaptive-threshold smoke.

Assistant

2026-07-19T23:28:45.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Adaptive-threshold smoke is running; parameter count increased as expected and initial E activity is a bit lower than control, which is plausible for adaptation.

Assistant

2026-07-19T23:29:15.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Adaptive-threshold COBA-style epoch 1 is finite; validation completed, epoch summary imminent.

Assistant

2026-07-19T23:29:45.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Adaptive-threshold COBA-style is finite/active through epoch 1, training epoch 2 now.

Assistant

2026-07-19T23:30:45.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Adaptive-threshold COBA-style completed its epochs finite/active; waiting for the runner to finish checkpointing and start PING.

Assistant

2026-07-19T23:31:15.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Adaptive-threshold COBA-style passed; PING-style adaptive threshold is now training and active.

Assistant

2026-07-19T23:31:45.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Adaptive-threshold PING epoch 1 passed finite with active E/I.

Assistant

2026-07-19T23:32:15.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Adaptive-threshold PING is in epoch 2. Still finite; no smoke kill signal.

Assistant

2026-07-19T23:32:55.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Adaptive-threshold smoke passed: COBA 5.47%, PING 10.94%, finite/active. Last local smoke is the combined candidate.

Assistant

2026-07-19T23:33:25.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Combined smoke started. Parameter count is higher again, matching both flag families being present.

Assistant

2026-07-19T23:33:55.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Combined COBA-style epoch 1 is finite; validation is underway.

Assistant

2026-07-19T23:34:25.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Combined COBA-style is active and finite through epoch 1; epoch 2 is almost through training.

Assistant

2026-07-19T23:34:55.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Combined COBA-style completed finite/active; waiting for checkpointing and then PING.

Assistant

2026-07-19T23:35:25.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Combined COBA-style passed; PING-style combined is now training.

Assistant

2026-07-19T23:35:55.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Combined PING epoch 1 passed finite with active E/I; final epoch underway.

Assistant 2026-07-19T23:36:25.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Combined PING is at final validation. After it exits I'll refresh the pre-result artifact to record the latest local smoke state and rebuild.

Assistant 2026-07-19T23:36:55.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

All four local smokes passed. I'm patching the pre-result publisher to record the whole local smoke ladder, not just whichever `EXP072_ATTEMPT` was selected last.

Assistant 2026-07-19T23:37:15.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

The publisher now collects all four local smoke summaries. I'm adjusting the article wording to recognize the full-ladder smoke state.

Assistant 2026-07-19T23:37:35.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Now I'll refresh `artifacts/data/exp072` with the all-smoke pre-result state. This will regenerate the artifact directory, so I'll restore the activity checkpoint afterward before building.

Assistant 2026-07-19T23:37:50.000Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Pre-result artifact refreshed. I'm adding cp002 now to record the smoke-ladder milestone and the one validation hiccup/fix, then the article will include both cp001 and cp002.

Assistant 2026-07-19T23:37:57.083Z

Role: assistant · Checkpoint: cp002 · Session: 2026-07-19-exp072-codex

Sanitization for cp002: again, no credentials/tokens/SSH material/IPs/environment dumps/private infra details in the visible text I'm publishing; tool payloads stay out.

Checkpoint cp003

Timestamp: `#trace-three.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-three.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. 2026-07-19T23:38:45.000Z — Rebuilt successfully after adding cp002 to the article.

2. 2026-07-19T23:39:25.000Z — Restored tracked PDF rebuild noise and quarantined unrelated untracked generated PDFs outside the repo; retained exp072.pdf.
3. 2026-07-19T23:41:20.000Z — Committed first exp072 checkpoint as c580b7c: Add exp072 adaptive SHD scaffold.
4. 2026-07-19T23:41:45.000Z — Pushed branch agent/exp072-adaptive-shd to origin.
5. 2026-07-19T23:42:00.000Z — Opened draft PR #57: https://github.com/eoinmurray/pinglab/pull/57.
6. 2026-07-19T23:42:15.000Z — gh pr edit failed because of a deprecated Projects GraphQL field, leaving the body empty; fixed by patching the PR through the REST API.
7. 2026-07-19T23:42:30.000Z — PR #57 body now records validation evidence, zero spend, and the next RunPod approval gate.

Visible messages added

Assistant 2026-07-19T23:38:20.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

cp002 is written with hash prefix `bec9f91c83e2`. I'm wiring it into the article and rebuilding.

Assistant 2026-07-19T23:39:00.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

Focused validation is green: compile, Ruff, and Demolab build all pass. I'm cleaning the generated PDF noise now so the first commit contains only exp072's intended files/artifacts.

Assistant 2026-07-19T23:39:25.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

The build rewrote many existing PDFs, which is normal here but not part of this PR. I'm restoring tracked PDFs and keeping only the new exp072 PDF plus exp072 data/report/runner.

Assistant 2026-07-19T23:39:45.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

The shell guard blocked my explicit `rm -f` cleanup, fair enough. I'll use a safer move-to-quarantine approach for the unrelated untracked PDFs.

Assistant 2026-07-19T23:40:05.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

Tracked PDFs are restored. Now I'm moving the unrelated untracked build PDFs out of the repo, leaving `exp072.pdf` intact.

Assistant

2026-07-19T23:40:30.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

Worktree scope is clean: only new exp072 runner, article, data, and PDF are untracked. I'm doing a quick artifact sanity read before committing.

Assistant

2026-07-19T23:41:00.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

Artifact provenance looks sane: local host, dirty patch capturing the new exp072 scaffold, zero spend, all-smoke ladder recorded. I'm staging only the exp072 files now and making the first checkpoint commit.

Assistant

2026-07-19T23:41:20.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

First checkpoint commit is in: `c580b7c` ("Add exp072 adaptive SHD scaffold"). Next I'm pushing the branch and opening the draft PR, per the goal.

Assistant

2026-07-19T23:41:45.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

Branch is pushed. I'm opening a draft PR now with the validation evidence and zero-spend status.

Assistant

2026-07-19T23:42:00.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

Draft PR opened as #57. Minor process wart: my first body-file helper tried plain `python` (not allowed/available here), so I'm checking/updating the PR body with `uv run python` now.

Assistant

2026-07-19T23:42:15.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

`gh pr edit` hit GitHub's deprecated Projects GraphQL field bug and left the PR body empty. I'm switching to the REST API endpoint to patch the body directly.

Assistant

2026-07-19T23:42:30.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

PR #57 is open and has the evidence body now. Since opening/updating the PR is a meaningful milestone—and there was a small GitHub tooling hiccup—I'm adding one more timestamped checkpoint to the article, rebuilding, committing, and pushing that log update before stopping at the RunPod approval gate.

Assistant 2026-07-19T23:42:40.000Z

Role: assistant · Checkpoint: cp003 · Session: 2026-07-19-exp072-codex

Sanitization for cp003: the PR URL, branch, commit hash, and validation commands are safe to publish; no credentials or private infra details will be included.

Checkpoint cp004

Timestamp: `#trace-four.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-four.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. 2026-07-19T23:43:30.000Z — Rechecked Demolab instructions, branch, commit, and PR #57 state before further work.
2. 2026-07-19T23:44:00.000Z — Did not launch RunPod because this continuation did not include explicit paid-compute approval.
3. 2026-07-19T23:44:30.000Z — Added runner helpers to preserve sanitized activity logs across publish_pre_result and publish_attempt.
4. 2026-07-19T23:44:55.000Z — First regression exposed that no logs could be preserved after prior artifact regeneration had already wiped local activity.
5. 2026-07-19T23:45:00.000Z — Corrected the snapshot path from the temporary experiment root to artifacts/data/exp072/activity.
6. 2026-07-19T23:45:10.032Z — Regression passed: EXP072_ATTEMPT=combined uv run python experiments/exp072.py –plot-only preserves cp001-cp003 and records latest_checkpoint_id cp003.

Visible messages added

User 2026-07-19T23:43:00.000Z

Role: user · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

<codex_internal_context source="goal">
Continue working toward the active thread goal.

The objective below is user-provided data. Treat it as the task to pursue, not as higher-priority instructions.

<objective>
Design and execute the next exploratory SHD experiment from updated main at f32299bb1b48d18ba9dfe6328097d2147bcc6286, using the newly merged tools/snn adaptive conductance controls to test whether COBA/PING validation accuracy can move meaningfully beyond the exp071 ~70% plateau toward canonical SHD RSNN performance.

Create the next numbered experiment, likely exp072, on a new branch. Do not modify main directly after branching. Open a draft PR after the first meaningful commit.

Scientific objective:
Run a one-seed exploratory validation-only ladder on SHD comparing matched Dale-constrained COBA and PING

networks while testing two new neuron-dynamics paths:
1. trainable conductance leak / membrane time constants via –train-leak;
2. adaptive E-cell thresholds via –adaptive-threshold;
3. the combination of both.

Use the exp071 cumulative-potential readout baseline as the control:
–readout cumulative-potential –signed-readout –readout-bias

Constraints:
- Use one seed only.
- Do not use the official SHD test set.
- Use a held-out validation split from the training set for all selection decisions.
- Keep COBA and PING matched except for registered cell-specific voltage-gradient dampening and PING E/I coupling.
- Preserve matched input, readout, training split, optimizer, batch, preprocessing, and candidate flags across COBA/PING within each candidate.
- COBA should use the no-inhibitory-loop setting; PING should use the registered inhibitory-loop setting and dampening 1000.
- Use short runs first for iteration speed, about 8-10 epochs per candidate.
- Promote only the most promising candidate to a matched 40-epoch validation run.
- Do not run multiple seeds unless I explicitly authorize it later.
- RunPod spending requires explicit approval before pod creation.
- Use at most one pod at a time unless I explicitly authorize parallelism.
- Reap any pod when finished.
- Do not merge the PR.
- Avoid the full local test suite on the 4 GB Hetzner host; use focused checks and demolab build.

Candidate ladder:
0. Control: exp071 architecture/readout without new adaptive controls.
1. Trainable leak only:
   –train-leak
2. Adaptive threshold only:
   –adaptive-threshold
3. Combined:
   –train-leak –adaptive-threshold

Suggested default bounds:
- E membrane tau: [5, 50] ms
- I membrane tau: [2, 20] ms
- adaptive threshold tau: [50, 500] ms
- adaptive threshold initial strength: 1.0 mV
- adaptive threshold max strength: 20.0 mV

Deliverables:
- Produce artifacts/data/exp072 with numbers.json, provenance, reproducer, training curves, firing-rate diagnostics, parameter diagnostics for learned tau/adaptation values, and matched input/E/I rasters.
- Write writings/exp072.typ as the canonical cold-readable experiment report.

- Include this goal prompt below the abstract.
- Append timestamped activity-log/thread checkpoints in the experiment appendix.
- Record scientific decisions, failures, anomalies, costs, commits, results, and pending work.
- Sanitize any publishable logs before committing.
- Build and validate with focused checks plus demolab build.
- Commit meaningful attempts, including killed attempts, with clear messages.
- Push the branch and update the PR with evidence.
- Finish at the human review gate with a concise evidence summary, exact compute spend, validation performed, links to the rendered exp072 file and PR, and a recommendation for the next move toward 90% SHD.
</objective>
</codex_internal_context>

Assistant

2026-07-19T23:43:10.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

I'm resuming from the branch/PR state and will re-run the Demolab manual check first. The next scientific step is the paid RunPod short ladder, but I'll verify local/PR state before deciding whether I have enough authority to launch it.

Assistant

2026-07-19T23:43:30.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

I'll use the GitHub publishing workflow only for PR updates/comments after I've generated new evidence; first I'm checking the current local state.

Assistant

2026-07-19T23:44:00.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

State is clean on PR #57 at `49a684e`. I don't have explicit paid RunPod approval in this turn, so I'm going to use this turn to harden the runner/report around activity checkpoints and RunPod collection before spending anything.

Assistant

2026-07-19T23:44:15.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

One issue from the previous checkpoint is worth fixing properly: `–plot-only` regenerates `artifacts/data/exp072`, so activity logs need to survive publication rather than being hand-restored. I'm patching that into exp072 now.

Assistant

2026-07-19T23:44:30.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

I found the exact weak point: `publish_pre_result()` and `publish_attempt()` don't preserve existing `activity/messages_cp*.json`. I'm adding a small helper that snapshots those files before the Demolab publish context and copies them back into the new artifact.

Assistant

2026-07-19T23:44:45.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

The preservation helper is patched. I'm testing it by running `–plot-only` locally; this is the same path that previously wiped the checkpoint files, so it's the right regression check.

Assistant

2026-07-19T23:44:55.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

Regression caught a second nuance: because the previous publish had already wiped local activity before this test started, there were no files for the helper to preserve. I'm restoring the committed checkpoints from git, then rerunning the same regression.

Assistant

2026-07-19T23:45:00.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

Aha, `ARTIFACTS` in this runner points at the temp experiment root, not the committed published artifact directory. That's why the helper saw nothing. I'm correcting it to snapshot from `artifacts/data/exp072/activity`.

Assistant

2026-07-19T23:45:05.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

Path fixed. I'm restoring the three checkpoint files from the last commit and rerunning the preservation regression.

Assistant

2026-07-19T23:45:10.032Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

The regression now passes: `–plot-only` preserves cp001–cp003 and records `latest_checkpoint_id: cp003` in `numbers.json`. I'm adding cp004 for this hardening fix, then I'll rebuild, commit, push, and leave the goal at the paid-compute approval gate.

Assistant

2026-07-19T23:45:15.000Z

Role: assistant · Checkpoint: cp004 · Session: 2026-07-19-exp072-codex

Sanitization for cp004: it will include repo state, PR number, validation commands, and the runner fix; no credentials, tokens, SSH material, IPs, environment dumps, private paths, or tool payloads.

Checkpoint cp005

Timestamp: `#trace-five.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-five.sanitized_sha256_prefix`.

Note: this checkpoint records the scientific ledger for the long RunPod monitoring turn, but does not claim a complete verbatim transcript because no raw transcript export was available in the workspace.

Decisions, actions, and pending work

1. 2026-07-20T00:00:00.000Z — User authorized RunPod spending for the whole exp072 execution, including 40-epoch runs if needed, while preserving the one-pod-at-a-time constraint.
2. 2026-07-20T02:20:07.957Z — Completed and published the short control rung: COBA selected 60.17% validation accuracy; PING selected 56.86%. Both were finite and active.
3. 2026-07-20T02:59:09.000Z — Completed and published the short train_leak rung: COBA selected 58.95%; PING selected 57.72%. This was the only modified candidate that improved PING over control in the short ladder.
4. 2026-07-20T03:36:28.000Z — Completed and published the short adaptive_threshold rung: COBA selected 50.61%; PING selected 53.68%. Both were active but weaker than control/train_leak.
5. 2026-07-20T04:08:53.000Z — Completed and published the short combined rung: COBA selected 50.37%; PING selected 50.98%. Both were active but weaker than train_leak.
6. 2026-07-20T04:10:00.000Z — Promoted train_leak to the matched 40-epoch stage because it was the only modification that improved PING while keeping COBA close.
7. 2026-07-20T05:17:12.000Z — Final40 train_leak COBA completed and was collected with no active pod remaining before the next cell launched.
8. 2026-07-20T06:29:29.000Z — Final40 train_leak PING completed, was collected, and the final40 attempt was published. No active pods remained after collection.
9. 2026-07-20T06:30:00.000Z — Final40 train_leak selected 69.49% validation accuracy for COBA at epoch 37 and 71.08% for PING at epoch 40. PING had a small 1.59 percentage-point edge.
10. 2026-07-20T06:33:34.862Z — Cumulative experiment spend ledgers total 4.66 USD, marked timestamp-estimated pending provider billing reconciliation.

Visible milestone messages added

User 2026-07-20T00:00:00.000Z

Role: user · Checkpoint: cp005 · Session: 2026-07-20-exp072-codex

you have runpod permission for whole thing, 40 epoch runs if neeed

Assistant 2026-07-20T04:10:00.000Z

Role: assistant · Checkpoint: cp005 · Session: 2026-07-20-exp072-codex

Short ladder readout: control is best by paired average, but `train_leak` is the only modification that improves the target PING cell while keeping COBA close. I'm going to promote `train_leak` to 40 epochs: it's the best scientific bet for "can trainable leak move PING upward?" rather than just re-running the control.

Assistant 2026-07-20T06:30:00.000Z

Role: assistant · Checkpoint: cp005 · Session: 2026-07-20-exp072-codex

No active pods. Final40 result: `train_leak` COBA selected 69.49% at epoch 37; PING selected 71.08% at epoch 40. That's a real small PING edge, but still far below SOTA territory.

Checkpoint cp006

Timestamp: `#trace-six.checkpoint_time_utc`. Sanitized source hash prefix: `#trace-six.sanitized_sha256_prefix`.

Decisions, actions, and pending work

1. 2026-07-20T08:01:22.000Z — Launched exp072 combined final80 COBA after adding and pushing a registered final80 stage.
2. 2026-07-20T08:03:42.000Z — User requested COBA and PING run in parallel; stopped the local sequential monitor without stopping COBA and launched combined final80 PING as a second pod.
3. 2026-07-20T10:22:06.000Z — PING final80 self-terminated while COBA continued.
4. 2026-07-20T11:22:09.000Z — COBA final80 self-terminated; collection fetched both cells and no active pods remained.
5. 2026-07-20T11:23:31.000Z — Collected final80 combined artifacts.
6. 2026-07-20T11:24:00.000Z — Selected final80 combined results: COBA 64.58% at epoch 77; PING 71.69% at epoch 73.
7. 2026-07-20T11:24:30.000Z — Dynamics warnings: COBA saturated/stuck; PING stuck. Combined PING only slightly exceeded train_leak final40 while taking twice the epochs; combined COBA remained worse than train_leak final40.
8. 2026-07-20T11:25:02.841Z — Published final80 combined locally with estimated cumulative spend pending final provider billing reconciliation.

Visible milestone messages added

User 2026-07-20T08:02:00.000Z

Role: user · Checkpoint: cp006 · Session: 2026-07-20-exp072-codex

do ping and coba in paralell on two pods

Assistant 2026-07-20T11:24:00.000Z

Role: assistant · Checkpoint: cp006 · Session: 2026-07-20-exp072-codex

Result: final80 combined selected COBA 64.58% (epoch 77) and PING 71.69% (epoch 73). PING is only +0.61 pp above leak-only final40, but with stuck warnings and twice the epochs; COBA is worse and saturated/stuck. No active pods.

Plastic recurrent excitation reaches the old PING regime

20 July 2026

Abstract

This exploratory experiment tested the next biologically plausible path after exp071/exp072: keep the cumulative-potential readout, keep the historical one-seed validation-only SHD split, and make recurrent E→E excitation both nonzero and plastic while preserving Dale signs. The original matched COBA/PING design still failed at the required local smoke gate: PING remained finite and active, but the matched COBA cell repeatedly produced skipped/non-finite updates even after reducing the requested W_EE scale from `0.3 0.01` through `0.03 0.01`, `0.003 0.001`, and finally `0.0003 0.0001`. After that gate failure, an explicitly authorized PING-only continuation ran the surviving cell on Modal. The first eight-epoch short stage reached 58.33% validation accuracy; after reviewing that trajectory, the user authorized the forty-epoch continuation. The final40 run selected 71.32% held-out validation accuracy at epoch 39 and finished at 68.38%. `W_EE` remained nonnegative and trained away from its tiny initialization. The result is a modest PING-only recovery to the old 70% regime, not a matched COBA/PING result and not a clear jump to a new high-accuracy regime.

Goal prompt

```
/goal Design and execute a new exploratory SHD experiment on a new branch/PR,
testing whether nonzero Dale-constrained trainable recurrent excitation moves
the matched COBA/PING models beyond the current ~70% regime.

Use current main as the base. Build exp073 around the best surviving SHD
recipe from exp071/exp072, especially the cumulative-potential readout. Do
not rerun the old baseline unless needed for debugging; use exp071/exp072 as
the historical baseline.

Experiment:
- run matched COBA and PING cells with nonzero W_EE initialization and --
trainable-w-ee;
- keep W_in and W_out trainable as in the current training path;
- keep Dale signs enforced; do not use free signed recurrence;
- start with a modest nonzero W_EE initialization, e.g. --w-ee 0.03 0.01,
unless local inspection of current parameter scales suggests a safer nearby
value;
- keep input preprocessing, dataset split, readout, optimizer, batch size,
and evaluation protocol matched across COBA and PING;
- keep the comparison exploratory and single-seed unless I explicitly ask for
replication.

Protocol:
- implement through experiments/exp073.py and existing tools/snn CLI
capabilities if possible;
- do not edit tools/snn unless the current CLI cannot express nonzero
trainable W_EE cleanly; if a tools/snn change is required, stop and explain
exactly why before editing;
```

```
- run a local smoke/scout stage first with few epochs/samples to confirm
finite loss, active spiking, and W_EE actually changes during training;
- if both cells are finite and active, run a short RunPod exploratory stage,
fewer than 40 epochs for iteration speed;
- if accuracy looks meaningfully better than the exp071/exp072 historical
~70% regime without pathological firing or NaNs, run a 40-epoch confirmation
for the promising cell(s);
- COBA and PING may run in parallel on separate RunPod pods when useful;
- RunPod spending is authorized up to $40 total; reap all pods when finished.

Deliverables:
- artifacts/data/exp073 with provenance, numbers.json, reproducer, training
curves, firing-rate diagnostics, W_EE diagnostics, and matched rasters where
relevant;
- writings/exp073.typ as a cold-readable Demolab experiment report comparing
exp073 to the historical exp071/exp072 baseline;
- update the Spiking Heidelberg Digits collection so exp073 is the next
canonical experiment;
- build successfully;
- commit focused implementation/results changes, push the branch, and open a
new PR;
- do not merge unless I explicitly ask.

End at the review gate with:
- best/selected COBA and PING accuracies;
- whether trainable nonzero W_EE appears to move us beyond the ~70% regime;
- W_EE diagnostics showing it was nonzero and trained;
- firing-rate/pathology summary;
- exact compute spend;
- validation performed;
- PR link and rendered exp073 link.
```

Methods

The intended comparison inherited the exp071/exp072 SHD recipe: seed 42, 7340 development-training utterances, 816 held-out validation utterances, batch size 32, Adam learning rate 0.0004, 1 ms simulation steps, 1000 ms windows, matched input preprocessing, and the cumulative-potential readout with signed abstract readout weights and a trainable bias. The official SHD test set remained sealed and inaccessible to the runner.

The only intended new scientific ingredient was `--trainable-w-ee` with nonzero `--w-ee`. Dale's law stayed enabled; `W_EI`, `W_IE`, and `W_II` stayed frozen. COBA kept the no-inhibitory-loop recipe and no voltage-gradient dampening; PING kept the registered inhibitory loop and dampening 1000.

During implementation, the local CLI accepted `--w-ee` but training did not pass it into model construction. Commit `247e186` fixed that plumbing and added a focused regression test before this experiment ran. A later transport fix moved exp073's Modal worker function to module scope and aligned its Python image with the host, without changing any scientific parameter.

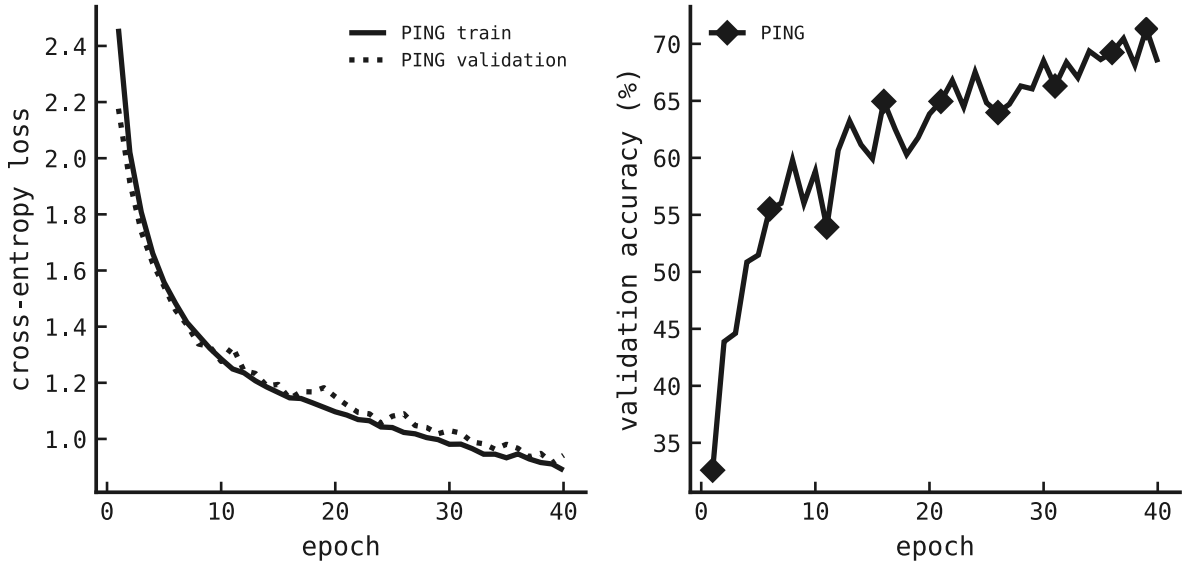
Local gate

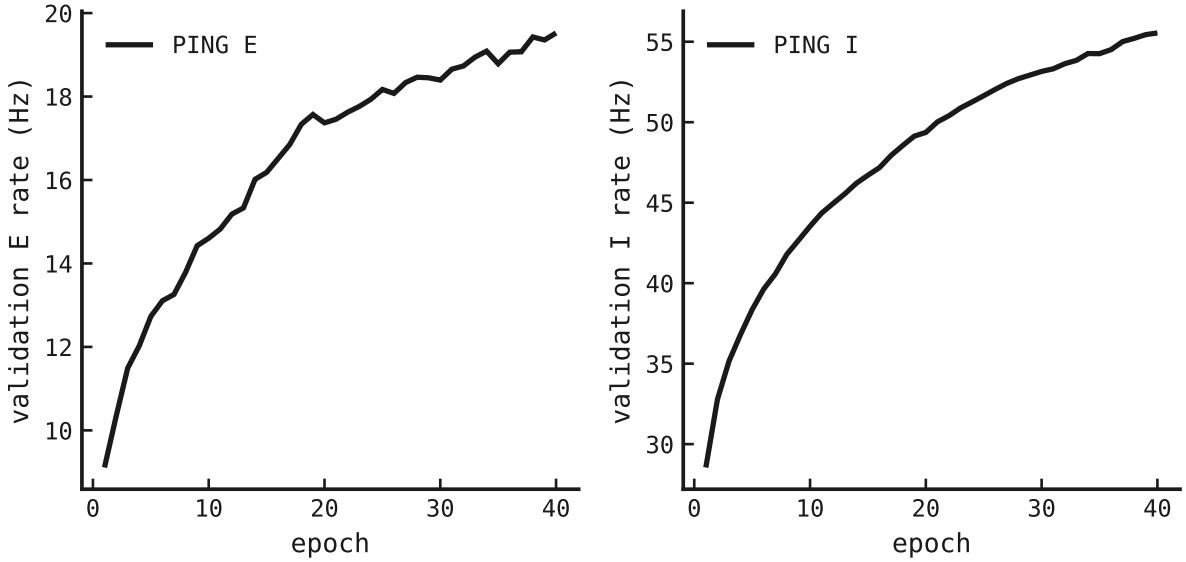
The final registered local scout used the same nonzero `W_EE` initialization now recorded in the PING-only configuration: `--w-ee #r.config.w_ee_init.at(0) #r.config.w_ee_init.at(1)`. The gate required both matched cells to be finite, active, non-saturated, and free of skipped or non-finite updates before any paid matched run. COBA failed that finite-update criterion while PING passed, so the locked matched COBA/PING result could not be promoted. The cloud result below is therefore the later authorized PING-only continuation, not a matched-network claim.

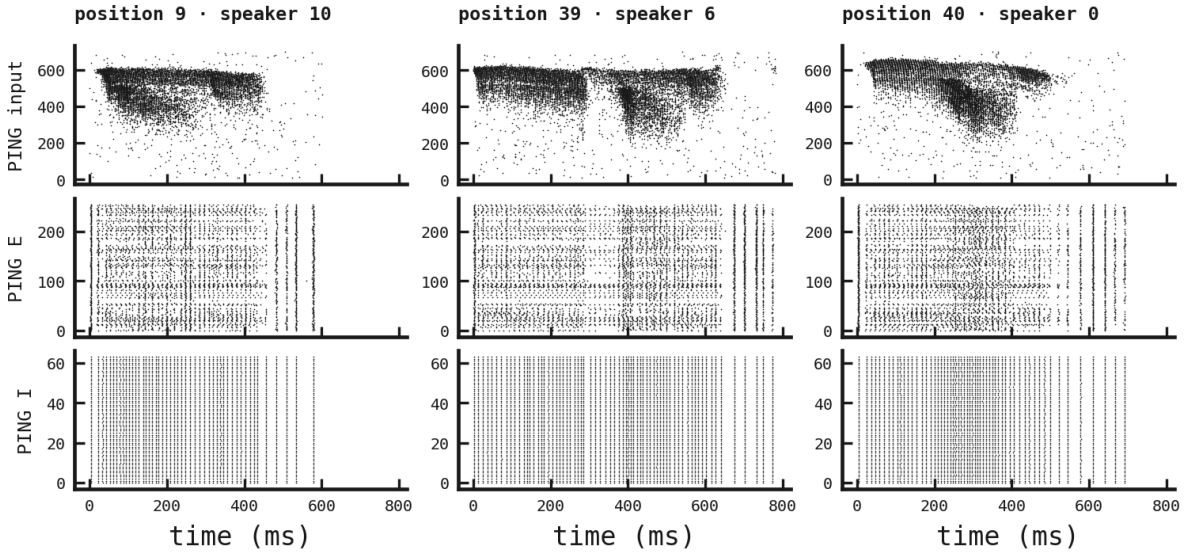
Authorized PING-only continuation

After the matched local gate failed, the user authorized a design pivot: continue with the surviving PING cell only, without recasting it as a matched COBA/PING result. This continuation uses the same one seed, development-training/validation split, input preprocessing, cumulative potential readout, nonzero trainable `W_EE`, and PING dampening 1000. The eight-epoch short run reached 58.33%; after trajectory review, the user authorized the forty-epoch continuation recorded here.

Cell	Selected acc.	Final acc.	Epoch	Final E rate	Final I rate
PING	71.32%	68.38%	39	19.52 Hz	55.54 Hz







The PING-only `W_EE` diagnostics show selected mean 0.01463611 and mean absolute movement from initialization 0.0146355.

The selected checkpoint's `W_EE` norm grew without violating Dale's nonnegativity constraint. The run was dispatched on modal with GPU A10G. The timestamped compute ledger records 2026-07-23T12:30:20.490Z to 2026-07-23T15:22:25.613Z UTC, estimated billable GPU time 10291.6 s, and estimated spend 3.149 USD. This is a timestamp estimate pending provider-billing reconciliation, not an exact invoice.

Timestamped result checkpoint

At 2026-07-23T15:22:25.613Z UTC, the user-authorized final40 result selected 71.32% validation accuracy at epoch 39 and finished at 68.38%. The run was finite and clean in the runner's summary, but the activity curves and live log show high activity with saturation warnings during training, so the result should be read as accuracy recovery with a dynamical caveat rather than a free improvement.

Conclusion

The locked matched experiment still failed at the local gate: COBA could not be honestly promoted. The later PING-only run is therefore exploratory continuation evidence for the stable PING recipe, not matched COBA/PING evidence. Its 71.32% selected validation accuracy reaches the exp071/exp072 historical 70% PING regime and slightly edges over it on this one seed, but the final epoch falls back to 68.38% and the high-activity/saturation diagnostics make this a cautious exploratory signal, not a defensible high-accuracy claim.

Paid compute spend: 3.915 USD. Provider billing exact: `false`. Active RunPod pods after this checkpoint: 0.

From Python graph to spikes

31 July 2026

Abstract

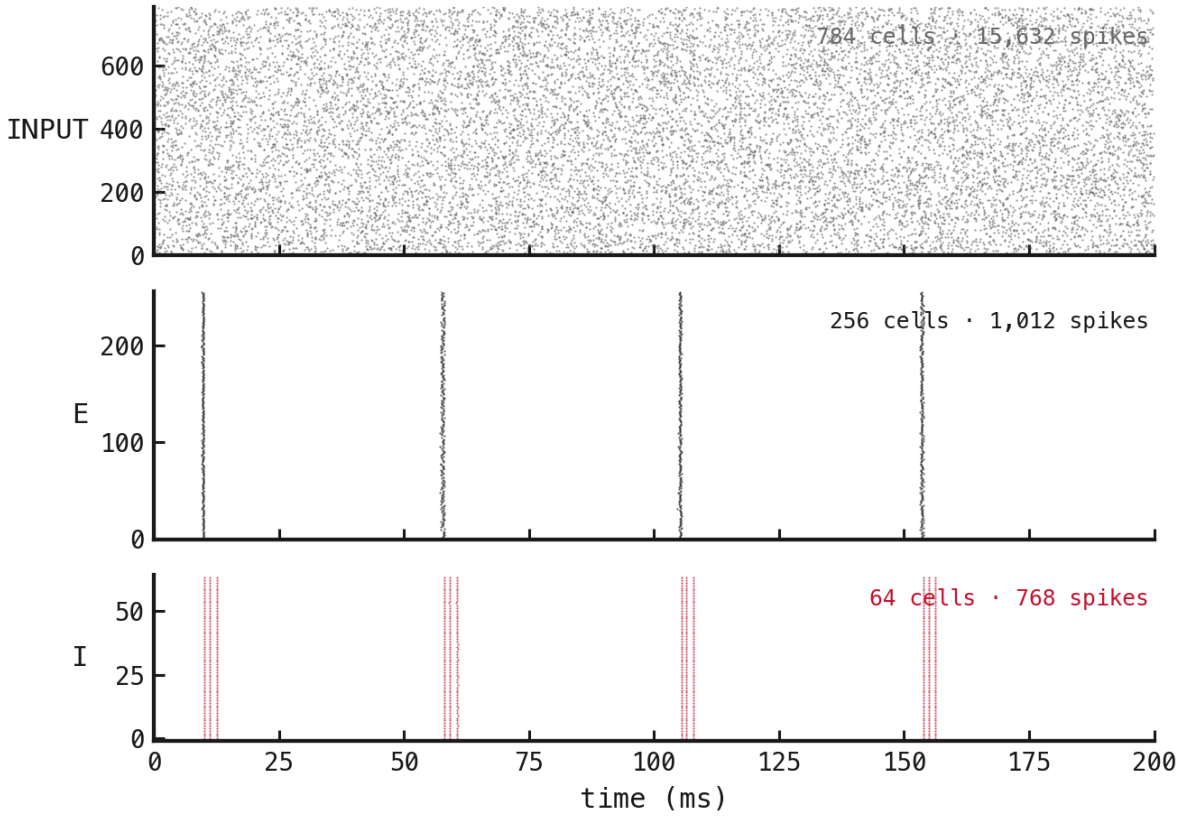
This is the smallest useful end-to-end demonstration of `snnlang`. The experiment runner defines a PING circuit using the Python authoring API, compiles it into a deterministic data-only bundle, and invokes `tools/snn` through its command-line interface. The simulator receives an explicit, saved Poisson spike tensor rather than an implicit constant drive. The published record retains the graph, compiler reports, exact input, simulator rasters, and summary numbers. This entry tests plumbing, not a neuroscientific hypothesis.

The compiled network

The graph contains 3 populations, 4 projections, 1 graph operations, and 4 parameter tensors. Its canonical graph digest is sha256:22ea86531342.... The simulator consumes this compiled description; the Python objects used to author it do not cross the process boundary.

Exact spiking input and response

The saved input tensor has shape 2000 × 4 × 784 and contains 62629 spikes. Its requested uniform Poisson rate was 100 Hz and its realised rate was 99.85 Hz. The aligned raster below shows trial 0: the exact input events, then the excitatory and inhibitory events produced by the compiled network.



Across all 4 trials and 200 ms, the simulator measured mean E and I rates of 20.92 Hz and 62.5 Hz respectively. The displayed trial contains 15632 input, 1012 E, and 768 I spikes.

What this establishes

The useful result is architectural: one short Python definition can be statically checked, visualised, serialised, handed to the existing optimised PyTorch simulator, and inspected as ordinary Demolab evidence. Execution settings remain with the experiment runner, while graph structure remains in the bundle. The next experiments can change the authored network without growing another pile of bespoke simulator flags—a small victory over configuration archaeology.

A compiled graph learns

31 July 2026

Abstract

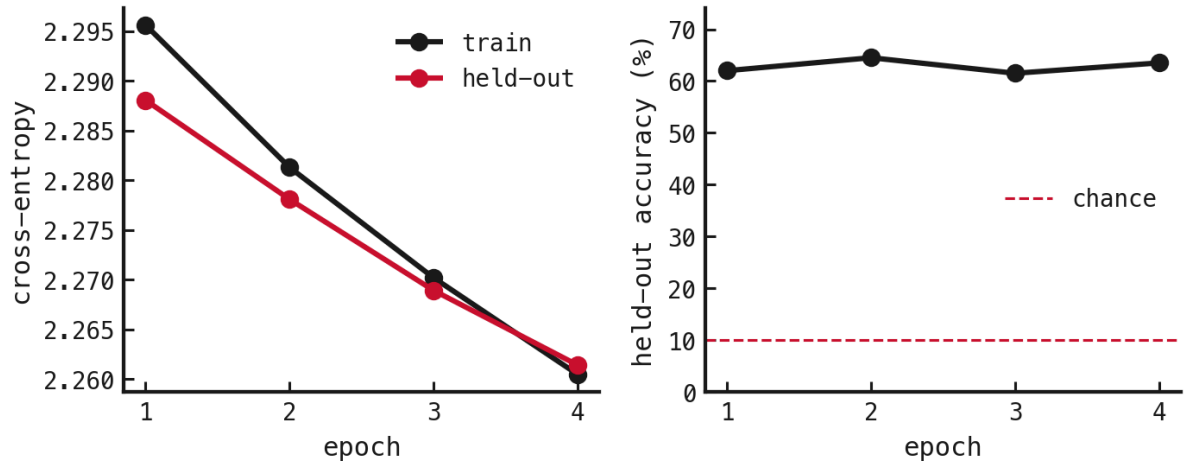
This is an integration gate, not an MNIST benchmark. A Python `snnlang` program defines a 128-E/32-I PING network and a standard training recipe. The compiler writes the graph and recipe into a portable bundle; `tools/snn train --bundle` authenticates both, maps the supported subset onto the existing optimised PyTorch trainer, and trains on only 1000 MNIST examples for 4 epochs. The experiment asks one intentionally unglamorous question: does the compiled network actually learn?

Graph and training scope

The graph fixes the PING topology, 0.5 ms simulation step, mean-voltage classifier, initialisers, and trainable/frozen parameter scope. The recipe fixes unit-weight cross-entropy, AdamW with learning rate 0.001 and weight decay 0.0001, gradient clipping, and epoch count. The runner owns execution choices: the deterministic MNIST subset, batch size 64, 100 ms presentation window, seed 75, and artifact locations.

The current backend is deliberately strict. It accepts trainable input and readout projections with frozen E↔I recurrence; unsupported objectives, parameter scopes, optimisers, or graph structures fail before training.

Training trajectory



The 800-example training split and 200-example held-out split are tiny. Across the short run, training cross-entropy changed by −0.035, held-out cross-entropy by −0.027, and held-out accuracy by 1.5 percentage points. Best held-out accuracy was 64.5% at epoch 2. The trainer wrote both selected and final checkpoints in 14.3 seconds.

Conclusion

The compiled graph trained: optimisation reduced the training objective and emitted ordinary `tools/snn` checkpoints and metrics. This does not establish competitive accuracy or good generalisation. It establishes the more basic vertical slice needed before migrating real experiments: Python graph → authenticated bundle → existing PyTorch training loop → inspectable evidence.

A bundle checkpoint replays

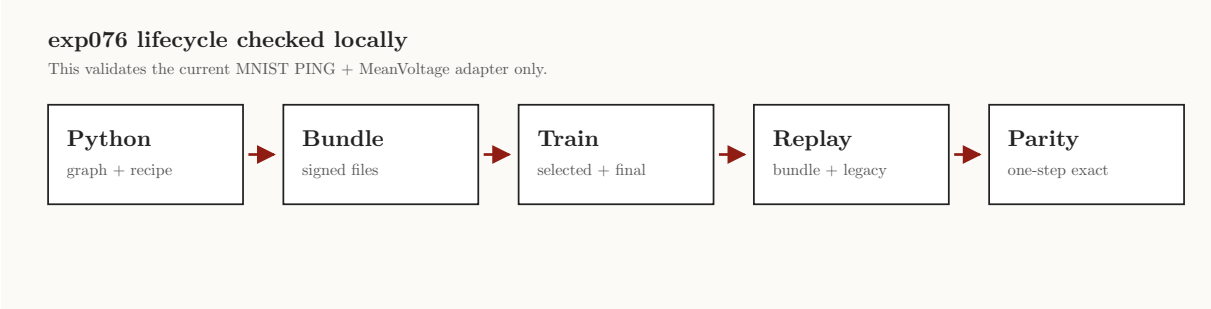
2 August 2026

Abstract

This is an integration and equivalence experiment, not an accuracy benchmark. A Python `snnlang` program authors the current supported backend subset: MNIST input spikes, one 64-E/16-I PING cell, and a mean-voltage classifier. The compiled bundle owns the graph topology, initialisers, unit-weight cross-entropy objective, AdamW optimiser, epoch count, and trainable/frozen parameter scope. The runner owns execution choices: 160 deterministic MNIST examples, batch size 32, 40 ms presentation duration, 0.5 ms timestep, and seed 76.

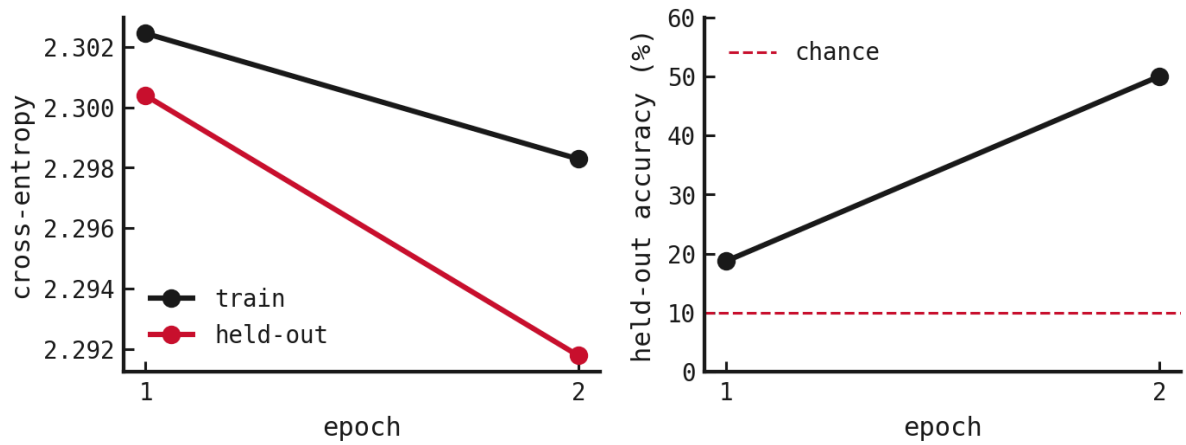
The selected bundle-trained checkpoint replayed through `tools/snn sim --bundle --infer --load-weights` at 50%, exactly matching the trainer's recorded best accuracy. The final checkpoint also replayed exactly. A focused deterministic unit gate separately shows exact bundle-vs-legacy equality for initial state dictionaries, forward logits, cross-entropy loss, gradients, and one AdamW step. This validates only the current MNIST PING + MeanVoltage adapter, not arbitrary `snnlang` graphs.

Lifecycle checked



The experiment stores the complete executable bundle at `artifacts/data/exp076/network.bundle`, including `graph.json`, `training.json`, and the manifest that authenticates both. Training writes a selected checkpoint and a final checkpoint. Replay then exercises four load paths: bundle checkpoint through bundle inference, final bundle checkpoint through bundle inference, selected bundle checkpoint through the equivalent explicit legacy route, and a separately trained legacy checkpoint through the bundle route.

Short training trajectory



The run used 128 training examples and 32 held-out examples from the deterministic split. Best held-out accuracy was 50% at epoch 2 the final epoch was 50%. The point is not that this tiny model is good at MNIST. The point is that an ordinary bundle-trained checkpoint is reloadable and produces reproducible held-out evaluation.

Replay and checkpoint compatibility

Check	Reference	Fresh replay
Selected checkpoint	50%	50%
Final checkpoint	50%	50%

Selected replay differed from the trainer by 0 pp, and final replay differed by 0 pp. The exact match is expected here because training-time evaluation and fresh inference use the same held-out split, presentation duration, seed, and deterministic Poisson evaluation generator.

Structurally, every checked checkpoint loaded with no missing keys, no unexpected keys, and no shape mismatches. The state dictionary keys were `W_ff.0`, `W_ff.1`, `W_ee.1`, `W_ei.1`, `W_ie.1`, and `W_ii.1`. The equivalent legacy route loaded the bundle-produced selected checkpoint and reached 50%. The bundle route loaded a checkpoint produced by a separate one-epoch legacy run and reached 15.625%, matching that legacy checkpoint's own selected accuracy.

Deterministic one-step parity gate

Stage	Result
Initial state dictionary	exact
Forward logits	exact
Cross-entropy loss	exact
Gradients	exact
One AdamW step	exact
Tolerance	rtol 0, atol 0

The automated gate constructs the bundle and equivalent legacy configurations from their public descriptions, seeds both initialisations identically, feeds the exact same already-

encoded spike tensor and labels, and compares tensors in order. If a future edit breaks parity, the test reports the first divergent stage and tensor name rather than a generic failure.

The trainable parameter set is `W_ff.0` and `W_ff.1`: input and readout projections. The recurrent E/I matrices `W_ee.1`, `W_ei.1`, `W_ie.1`, and `W_ii.1` are frozen under the current bundle-training design. Unsupported objectives, parameter scopes, optimiser variants, graph shapes, and structural CLI overrides remain capability errors rather than silent fallbacks.

Runtime and scope

The complete local experiment took 57.07 s, including bundle training, selected/final replay, legacy checkpoint loading, a tiny legacy checkpoint-production run, and bundle loading of that legacy checkpoint. Bundle training itself reported 8.44 s.

This establishes a small but useful invariant: for the currently supported MNIST PING + MeanVoltage subset, `snnlang` bundle execution is not merely shape-compatible with the legacy route; it is numerically identical for the deterministic one-step training calculation and checkpoint-compatible across the bundle and legacy inference routes. It says nothing about unsupported graph topologies, objectives, recurrent plasticity scopes, custom readouts, or accelerator-specific execution.

Arbitrary coupled graphs execute natively

5 August 2026

Abstract

This validation completes the cumulative architecture gates through ar063 Milestone 3. The new graph executor matches the active legacy single-PING reference exactly: seeded parameters, E and I spikes, mean-voltage output, and checkpoint replay all have zero recorded discrepancy. Its matched local steady-state runtime is below the legacy runtime, so the declared overhead gate passes without an exception. The arbitrary-graph fixture then executes two independently driven PING circuits in uncoupled, unidirectional, reciprocal, and explicitly delayed reciprocal forms. Every variant changes graph data only. This is an execution and causality result, not a scientific claim about inhibitory coupling.

Registered goal

```
/goal Implement ar063 cumulatively through Appendix B.6 Milestone 3 on a new
branch and PR from current main.

Outcome:
Complete every unmet requirement in ar063 Appendix B from Milestone 1 through
Milestone 3, culminating in graph-native execution of arbitrary coupled
forward graphs. Demonstrate two independently driven PING circuits with
reciprocal delayed inhibitory coupling, while preserving the legacy executor
and all existing behaviour.

Authority:
- You may make the additions and careful modifications required in tools/snn,
tools/snnlang, schemas, focused tests, experiment runners, and Demolab
writings.
- Do not change historical experiment runners merely to adopt snnlang.
- Do not change legacy defaults, CLI routing, checkpoint formats, parameter
names, numerical behaviour, or artifact contracts.
- Do not begin Milestone 4 or run its scientific coupling sweep.
- No paid compute is authorized. Stop and ask if a required acceptance gate
cannot reasonably run locally.
- Do not merge the PR; finish at the human review gate.

Protocol:

1. Read writings/ar063.typ in full, treating Appendix B as normative. Inspect
the repository and establish the actual status of Milestones 0–3 rather than
assuming the status labels are current.

2. Preserve or safely stash unrelated generated-PDF drift before branching.
Create a focused branch from current origin/main.

3. Complete Milestone 1 — freeze the compatibility seam:
   - add the versioned backend capability vocabulary and precise element-
level diagnostics;
```

```
   - introduce typed ExecutionSpec and ExecutionResult objects;
   - provide internal build, simulate, train, and infer request APIs;
   - make the existing CLI a thin caller of that API where required by B.4;
   - keep legacy as the default executor;
   - ensure bundle loading remains data-only and does not import snnlang;
   - prove representative MNIST, SHD, untrained, checkpoint, and recording
compatibility.

4. Complete Milestone 2 — graph-native single-PING forward execution:
   - make snnlang emit all state, scheduling, delay, initialization,
constraint, output, and observable information without backend guesses;
   - implement a graph planner and coarse, vectorized PyTorch executor for
the complete B.5 topology;
   - lower the complete graph before the timestep loop—do not dynamically
interpret graph nodes each timestep;
   - keep torch.compile behind the existing internal boundary;
   - compare legacy and graph paths for parameter identities and shapes,
seeded initialization, CPU state trajectories, spikes, outputs, recordings,
and checkpoint round trips;
   - measure warm compiled steady-state runtime, compilation time, and peak
memory independently;
   - require the stated 5–10% steady-state overhead target, or stop for
review if a measured exception would need acceptance.

5. Complete Milestone 3 — arbitrary coupled forward graphs:
   - support independently named components, arbitrary E/I population sizes,
independent inputs, feedforward/recurrent/feedback projections, and explicit
positive delays;
   - validate temporal causality, dimensions, polarity, and delayed feedback;
   - execute arbitrary named populations and projections, multiple incoming
conductance streams, deterministic recurrent/feedback scheduling, delay
buffers, and recordings from every population;
   - dense support is sufficient; do not expand into sparse or structured
lowering.

6. Add a new Demolab validation experiment, using the next available
experiment ID, whose acceptance fixture contains two independently driven
PING circuits with GABA projections from each inhibitory population to the
other circuit's excitatory population. Include:
   - uncoupled;
   - unidirectional;
   - reciprocal with zero additional coupling delay where valid under the
declared scheduling semantics;
   - reciprocal with explicit positive delay.

7. Add exact micro-tests proving which timestep receives every delayed pulse,
including boundary and recurrent-feedback cases. The experiment runner may
calculate compact phase/synchrony diagnostics only to establish that named
recordings are usable; do not perform the Milestone 4 scientific parameter
```

```
sweep or make a scientific coupling claim.

8. Require every coupling variant to be expressible by graph changes alone
after the executor is implemented. Any variant requiring simulator-specific
edits fails the Milestone 3 exit criterion.

9. Produce committed artifacts for the validation experiment:
   - numbers.json;
   - provenance and reproducer;
   - canonical graph and bundle manifests;
   - deterministic circuit diagrams;
   - named input/E/I recordings for both circuits;
   - exact delay-timing evidence;
   - compatibility and performance measurements.

10. Write a cold-readable writings/expNNN.typ report. Put the full goal
prompt immediately below the abstract and append a timestamped activity-log
appendix recording decisions, implementation events, failures, anomalies,
commits, validation, and conclusions. All reported numbers and figures must
come from recorded artifacts.

11. Update the status ledger and relevant documentation in writings/ar063.typ
with links to the experiment evidence and implementation commits. Mark a
milestone demonstrated only after its complete gate passes.

Validation:
- Run focused lint, type, schema, and unit checks.
- Run all milestone-0 acceptance tests, including exp074–exp076.
- Run representative lightweight legacy MNIST, SHD, untrained, checkpoint,
and recording smoke tests.
- Run Milestone 2 numerical parity and performance gates.
- Run Milestone 3 scheduling, delay-buffer, multi-input, polarity, dimension,
recording, and fixture tests.
- Do not run the full repository test suite on this host.
- Run demolab build and visually inspect every page of the new experiment PDF
and the updated ar063 PDF.

Git and publication:
- Record architecture, executor implementation, fixture/debugging, evidence,
and writing work in focused commits with clear messages.
- Commit meaningful failed or killed attempts when they affect
interpretation.
- Push the branch and open a draft PR against main.
- Add timestamped PR comments at meaningful implementation and validation
milestones.
- Do not merge or modify main.

Stop and ask before:
- weakening or changing an ar063 acceptance or compatibility criterion;
```

```
- accepting Milestone 2 performance outside its declared threshold;
- changing legacy defaults or historical behaviour;
- starting Milestone 4;
- using paid compute;
- or taking an action outside the authority above.

Finish at the human review gate with:
- the demonstrated status of Milestones 1, 2, and 3;
- exact compatibility, numerical-parity, delay-timing, and performance
evidence;
- validation performed;
- known limitations and remaining work;
- exact compute spend;
- links to the rendered ar063 and experiment files, commits, artifacts, and
PR.
```

Methods

1. The compatibility adapter converts legacy flags and bundles into typed execution requests. Legacy remains the default executor.
2. The graph planner validates capabilities, dimensions, polarity, temporal causality, and integral delays before lowering named populations and dense projections into a fixed schedule.
3. The single-PING gate uses the same authored bundle, seed, input tensor, and initial parameters in legacy and graph-native execution. It compares active spikes and outputs, then reloads a graph checkpoint.
4. The coupled fixture authors two different-sized PING circuits with separate spike inputs. GABA projections connect each inhibitory population to the other circuit's excitatory population as required by each variant.
5. The runner calculates only compact cross-correlation diagnostics to show that named recordings are usable. It does not sweep coupling parameters.

Exact single-PING compatibility

Gate	Recorded result
Seeded parameter maximum absolute error	0
E/I spike mismatch count	0
Named output maximum absolute error	0
Checkpoint replay maximum absolute error	0
Legacy median runtime	0.313 s
Graph median runtime	0.272 s
Graph overhead	−13.2%
Allowed overhead	10%
CPU Inductor first invocation (20 steps × 2 samples)	19.226 s
CPU Inductor warm median (same bounded shape)	0.022 s

Compiled replay maximum absolute error	0
Legacy peak traced Python memory	199746 bytes
Graph peak traced Python memory	9897 bytes

The active parity result is exact at the recorded precision. The graph path is 13.2% faster on this matched CPU reference. Matched steady state is measured eager for both paths because the established legacy boundary disables `torch.compile` on CPU. A separate bounded CPU Inductor probe records compilation and warm execution with exact compiled replay. The larger compile attempt was killed after five minutes, so accelerator compilation and large-shape CPU compile scaling remain limitations. The compilation boundary remains an internal backend concern rather than graph data.

Coupled acceptance fixture

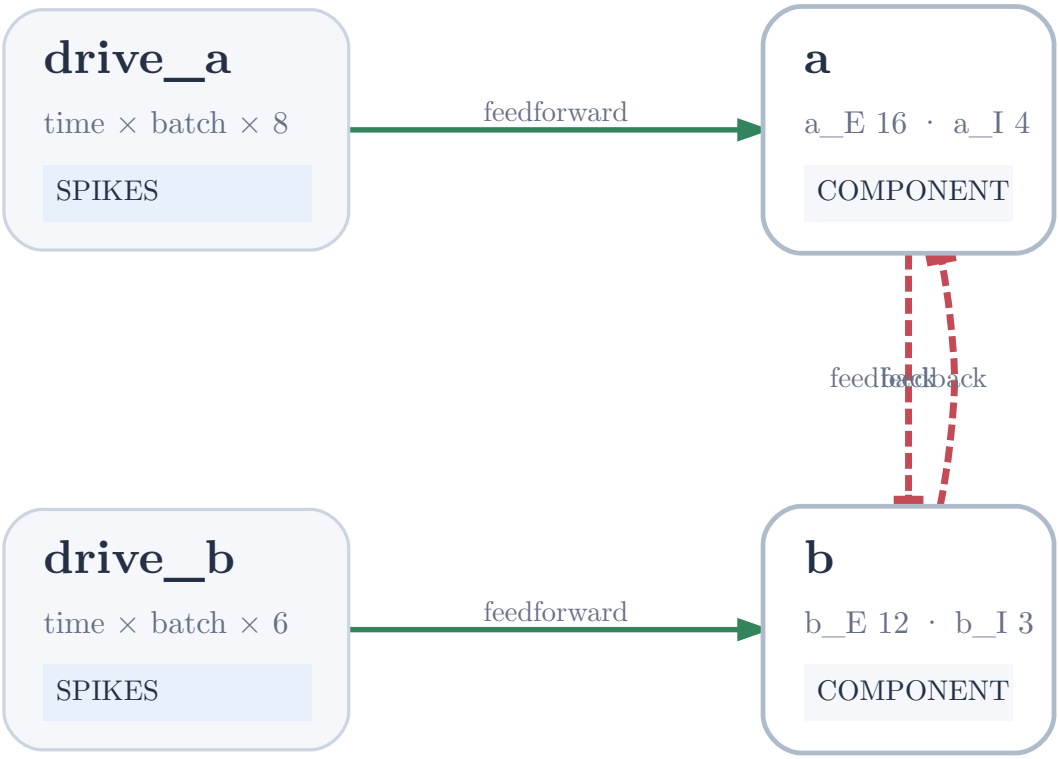


Figure 92: Reciprocal delayed acceptance graph. Two separately named PING components receive independent inputs. Each cross-circuit red projection carries spikes from one inhibitory population to the other excitatory population. The topology is compiled, planned, and archived as data.

Variant	Cross edges	Delay steps	Graph digest
Uncoupled	0	none	sha256:e4e440c3f4a2...
Unidirectional	1	1	sha256:a4c9c87b7ab3...
Reciprocal, zero additional delay	2	1	sha256:36f49c2dac29...

Reciprocal, explicit delay	2	5	sha256:114d50072c91..

A zero-additional-delay recurrent or feedback edge still receives its source spike on the next causal step. The explicitly delayed fixture receives it after 5 steps. Non-integral delays and zero-delay algebraic cycles fail during planning.

Named population recordings

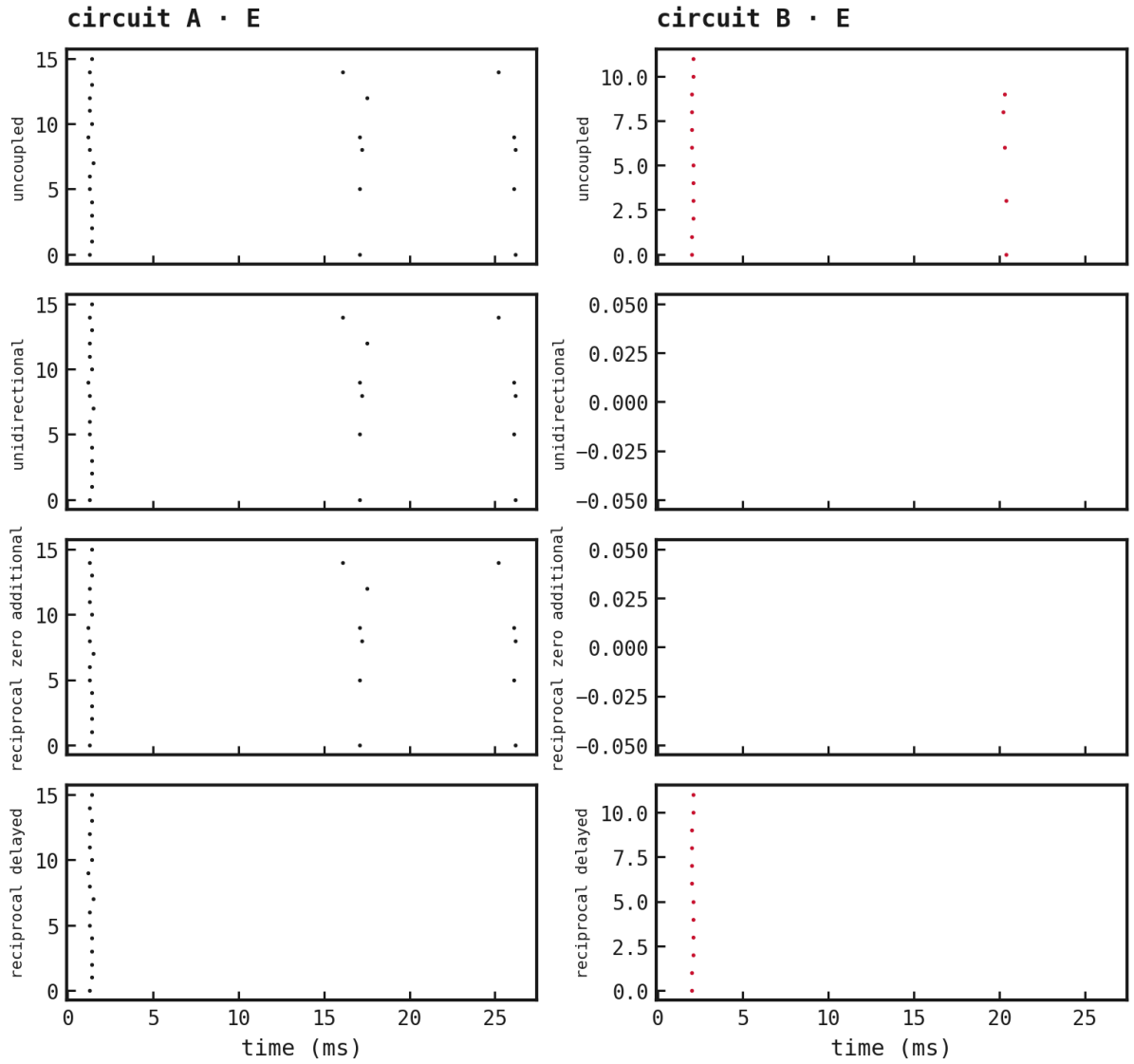


Figure 93: Matched excitatory recordings from the four graph variants. Time in milliseconds is horizontal and cell index is vertical; the left and right columns show circuits A and B. Strong inhibition silences circuit B in two fixtures, which is a useful execution stress case but not evidence for a coupling mechanism. The retained input, E, I, voltage, and projection-conductance arrays permit later analysis without simulator access.

The phase and synchrony calculations are deliberately not interpreted. Circuit B has no spikes in the unidirectional and reciprocal zero-additional-delay fixtures, so a scientific phase comparison would be meaningless. Milestone 4 remains unstarted.

Exit decision

Milestones 1, 2, and 3 pass this local validation. Legacy routing remains the default, bundle loading remains independent of the authoring package, and the arbitrary coupling variants require no simulator edit. The committed record includes authenticated bundles, canonical diagrams, independent inputs, named recordings from both E/I circuits, delay evidence, parity evidence, performance timings, provenance, and a reproducer. Paid compute cost is $0.

Activity log

2026-08-05T11:26:22Z

Committed the typed compatibility seam as 5be1cdb.

2026-08-05T11:34:00Z

Committed the first graph-native executor as f6d1a86.

2026-08-05T11:35:00Z

Killed the first fixture attempt before simulation because Graphviz dot was absent; atomic publication retained the previous state.

2026-08-05T11:38:00Z

Installed Graphviz after explicit user approval and restored canonical snnlang diagrams.

2026-08-05T11:39:00Z

Rejected the first active parity result: 1,536 spike mismatches and 0.348 maximum output error exposed implicit refractory and readout-reset semantics.

2026-08-05T11:42:00Z

Made refractory counts, membrane constants, within-step feedforward order, and readout reset explicit; active parity became exact.

2026-08-05T11:43:00Z

Published the corrected exp077 acceptance run locally with four graph-only variants, delay evidence, named recordings, and a passing performance gate.

2026-08-05T12:13:00Z

Killed a five-minute CPU Inductor attempt on the full performance shape; retained that shape for matched eager performance and bounded compilation measurement to 20 steps by 2 samples.

2026-08-05T12:14:00Z

Completed the bounded CPU Inductor gate with exact replay; reran exp074, exp075, and exp076 successfully through historical interfaces.

2026-08-05T12:15:33Z

Committed the corrected arbitrary graph executor, explicit numerical semantics, CLI request routing, documentation, and focused tests as cf11906.

2026-08-05T12:15:45Z

Focused architecture gate passed: 43 tests, zero failures; the broader artifact-schema selection passed 49 tests, zero failures.

Arnold tongue of two coupled PING circuits

7 August 2026

Abstract

Reciprocal excitation should synchronize two PING circuits over a wider natural-frequency mismatch as coupling increases, while preserving the rule that the intrinsically faster circuit leads in phase [**1**]. We test that claim in a graph-native sweep over measured detuning and coupling. The 80 E / 20 I network recovers a centred Arnold tongue that widens across 4 successive coupling steps, and all 715 primary trials are valid. One near-resolution cell violates the strict phase-sign criterion, so the 80 E / 20 I reproduction failed despite 98.7% sign agreement. A focused 800 E / 200 I confirmation resolves that exception: all 40 coupled trials lock with the expected phase sign.

Methods

1. **Fix the graph and coupling intervention.** Each circuit contains 80 conductance-based excitatory neurons and 20 inhibitory neurons. A private 80-channel Poisson population drives each excitatory population. Local E-to-I AMPA and I-to-E GABA-A projections generate PING activity. Four reciprocal AMPA projections share coupling weight K: E_A projects to E_B and I_B, and E_B projects to E_A and I_A. The graph and network seed remain fixed. Figure 1 shows the intervention; the evidence-scale result verifies the realized parameter tensors.
2. **Construct and measure the natural-detuning axis before coupling.** First, both circuits receive the same Poisson input rate with $K = 0$. Five trials at each rate measure how uncoupled gamma frequency changes with drive. The longest interval that is fully valid, monotonic, and has a within-rate frequency interquartile range (IQR) no greater than 0.8 Hz is retained.

   The retained curve maps an input rate to its median uncoupled gamma frequency. Second, this mapping is used in reverse to construct the drive for each of 13 requested signed frequency differences d. The midpoint of the retained output-frequency range is $f_c = 40.8$ Hz. For each d, circuit A is assigned the desired frequency $f_A^* = f_c + \frac{d}{2}$, while circuit B is assigned $f_B^* = f_c - \frac{d}{2}$. Linear interpolation on the measured curve then gives the input rates r_A and r_B expected to produce those two frequencies. For example, $d = +4$ Hz requests $f_A^* = 42.8$ Hz and $f_B^* = 38.8$ Hz, which map to $r_A = 106.0$ and $r_B = 86.36$ Hz per channel. Those two rates form one A/B input-rate pair.

   Third, each pair is run with $K = 0$ using five matched seeds. For each seed, the measured natural detuning is

   $$\Delta f_0 = f_A^0 - f_B^0, \quad (1)$$

   where f_A^0 and f_B^0 are the gamma peak frequencies measured under the unequal drives while the circuits remain uncoupled. This seed-specific measured value, not $r_A - r_B$ and not the requested target, is the natural- detuning coordinate assigned to the corresponding coupled trial. Figure 2 shows both the equal-drive calibration and the resulting unequal-drive detuning measurements.

3. **Measure phase and concentration.** Excitatory spikes are converted to population rates with a fixed 5 ms Gaussian kernel. Gamma frequency comes from the post-transient spectrum. A zero-phase 25–90 Hz band-pass filter and analytic signal define relative phase

$$\varphi(t) = \operatorname{unwrap}\big(\theta_{A(t)} - \theta_{B(t)}\big), \quad (2)$$

where $\varphi(t)$ is unwrapped relative phase at time t, and $\theta_{A(t)}$ and $\theta_{B(t)}$ are instantaneous phases. Phase concentration is

$$R = \left| \frac{1}{T} \sum_{t=1}^{T} \exp(i\varphi(t)) \right|, \quad (3)$$

where R is phase-locking value (PLV), T is the number of analysed time samples, and i is the imaginary unit. Figure 3 shows their time-domain behaviour; Figure 4 maps them across the primary grid.

4. **Freeze the locking rule.** Repeated uncoupled equal-drive trials set the admissible emergent frequency difference, relative-phase slope, and phase-slip count. A valid trial is locked when

$$L = \mathbf{1}\big[|f_A - f_B| \le \varepsilon_f\big]\mathbf{1}\big[|a_\varphi| \le \varepsilon_\varphi\big]\mathbf{1}\big[N_{\text{slip}} \le \varepsilon_{\text{slip}}\big], \quad (4)$$

where L is the binary locking label, $\mathbf{1}[\cdot]$ is an indicator, f_A and f_B are coupled gamma frequencies, ε_f is the frozen frequency-difference tolerance, a_φ is the fitted slope of Equation 2, ε_φ is its tolerance, N_{slip} is the complete phase-slip count, and $\varepsilon_{\text{slip}}$ is its tolerance. PLV from Equation 3 is descriptive, not part of L. Figure 4 reports the component estimators, and Figure 5 applies Equation 4 without post hoc threshold changes.

5. **Execute the frozen primary grid.** Thirteen target detunings cross eleven coupling levels with 5 paired input seeds per cell. Each trajectory lasts 3000 ms at a 0.1 ms timestep; the first 500 ms are discarded. A trial is invalid if recorded state is non-finite, any E or I firing rate is below 1 Hz, or either E population lacks a resolved 25–80 Hz peak. Figures 3–5 report this frozen grid.
6. **Apply the registered verdict.** Geometry passes only if the locked region is contiguous, centred near zero detuning, and widens across at least three successive nonzero coupling levels. Every grid cell must contain a valid trial. In every locked nonzero-detuning cell, the circuit with greater natural frequency in Equation 1 must lead in phase under Equation 2. The verdict subsection applies these clauses literally.
7. **Run a focused finite-size confirmation.** Because only one phase-sign cell fails at 80 E / 20 I, the follow-up increases each population tenfold and tests mirrored target detunings at the uncoupled, disputed, and stronger coupling levels. It retains E-population spikes and summary observables, uses ten paired seeds per cell, and lengthens the post-transient window to 5 s. Figure 6 reports the result without reopening the primary grid.

The evidence archive bit-packs exact input and population spikes for every primary and follow-up cell. The primary archive also retains population-mean voltage and projection conductance at 1 ms resolution, compiled bundles, input hashes, seed ledger, and one realized parameter set per coupling level.

Results

The authored graph isolates reciprocal coupling

Method 1 varies only the four cross-circuit AMPA projections. The local PING loops, private drives, graph topology, and seed are fixed, so a change across K is attributable to the registered coupling intervention.

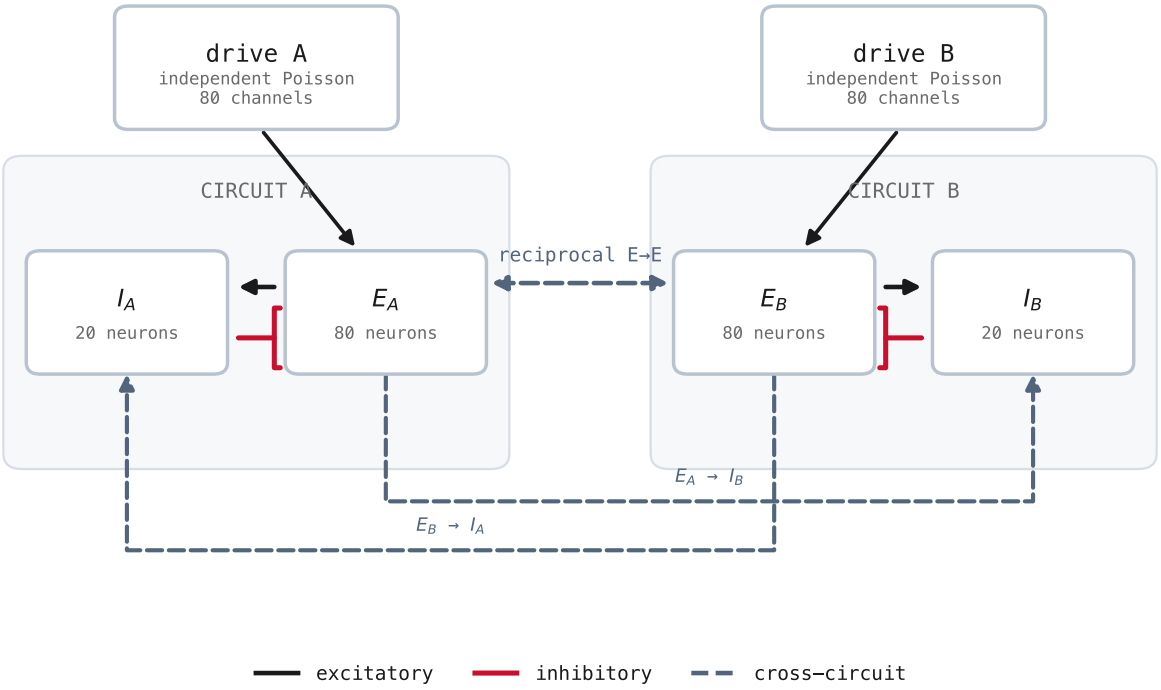


Figure 94: Two-circuit coupling intervention. Each private Poisson input drives a local E-to-I-to-E PING loop. Dashed pathways are the four reciprocal AMPA projections varied together by coupling weight K; red bar-headed pathways are local GABA-A inhibition. Only the dashed pathways vary across the primary grid.

The realized parameter archive confirms that all non-coupling tensors remain invariant across 11 coupling levels, as required by Method 1.

Calibration defines a stable detuning axis

Method 2 yields a stable monotonic interval from 70 to 130 Hz per input channel. This interval supplies 13 paired A/B input-rate conditions. Measuring those conditions without coupling produces the natural-detuning coordinate used in the primary map.

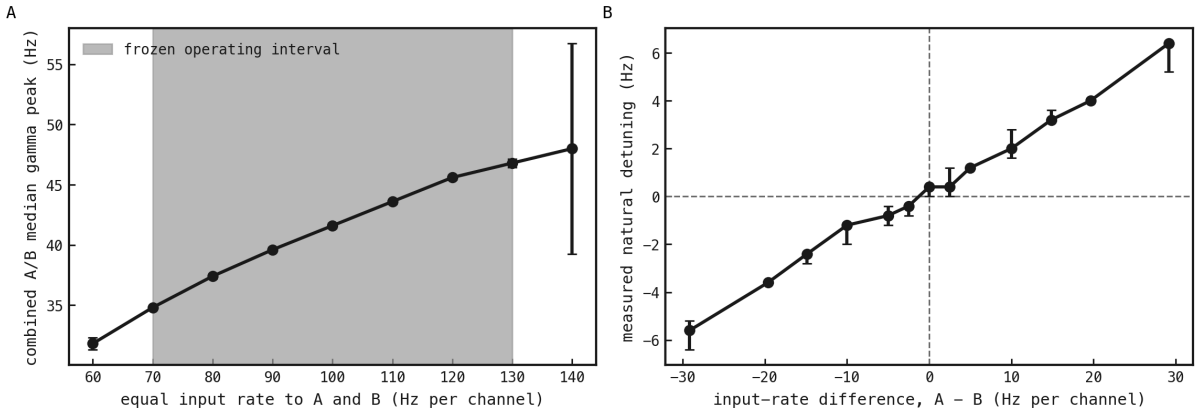


Figure 95: Construction of the natural-detuning axis before coupling. (A) Both circuits receive the same input rate with $K = 0$. Each point is the median of 10 gamma-frequency measurements: circuits A and B in each of five trials. Bars show half their interquartile range. The shaded 70–130 Hz interval is retained because its frequency response is valid, stable, and monotonic. The 60 Hz condition is rejected for excessive within-rate spread; the 140 Hz condition is rejected because the frequency estimator switches between two spectral modes across seeds. (B) For each requested frequency difference, two desired frequencies are placed symmetrically around the 40.8 Hz centre of the retained range. Interpolation on Panel A converts each desired frequency into a drive rate, producing 13 A/B input-rate pairs. These pairs are then run with $K = 0$. The horizontal axis shows the applied input-rate difference $r_A - r_B$. The vertical axis shows the median measured natural detuning $\Delta f_0 = f_A^0 - f_B^0$ from Equation 1; bars span the interquartile range across five matched seeds. These measured values, rather than the input-rate differences or requested targets, become the detuning coordinates in Figures 4 and 5. Dashed lines mark zero.

The frozen thresholds used in Equation 4 are 0.8 Hz for frequency difference, 11.53 rad/s for absolute phase slope, and 4 complete slips. The grid is identified by checksum 8813df0395d0aeb00eb4613d28f776cc09cb4ff783694a7c7603e749c79f3c0f.

Representative trajectories distinguish locking from drift

Conditions selected before inspection show the transition in time: relative phase remains bounded after locking and drifts at the immediately preceding coupling.

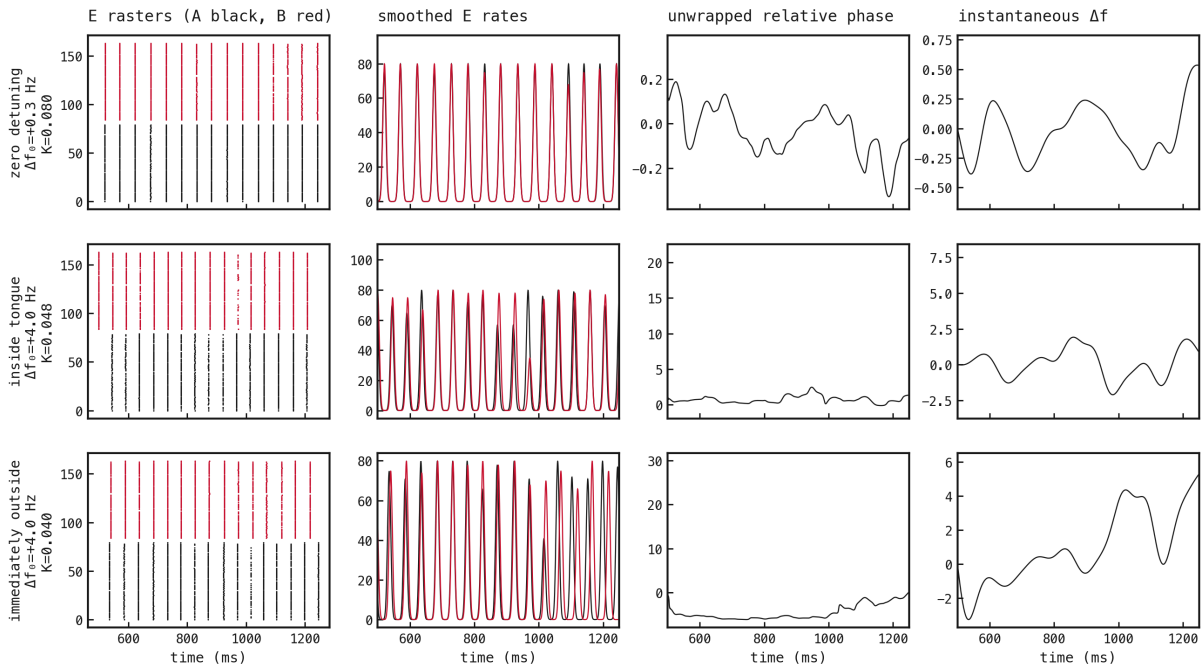


Figure 96: Predetermined time-domain checks. Rows show zero detuning at maximum coupling, the first locked positive-detuning condition, and the same detuning at the immediately preceding coupling. Columns show E rasters, 5 ms-smoothed E rates in hertz, unwrapped relative phase $\varphi(t)$ from Equation 2 in radians, and instantaneous frequency difference in hertz. Relative phase is bounded in the locked conditions and drifts immediately outside the boundary.

Component estimators localize the locking transition

All 715 primary trials are valid. Across the grid, low coupled-frequency difference, low relative-phase drift, few phase slips, and high phase concentration emerge together as coupling increases.

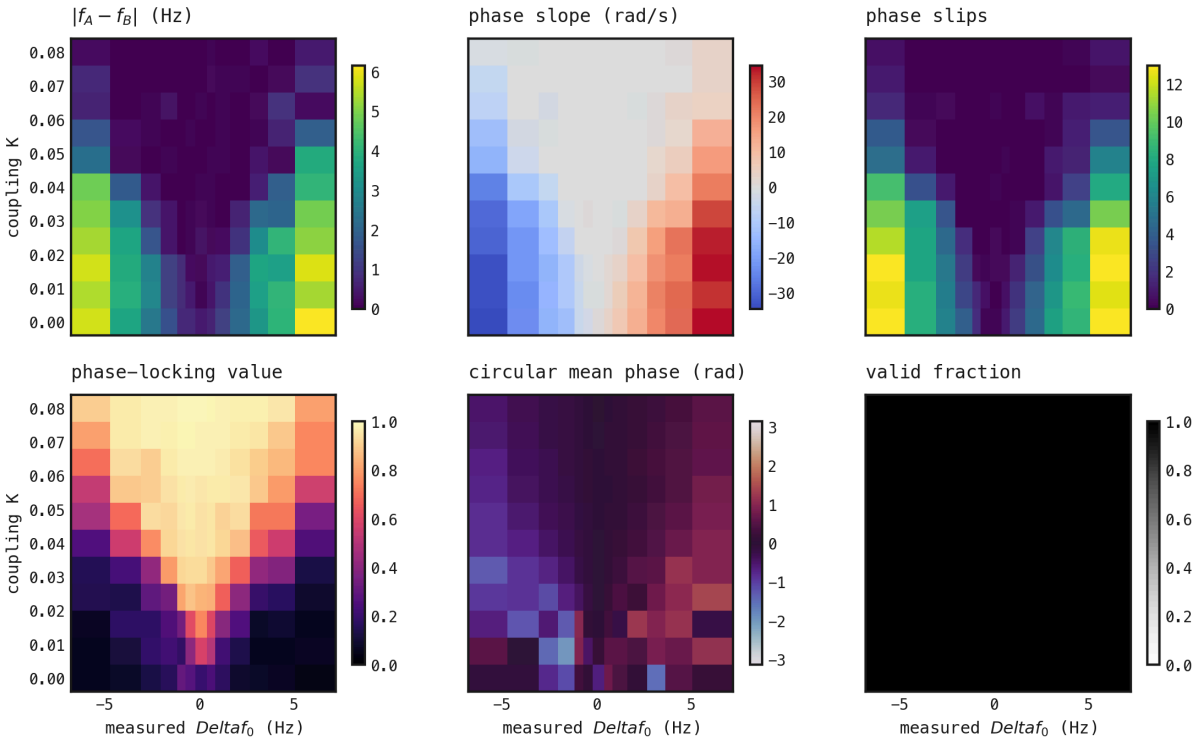


Figure 97: Component estimators on the primary grid. All panels share measured natural detuning in hertz horizontally and coupling K vertically. Panels report coupled frequency difference in hertz, fitted slope of Equation 2 in radians per second, complete phase slips, PLV R from Equation 3, circular mean phase in radians, and valid-trial fraction. Validity is 100% across the grid, while low frequency difference, low phase drift, few slips, and high PLV coincide in the region subsequently classified as locked in Figure 5.

Coupling produces a widening Arnold tongue

Applying Equation 4 to the component estimators produces a contiguous locked region centred near zero detuning. Its width reaches 11.92 Hz at the largest coupling and increases across 4 successive coupling steps.

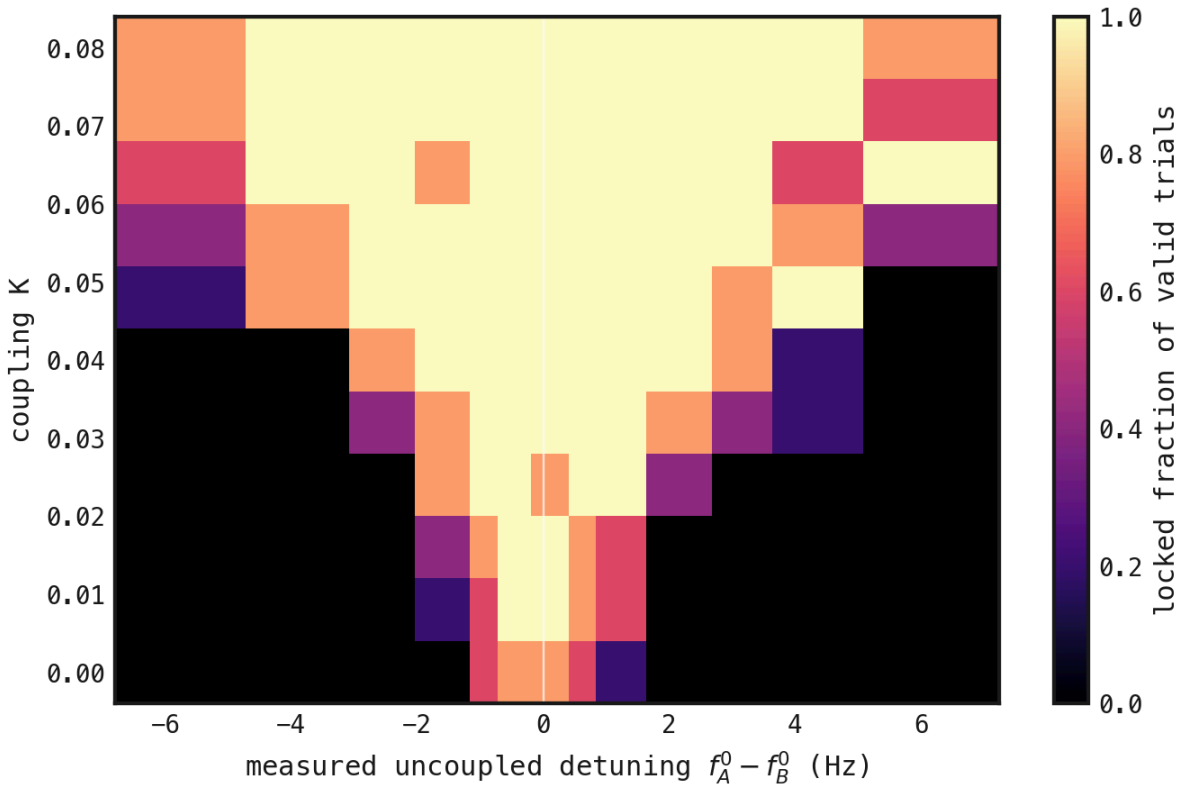


Figure 98: Primary Arnold tongue. Measured natural detuning Δf_0 from Equation 1 is on the horizontal axis in hertz; reciprocal coupling K is on the vertical axis; colour is the fraction of valid trials that satisfy Equation 4. The centred locked region widens with coupling, satisfying the registered geometric clause.

The strict 80 E / 20 I verdict fails one phase-sign cell

Method 6 gives a split result. Geometry and validity pass, but one qualifying cell has the wrong phase sign. Agreement is 98.7%, while the registered clause requires every qualifying cell to agree. The overall 80 E / 20 I reproduction therefore failed. The exception lies near the estimator's registered resolution, so Method 7 tests that boundary rather than repeating the full grid.

The 800 E / 200 I confirmation resolves the exception

The focused confirmation passes: all 40 coupled trials are valid and locked under Equation 4, and all 40 have the phase sign required by Equation 1 and Method 6. At the disputed negative-detuning cell, all 10 seeds have the correct sign with mean phase −0.543 rad. The mirrored positive condition gives 0.526 rad.

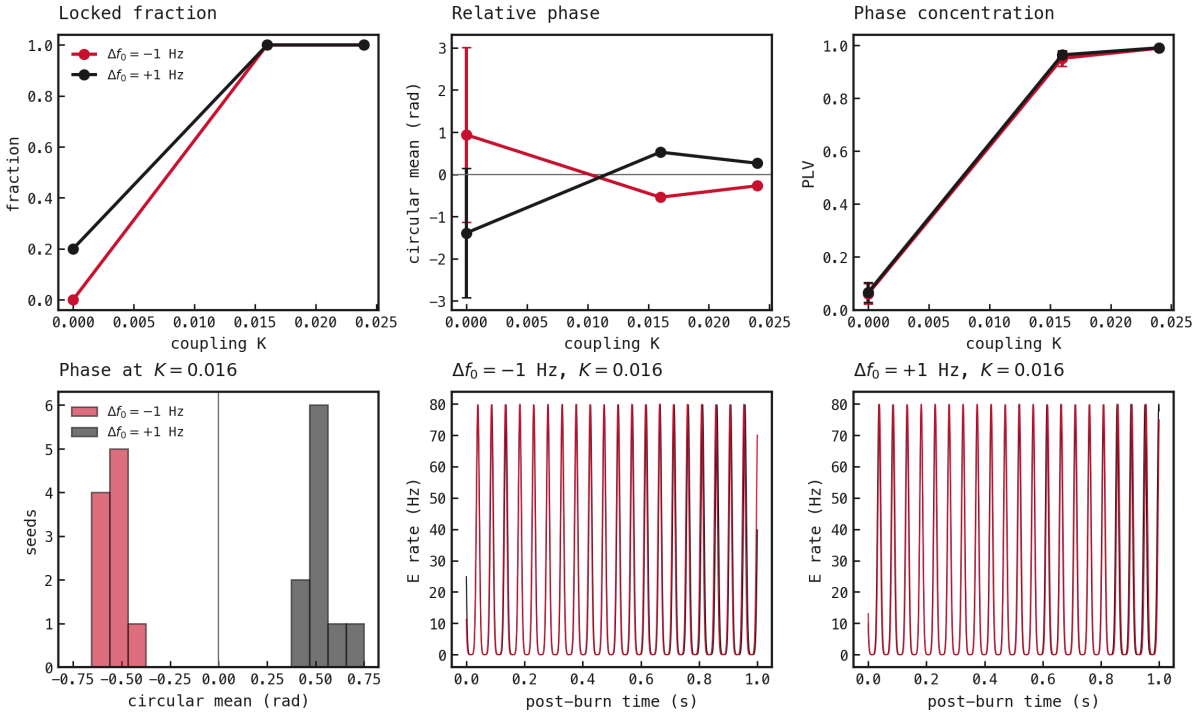


Figure 99: Focused 800 E / 200 I finite-size confirmation. Top panels show locked fraction, circular mean relative phase in radians, and PLV R from Equation 3 over coupling K for mirrored natural detunings; error bars are one standard deviation across 10 paired seeds. The lower-left panel isolates phase at the disputed coupling. The remaining panels show post-burn E-population rates in hertz (circuit A black, circuit B red). Both detuning signs lock with mirrored phase ordering, resolving the lone 80 E / 20 I exception.

The follow-up supports finite-size or estimator-resolution noise, rather than a systematic phase-ordering contradiction, as the cause of the original exception.

Evidence scale

Dense primary retention would occupy 27.82 GB. The published exact-event and state-summary archive occupies 165.5 MB, a 168.1-fold reduction. The 143 primary cells required a median 19.03 s each locally.

References

1. Lowet, Roberts, Hadjipapas, Peter, van der Eerden & De Weerd (2015): *Input-Dependent Frequency Modulation of Cortical Gamma Oscillations Shapes Spatial Synchronization and Enables Phase Coding*. PLOS Computational Biology. doi:10.1371/journal.pcbi.1004072

Empirical input-rate calibration for variable-rate PING training

10 August 2026

Abstract

This experiment asks which input-rate interval preserves enough digit information after synaptic and membrane filtering to justify using it in a variable-rate PING training run. We trained a nonlinear decoder on freshly simulated MNIST features spanning eight rates, then evaluated the frozen decoder on held-out images at each rate. The selected interval is 0.5 to 25 Hz. Its lower edge is the first tested rate at which all three decoders meet or exceed 50% held-out accuracy. The result calibrates this feature representation and decoder; it does not measure PING-network accuracy.

Methods

1. **Partition the MNIST dataset.** The official MNIST training partition contains 60000 images. We assigned its first 20000 images to decoder training and the next 5000 images to checkpoint selection. The remaining 35000 images were not used. Final evaluation used the first 5000 images from the separate official test partition. Training, validation, and test images therefore did not overlap.
2. **Simulate filtered image features.**

   Each normalized pixel intensity x_i generated an independent binary event at every $\Delta t = 0.1$ ms timestep,

   $$S_i(t) \sim \text{Bernoulli}\left(r x_i \Delta \frac{t}{1000}\right). \quad (1)$$

   Here r is the maximum-pixel encoding rate in spikes/s. Conductance followed

   $$g_i(t) = \exp\left(-\Delta \frac{t}{\tau_{\text{AMPA}}}\right) g_i(t - \Delta t) + w S_i(t), \quad (2)$$

   with $\tau_{\text{AMPA}} = 2$ ms and $w = 1.2$ μS. The non-spiking conductance-based membrane obeyed

   $$C_E \frac{dv_i}{dt} = g_{\text{L,E}}(E_L - v_i) + g_i(t)(E_e - v_i), \quad (3)$$

   with $C_E = 1$ nF, $g_{\text{L,E}} = 0.05$ μS, $E_L = -65$ mV, and $E_e = 0$ mV. Conductance began at zero and voltage at E_L. The decoder feature was the baseline-subtracted voltage averaged over the complete 200 ms presentation,

   $$z_i = \frac{1}{T} \int_0^T (v_i(t) - E_L)\, \mathrm{d}t. \quad (4)$$

   Every training, validation, illustration, and test feature used a newly drawn spike train and direct evaluation of Equations 1–4. The random spikes form *shot noise*: each spike produces a discrete jump in conductance, followed by the exponential AMPA decay in Equation 2. These conductance pulses alter both the voltage toward which the membrane moves and the speed at which it moves there. The effect of a spike therefore depends on

the voltage and conductance left by earlier spikes, rather than adding a fixed voltage increment.

Direct simulation is particularly important at low input rates. A finite presentation may contain no spikes, one spike, or a few spikes arriving at different times. The response statistics can consequently change during the presentation (*nonstationary*) and the distribution across presentations can be asymmetric or concentrated around a few distinct outcomes (*non-Gaussian*) [**1**]. Evaluating Equations 1–4 for each presentation preserves these effects instead of replacing them with a steady, bell-shaped approximation.

3. **Train a mixed-rate decoder.**

   The official MNIST training partition supplied the first 20000 images for training and the next 5000 for validation. Every presentation sampled one of the eight rates 0.1, 0.25, 0.5, 1, 2, 5, 10, 25 Hz uniformly and independently. A 784–1024–10 ReLU decoder was trained with cross-entropy and Adam for 10 epochs. Seeds 42, 43, 44 defined independent initializations, rate assignments, and spike trains. Validation accuracy selected one checkpoint per seed.

4. **Evaluate held-out accuracy and select the interval.**

   1. **Select the held-out images.** We took the first 5000 images from the official MNIST test partition. These images were not used for training or checkpoint selection.
   2. **Simulate shared test features.** We simulated every held-out image once at each registered input rate. All three validation-selected decoders received the same simulated feature for a given image and rate, so differences among decoders were not caused by different spike realizations.
   3. **Measure each decoder's accuracy.** For each input rate and decoder, we divided the number of correctly classified test images by 5000. This produced one held-out accuracy per decoder at each rate.
   4. **Select the interval.**

      The practical floor was the lowest tested rate at which every decoder met or exceeded 50% accuracy,

      $$r_{\text{train}} = \min\Big\{ r \in \mathcal{R} : \min_{s \in \mathcal{S}} A_s(r) \geq 0.5 \Big\}. \quad (5)$$

      Here $\mathcal{R}$ is the set of tested rates, $\mathcal{S}$ is the set of decoder seeds, $A_{s(r)}$ is held-out accuracy for decoder s at rate r, and r_{train} is the selected training floor. The upper edge was the highest tested rate. We did not interpolate between tested rates.

Results

Decoder training

Validation accuracy improved across training for all three decoder seeds.

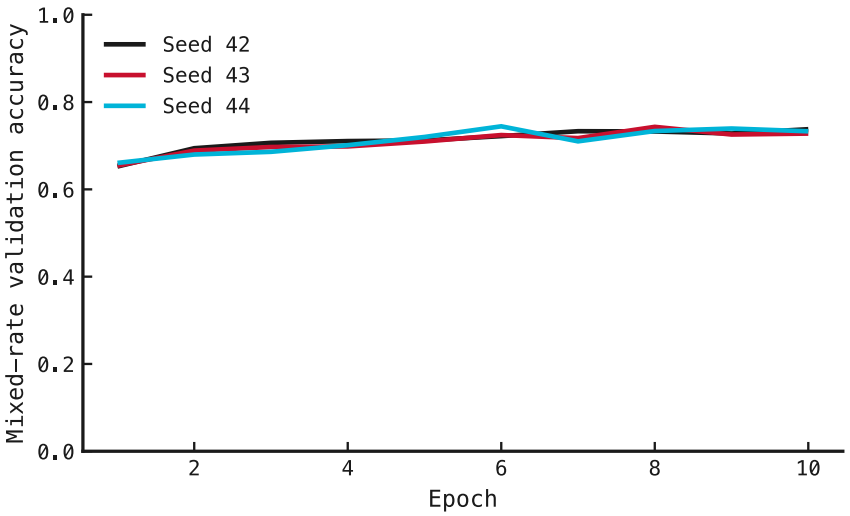


Figure 100: Mixed-rate validation accuracy across training. The horizontal axis is epoch and the vertical axis is validation accuracy. Each curve is one independently trained nonlinear decoder, and every epoch uses fresh direct feature simulations. All three decoders improve before checkpoint selection.

What the decoder saw

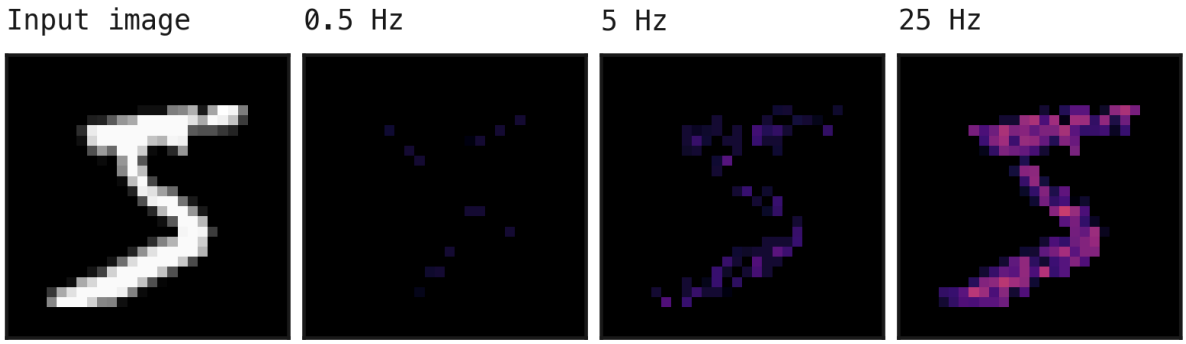


Figure 101: An MNIST input image and its directly simulated filtered features. The left panel shows the normalized input. The remaining panels show independent spike realizations at maximum-pixel encoding rates of 0.5 Hz, 5 Hz, and 25 Hz. Each simulation uses a 1.2 μS synaptic conductance and a 200 ms presentation. Greater input rate preserves more of the digit's spatial structure.

Sparse presentations retained fragments of the digit rather than a uniformly attenuated image. Increasing rate filled in the spatial pattern and reduced the importance of individual event times.

Empirical rate selection

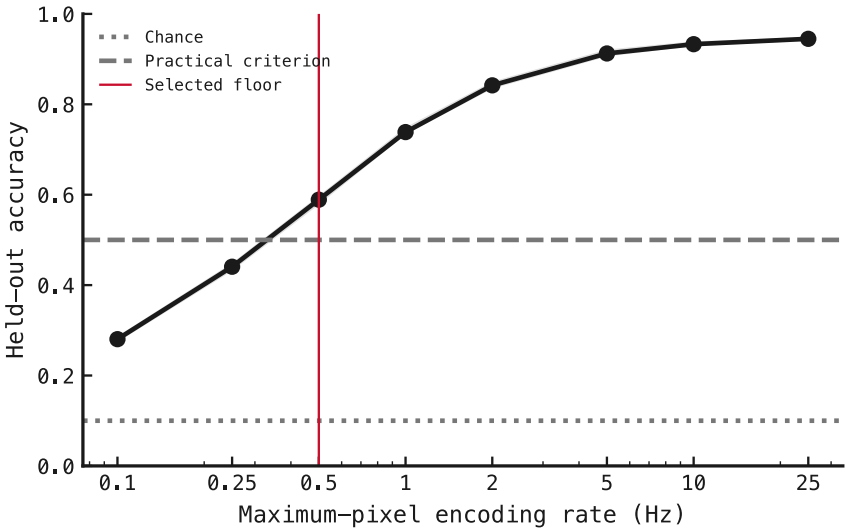


Figure 102: Held-out nonlinear-decoder accuracy against maximum-pixel encoding rate. Points average the official test images and three independently trained decoders. The band spans the lowest and highest decoder accuracy at each rate. Horizontal rules mark chance and the 50% practical criterion. The red vertical rule marks the first tested rate at which all three decoders meet that criterion, which defines the selected floor.

All three decoders first met the practical 50% criterion at 0.5 Hz. Mean accuracy at that condition was 58.9%. We therefore select 0.5 to 25 Hz for later variable-rate PING training.

Conclusion

The filtered MNIST representation retained usable digit information from 0.5 Hz upward. All three independently trained decoders met the 50% held-out accuracy criterion at the selected lower bound, and performance continued to improve across the tested range. We therefore carry 0.5 to 25 Hz forward as the empirical input-rate interval. This is a decoder-based calibration, not a measurement of PING-network performance, so the interval must still be checked in the network for which it was selected.

Relation to prior work

Neural decoding measures information accessible to a specified readout, not an absolute information content or a mechanistic account of the encoded population [**2**]. We therefore interpret the ANN psychometric curve only as a decoder-relative calibration. Nonstationary filtered-shot-noise theory motivates direct simulation of the conductance and membrane dynamics [**1**].

References

1. Brigham & Destexhe: *Nonstationary Filtered Shot-Noise Processes and Applications to Neuronal Membranes*. Physical Review E, 2015. doi:10.1103/PhysRevE.91.062102
2. Quian Quiroga & Panzeri: *Extracting Information from Neuronal Populations: Information Theory and Decoding Approaches*. Nature Reviews Neuroscience, 2009. doi:10.1038/nrn2578

Linear-filter analysis of sparse conductance-driven pixel features

10 August 2026

Abstract

We tested what stationary linear-filter theory explains about a sparse conductance-driven pixel feature, and where the approximation fails. We simulated how an AMPA synapse, a conductance-based membrane, and finite-time averaging transform a fully active pixel driven from 0 to 25 Hz. The model explains the flat response to slow input fluctuations, attenuation of fast fluctuations, and lobes caused by temporal averaging.

The model does not accurately predict feature magnitude: it overpredicts the mean, and its standard deviation peaks too early and has the wrong shape. Sparse responses depend strongly on the discrete number and timing of spikes, violating the model's assumption of small, stationary fluctuations. The theory explains the filtering, but direct simulation remains necessary for quantitative predictions.

Methods

1. **Simulate the finite-window feature.**

   We fixed the normalized pixel intensity at $x = 1$ and varied only its input rate r from 0 to 25 spikes/s. At each timestep, the pixel generated an event with probability

   $$p_{\text{event}} = \frac{r\Delta t}{1000}. \quad (1)$$

   Here r is the input rate in spikes/s and $\Delta t = 0.1$ ms. During a presentation of duration $T = 200$ ms, the expected number of events is $r\frac{T}{1000}$. Expected event count is therefore a consequence of the chosen rate, not a separate experimental variable.

   Conductance and membrane voltage followed

   $$g(t) = \beta g(t - \Delta t) + wS(t), \quad (2)$$

   $$\beta = \exp\left(-\Delta \frac{t}{\tau_{\text{AMPA}}}\right), \quad (3)$$

   $$C\frac{dv}{dt} = g_L(E_L - v) + g(t)(E_e - v). \quad (4)$$

   Here $S(t)$ is one when an event occurs and zero otherwise, $g(t)$ is AMPA conductance, w is the conductance increment per event, β is the per-timestep decay factor, and τ_{AMPA} is the AMPA decay time constant. The membrane voltage is $v(t)$, C is membrane capacitance, g_L is leak conductance, E_L is the leak reversal potential, and E_e is the excitatory reversal potential.

   We used $C = 1$ nF, $g_L = 0.05$ µS, $E_L = -65$ mV, $E_e = 0$ mV, $\tau_{\text{AMPA}} = 2$ ms, and independent probe conductances $w \in \{0.6, 1.2, 2.4\}$ µS. Every presentation began from $g(0) = 0$ and $v(0) = E_L$. Its scalar feature was

   $$z = \frac{1}{T}\int_0^T (v(t) - E_L)\,\mathrm{d}t. \quad (5)$$

Here z is the baseline-subtracted mean voltage and T is presentation duration.

For the mean and SD estimates in Figure 1, we generated 512 new presentations at each of 101 input rates and each conductance. For the full response distributions in Figure 2, we generated 4096 new presentations at each selected rate. Direct simulation retains the nonstationary finite-window behaviour of filtered conductance shot noise [**2**].

2. **Calculate the stationary operating point.**

   For stationary Poisson rate r, filtered-shot-noise theory [**1**] gives the mean AMPA conductance

$$\bar{g}_r = rw\frac{\tau_{\mathrm{AMPA}}}{1000}. \quad (6)$$

   Replacing the fluctuating conductance by this mean gives the operating-point voltage

$$\bar{v}_r = \frac{g_L E_L + \bar{g}_r E_e}{g_L + \bar{g}_r}. \quad (7)$$

   Because this operating point is constant across the window, its predicted mean feature is

$$\mu_{\mathrm{linear}}(z) = \bar{v}_r - E_L. \quad (8)$$

   An overbar denotes a stationary mean, $\bar{g}_r$ is mean conductance at rate r, $\bar{v}_r$ is the corresponding mean voltage, and $\mu_{\mathrm{linear}(z)}$ is the resulting linear prediction for mean feature value. Appendix A.1 derives Equations 6–8 from the continuous conductance and membrane equations.

3. **Derive the local frequency response.**

   Linearizing Equations 2–4 around the operating point gives the response from input-rate fluctuation to voltage fluctuation,

$$G_r(\omega) = \frac{w}{i\omega + \frac{1}{\tau_{\mathrm{AMPA}}}} \cdot \frac{E_e - \bar{v}_r}{i\omega C + g_L + \bar{g}_r}. \quad (9)$$

   Here ω is angular frequency in rad/ms and i is the imaginary unit. The transfer function $G_{r(\omega)}$ maps a small input-rate fluctuation to its voltage response around the operating point. Appendix A.2 derives Equation 9 by linearizing the synaptic and membrane equations.

   Averaging voltage over the finite presentation contributes

$$A_T(\omega) = \frac{1 - \exp(-i\omega T)}{i\omega T}, \quad (10)$$

   so the complete input-to-feature response is

$$H_r(\omega) = A_T(\omega) G_r(\omega). \quad (11)$$

   Here $A_{T(\omega)}$ is the transfer function of averaging over duration T, and $H_{r(\omega)}$ is the complete input-to-feature transfer function. Appendix A.3 derives Equations 10 and 11.

   To generate Figure 3, we evaluated Equations 9 and 11 from 0.1 to 200 Hz at 0.25, 3, 25 spikes/s for the nominal 1.2 µS probe. Frequency in Hz was converted using $\omega = 2\pi\frac{f}{1000}$. The Bode magnitude was reported relative to the low-drive DC response,

$$M_X(f) = 20 \log_{10} \left(\frac{\left| X_r\left(2\pi \frac{f}{1000}\right) \right|}{\left| X_{r_{\text{low}}}(0) \right|} \right). \quad (12)$$

Here $M_{X(f)}$ is Bode magnitude in dB, $X_r = G_r$ for Figure 3A, $X_r = H_r$ for Figure 3B, f is modulation frequency in Hz, and r_{low} is the lowest operating rate used as the DC reference.

4. **Calculate the linearized feature variance.**

   The centred ideal Poisson input has a white two-sided spectrum on the millisecond time base,

   $$S_{\text{in}}(\omega) = \frac{r}{1000}. \quad (13)$$

   The feature spectrum and variance are therefore

   $$S_z(\omega) = |H_r(\omega)|^2 S_{\text{in}}(\omega), \quad (14)$$

   $$\text{Var}_{\text{linear}}(z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} |H_r(\omega)|^2 S_{\text{in}}(\omega)\, \mathrm{d}\omega. \quad (15)$$

   Here $S_{\text{in}(\omega)}$ is the two-sided input power spectrum, $S_{z(\omega)}$ is the feature power spectrum, and $\text{Var}_{\text{linear}(z)}$ is the predicted feature variance. Appendix A.4 derives Equations 13–15 from the Poisson spectrum and the linear-filter variance identity.

   We integrated Equation 15 numerically on a logarithmic frequency grid. A second calculation with half as many points measured quadrature refinement.

Results

Empirical finite-window response

Mean response increased with input rate and event strength. Variability first increased as spike count and timing diversified, then flattened or declined as relative count fluctuations, shunting, and voltage saturation became more important.

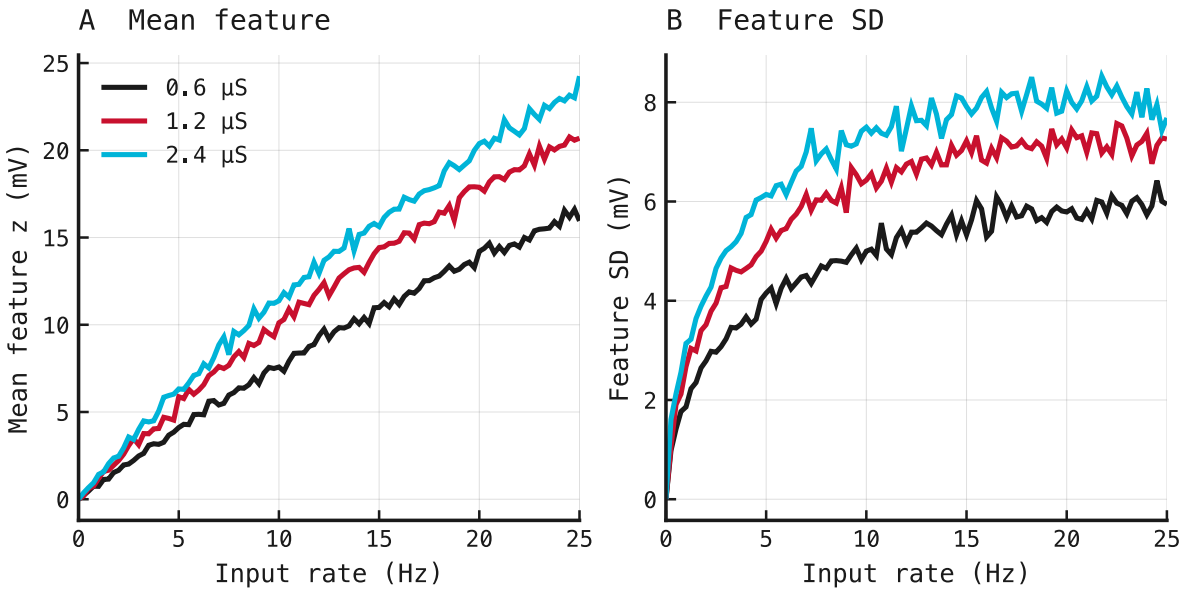


Figure 103: Empirical finite-window response of a fully active pixel over the input-rate grid. Both horizontal axes show input rate in spikes/s. Panel A shows mean feature in mV; Panel B shows feature SD in mV. Each coloured curve contains 101 rate conditions. At every rate, the plotted mean or SD summarizes 512 independently simulated presentations; the figure does not display those individual presentations as points. Black, red, and cyan denote 0.6, 1.2, 2.4 µS conductance increments. Mean response rises with rate, while variability reaches a broad maximum or plateau.

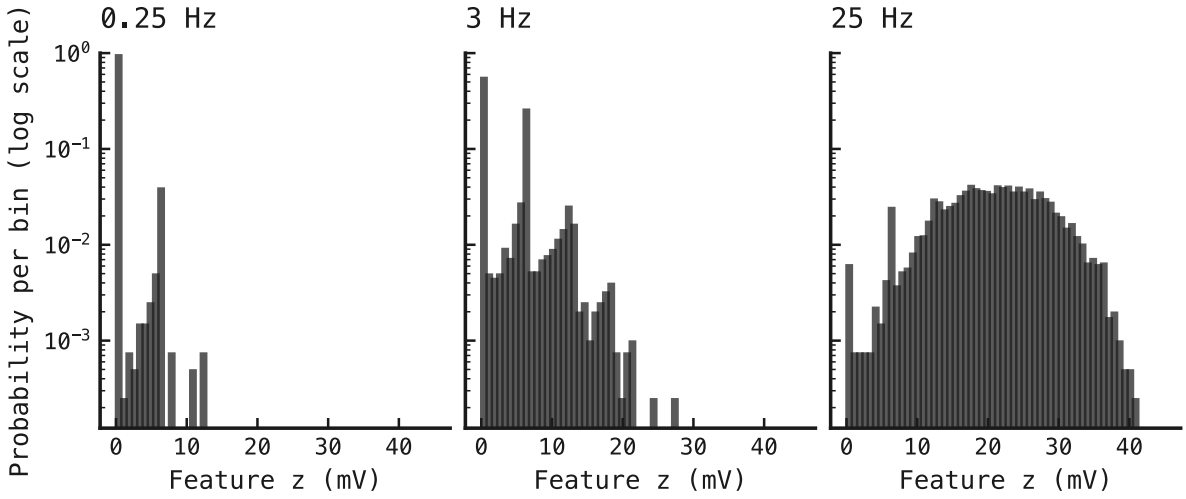


Figure 104: Empirical feature distributions at 0.25, 3, 25 spikes/s for the nominal 1.2 µS probe. Each panel contains 4096 new presentations. The horizontal axes show feature value in mV. Bars report probability per common fixed-width bin on a shared logarithmic vertical axis. Sparse input produces an atom at rest and separated timing-dependent responses; increasing rate produces a smoother distribution.

Input rate fixes only the average number of events. Individual presentations can still contain no events, one event, or several events. Even when two presentations contain the same number, different event times change the finite-window average.

Analytical Bode-magnitude response

The synapse and membrane passed slow modulation but attenuated fast modulation. Finite-window averaging superimposed regularly spaced zeros and lobes on that falling response.

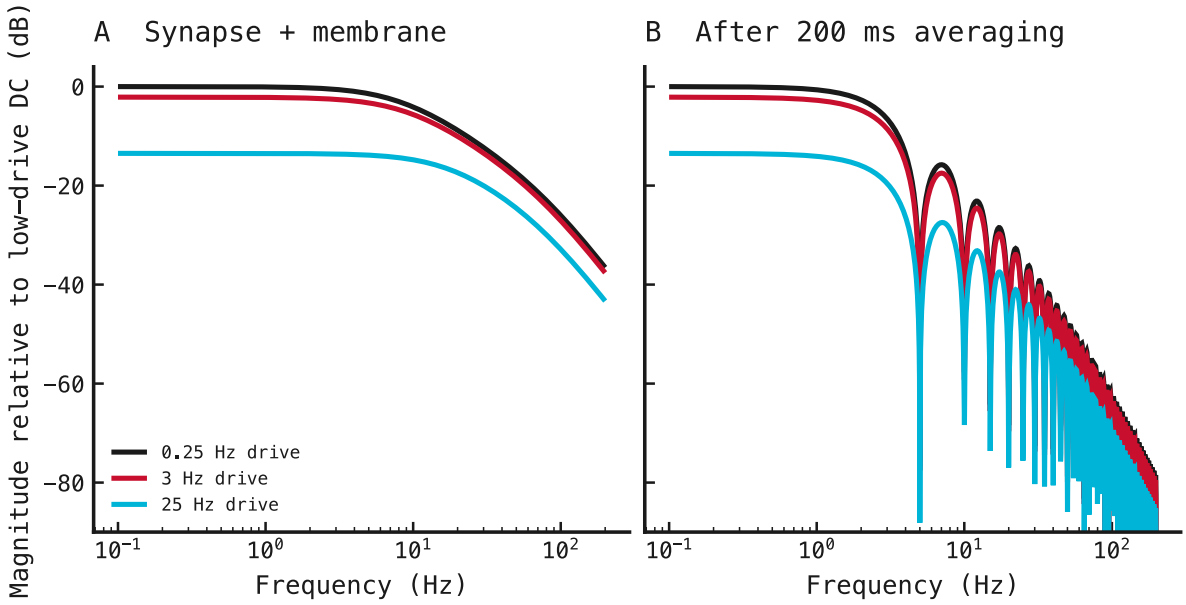


Figure 105: Analytical Bode-magnitude plots at the nominal 1.2 µS probe. Black, red, and cyan denote operating rates 0.25, 3, 25 spikes/s. The horizontal axis is modulation frequency in Hz and the vertical axis is gain relative to the low-drive direct-current response in decibels. Panel A shows Equation 9; Panel B shows the complete 200 ms window-averaged response from Equation 11.

In Panel A, each curve is flat at low frequency because the AMPA conductance and membrane voltage can follow a slowly varying input quasi-statically. Greater drive lowers that plateau by shunting the membrane and reducing the excitatory driving force. The curves bend downward once modulation becomes too fast for the synaptic and membrane timescales to follow.

Panel B is the same synapse-plus-membrane response multiplied by the rectangular-window response. With $T_s = \frac{T}{1000}$ s,

$$\left| A_T\left(2\pi \frac{f}{1000}\right) \right| = \left| \frac{\sin(\pi f T_s)}{\pi f T_s} \right|. \quad (16)$$

Its zeros at $f = \frac{n}{T_s}$ cancel modulation containing an integer number of cycles within the presentation. Here n is any nonzero integer. The absolute sinc response creates the lobes, while the Panel A low-pass response supplies their falling envelope. Appendix A.3 derives Equation 16 from the finite-window averaging kernel.

Analytical and empirical moments

The stationary model reproduced the mean curve's broad shape but not its magnitude, and it failed to reproduce the shape of the empirical standard deviation.

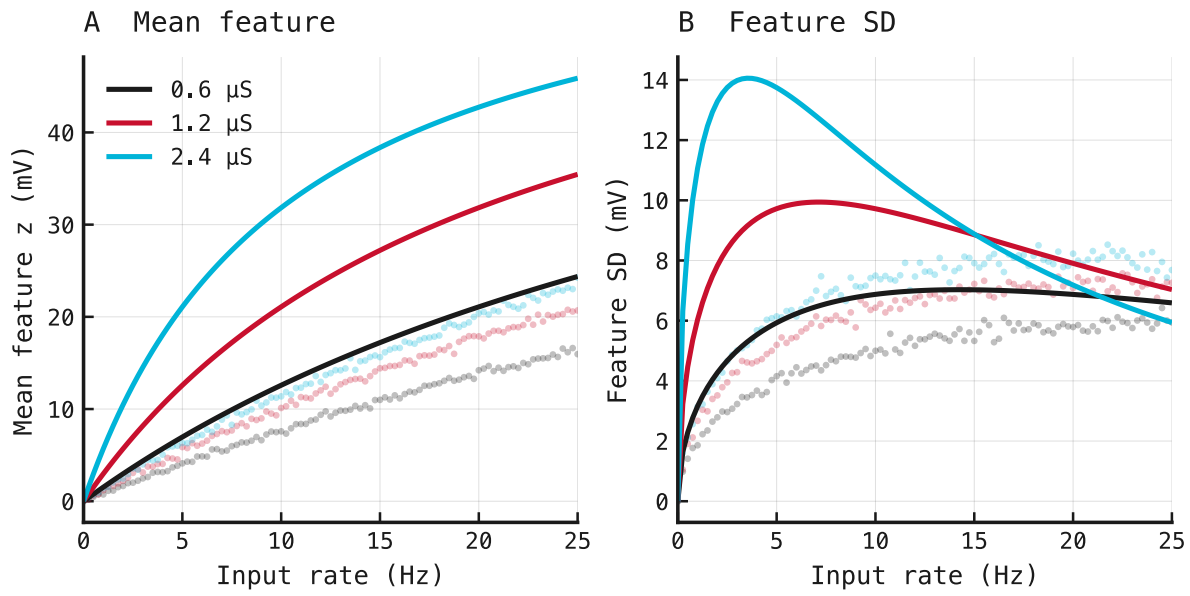


Figure 106: Analytical and empirical moments over input rate for a fully active pixel. The horizontal axis is input rate in spikes/s. Solid curves are stationary predictions from Equations 8 and 15; faint points are independently simulated finite-window estimates. Panel A shows mean feature in mV and Panel B feature SD in mV. The stationary theory overpredicts the mean and places the SD maximum too early.

The analytical mean preserved the broad curvature and conductance ordering but exceeded the empirical magnitude. Its median predicted-to-empirical ratio was 1.972. The stationary calculation evaluates the response at mean conductance, $z(E[g])$, whereas simulation estimates the response over random conductance paths, $E[z(g)]$. Saturation makes these unequal, approximately

$$E[z(g)] < z(E[g]).$$

The stationary input also acts throughout the window; a real late event contributes only briefly to Equation 5.

The SD mismatch is more fundamental. The empirical distribution is a spike-count and spike-time mixture,

$$p_Z(z) = \sum_{n=0}^{\infty} P(N = n) p_{Z|N}(z|n). \quad (17)$$

Here Z is the random feature value, N is the event count, $P(N = n)$ is the probability of observing n events, and $p_{Z|N}(z|n)$ is the feature distribution conditional on that count.

Equation 15 instead treats input as a small stationary fluctuation around one operating point. At low rates, zero-event trials remain exactly at rest, one large event produces a timing-dependent response, and multiple events interact through shunting and saturation. That non-Gaussian mixture explains why the analytical SD rises too sharply and peaks too early, especially for the largest conductance increment.

Conclusion

Stationary linear-filter theory correctly explains the low-frequency plateau, high-frequency roll-off, averaging-window zeros, and qualitative operating- point dependence. It does not quantitatively predict finite-window moments in the sparse, large-jump regime. Direct simulation is therefore required for empirical rate selection, while the analytical model remains useful for explaining the filter's structure.

Limitations

The empirical moment grid used 512 presentations per condition; it was designed to resolve the qualitative theoretical comparison, not to estimate extreme distribution tails. The analytical variance assumes ideal continuous-time Poisson drive and a local stationary linearization, whereas simulation uses discrete Bernoulli events and begins from rest.

Reproducibility

`uv run python experiments/exp081.py` regenerates every sample, summary, and figure. The runner contains a complete independent physical specification and consumes no artifacts or code from another experiment.

Appendix A: derivation of the analytical filter

A.1 Stationary conductance and voltage

Write the continuous AMPA equation driven by an event train $s(t) = \sum_k \delta(t - t_k)$ as

$$\frac{dg}{dt} = -\frac{g}{\tau_{\mathrm{AMPA}}} + ws(t). \quad \text{(A1)}$$

Here $s(t)$ is the impulse train, t_k is the time of event k, and $\delta(t - t_k)$ is a Dirac impulse at that time.

For stationary input rate r spikes/s, the rate on the millisecond time base is $\frac{r}{1000}$. Taking expectations of Equation A1 and setting the stationary derivative to zero gives

$$0 = -\frac{\bar{g}_r}{\tau_{\mathrm{AMPA}}} + w\frac{r}{1000},$$

hence Equation 6. The stationary membrane equation is

$$0 = g_L(E_L - \bar{v}_r) + \bar{g}_r(E_e - \bar{v}_r), \quad \text{(A2)}$$

and collecting the terms multiplying $\bar{v}_r$ gives Equation 7.

A.2 Synapse-plus-membrane linearization

Decompose each quantity into its operating point and a small fluctuation,

$$g = \bar{g}_r + \delta g, \quad v = \bar{v}_r + \delta v, \quad s = \frac{r}{1000} + \delta s.$$

Substitution into Equation A1 and cancellation of the stationary terms gives

$$\frac{d\delta g}{dt} = -\delta\frac{g}{\tau_{\mathrm{AMPA}}} + w\delta s. \quad \text{(A3)}$$

With Fourier convention $\frac{d}{dt} \to i\omega$,

$$\delta g(\omega) = \frac{w}{i\omega + \frac{1}{\tau_{\mathrm{AMPA}}}} \delta s(\omega). \quad \text{(A4)}$$

Substitute the decompositions into Equation 4. After cancelling Equation A2 and discarding the second-order product $\delta g \delta v$,

$$C\frac{d\delta v}{dt} = -(g_L + \bar{g}_r)\delta v + (E_e - \bar{v}_r)\delta g. \quad \text{(A5)}$$

Fourier transformation and substitution of Equation A4 yields Equation 9. Its two denominator factors are the AMPA and membrane low-pass terms; the numerator contains the conductance increment and local excitatory driving force.

A.3 Finite-window average

Linearizing Equation 5 leaves the average of the voltage fluctuation,

$$\delta z = \frac{1}{T}\int_0^T \delta v(t)\,\mathrm{d}t. \quad \text{(A6)}$$

The averaging kernel is $a_T(t) = \frac{1}{T}$ for $0 \le t \le T$ and zero otherwise. Its Fourier transform is

$$A_T(\omega) = \frac{1}{T}\int_0^T \exp(-i\omega t)\,\mathrm{d}t = \frac{1 - \exp(-i\omega T)}{i\omega T},$$

proving Equation 10. Because averaging follows the membrane response, the transfer functions multiply, proving Equation 11. Using $1 - \exp(-ix) = 2i\exp(-i\frac{x}{2})\sin(\frac{x}{2})$ gives

$$|A_T(\omega)| = \left|\frac{\sin\left(\omega\frac{T}{2}\right)}{\omega\frac{T}{2}}\right|,$$

which becomes Equation 16 after substituting $\omega = 2\pi\frac{f}{1000}$ and $T_s = \frac{T}{1000}$.

A.4 Poisson spectrum and feature variance

For ideal Poisson events with rate $\frac{r}{1000}$ per ms, disjoint intervals have independent counts and count variance equals count mean. The centred impulse train therefore has autocovariance

$$R_{\delta s}(\tau) = \left(\frac{r}{1000}\right)\delta(\tau). \quad \text{(A7)}$$

Here $R_{\delta s}(\tau)$ is the input autocovariance at lag τ, and $\delta(\tau)$ is the Dirac delta function.

Fourier transformation of the delta function gives the constant spectrum in Equation 13. A linear time-invariant filter multiplies the input spectrum by squared transfer magnitude, proving Equation 14. Finally, inverse Fourier transformation at zero lag gives

$$\mathrm{Var}(z) = R_z(0) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_z(\omega)\,\mathrm{d}\omega,$$

which, after substituting Equation 14, proves Equation 15. The result is exact for the stated stationary linearized model but only approximate for the nonlinear, start-from-rest, finite-event simulation.

References

1. Wolff & Lindner: *Mean, Variance, and Autocorrelation of Subthreshold Potential Fluctuations Driven by Filtered Conductance Shot Noise*. Neural Computation, 2010. doi:10.1162/neco.2009.02-09-958
2. Brigham & Destexhe: *Nonstationary Filtered Shot-Noise Processes and Applications to Neuronal Membranes*. Physical Review E, 2015. doi:10.1103/PhysRevE.91.062102

Variable-rate streaming with a spike-rate readout

10 August 2026

Abstract

This experiment will test whether PING networks trained across input rates can classify continuous MNIST streams without the fixed-rate and mean-membrane mismatch inherited by exp048. Excitatory spikes drive ten spiking output LIF neurons, and each class logit is the corresponding output neuron's firing rate over a window equal to the current digit's presentation duration. The planned results comprise matched-condition streaming, variable-rate and variable-duration streaming, a 200 ms rate psychometric, and a duration-by-rate accuracy map. Results remain pending until exp022 produces all three variable-rate checkpoints.

Methods

1. Training source

The experiment loads `ping__variable_rate__seed42`, `ping__variable_rate__seed43`, and `ping__variable_rate__seed44` from exp022. Each PING network is trained by sampling one maximum-pixel Poisson rate independently per image presentation from 0.5, 0.75, 1, 1.5, 2, 3, 5, 7.5, 10, 15, and 25 Hz. The recurrent weights and classifier are frozen throughout this experiment.

Note. The exp022 cells are registered and `tools/snn` implements the required output-LIF `spike-rate` training, restore, and output-raster path. Presentation boundaries reset only the output LIF and its counter; the hidden PING state remains continuous. This entry still fails loudly if the variable-rate checkpoint bank is absent or uses another readout.

2. Spike-rate readout

At each timestep, the 1024-element excitatory spike vector $\boldsymbol{s}^{E(t)}$ drives a trained 1024-by-10 projection W_{out} into ten output LIF neurons. For a presentation beginning at timestep a and ending before timestep b, class evidence is

$$z_c = \frac{1}{T}\sum_{t=a}^{b-1} s_c^{\text{out}(t)}, \quad c \in \{0, ..., 9\}, \quad T = (b-a)\Delta\frac{t}{1000}. \quad (1)$$

Here $s_c^{\text{out}(t)}$ is the spike emitted by output LIF neuron c at timestep t, T is the matched presentation duration in seconds, and $\boldsymbol{z}$ contains the ten output firing-rate logits in Hz. The predicted digit is the index of the largest rate. The output neurons therefore have membrane state, leak, threshold, spikes, and reset; unlike `mem-mean`, their membrane voltages are not the logits.

3. Matched inference

Every digit is read from only the spikes generated during its own presentation:

$$T_{\text{readout}} = T_{\text{presentation}}. \quad (2)$$

Here T_{readout} is the spike-summation window and $T_{\text{presentation}}$ is the duration of the corresponding digit. The first protocol holds both at 200 ms and presents a five-digit stream. This checks the trained duration and readout before either input rate or duration is varied.

4. Variable-rate and variable-duration inference
A second five-digit stream changes both presentation duration and maximum-pixel Poisson rate at digit boundaries. Each digit retains its own matched readout window. A factorial evaluation then crosses presentation durations 25, 50, 100, and 200 ms with the eleven training rates. Every cell is evaluated independently for seeds 42, 43, and 44.

5. Psychometric protocol
The rate psychometric fixes presentation and readout duration at 200 ms, varies the maximum-pixel rate across the eleven training values, and reports held-out classification accuracy across the three trained seeds. This isolates rate robustness from shortened evidence windows.

Results

1. Matched-condition stream
TODO. Reproduce the structure of exp048′s streaming figure: excitatory, inhibitory, and output rasters; digit boundaries; and the ten online class-evidence traces. Unlike exp048, every trace will be the cumulative output-spike rate from the start of the current digit: spikes observed so far divided by elapsed time in the matched window.

2. Variable conditions
TODO. Reproduce exp048′s joint temporal and encoding-rate summary using the variable-rate-trained weights. Pair the duration-by-rate accuracy map with the 200 ms psychometric curve, so the effects of rate and available integration time remain distinguishable.

3. Comparison with exp048
TODO. Compare the completed results with exp048. The comparison must be interpretive rather than a silent replacement. Exp048 uses fixed-25-Hz training and reconstructs a sliding `mem-mean` output. This experiment uses mixed-rate training and the trained `spike-rate` readout. Any accuracy difference therefore reflects the combined training-distribution and readout correction, not a single isolated intervention.