A bundle checkpoint replays

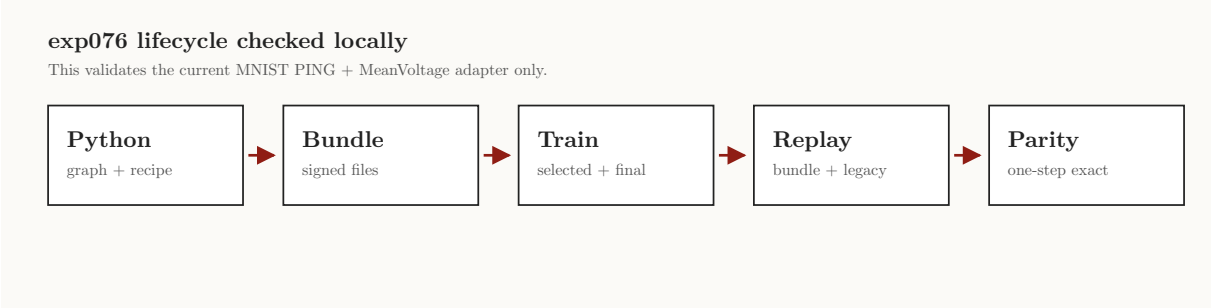
exp076 · 2 August 2026 · Draft

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.

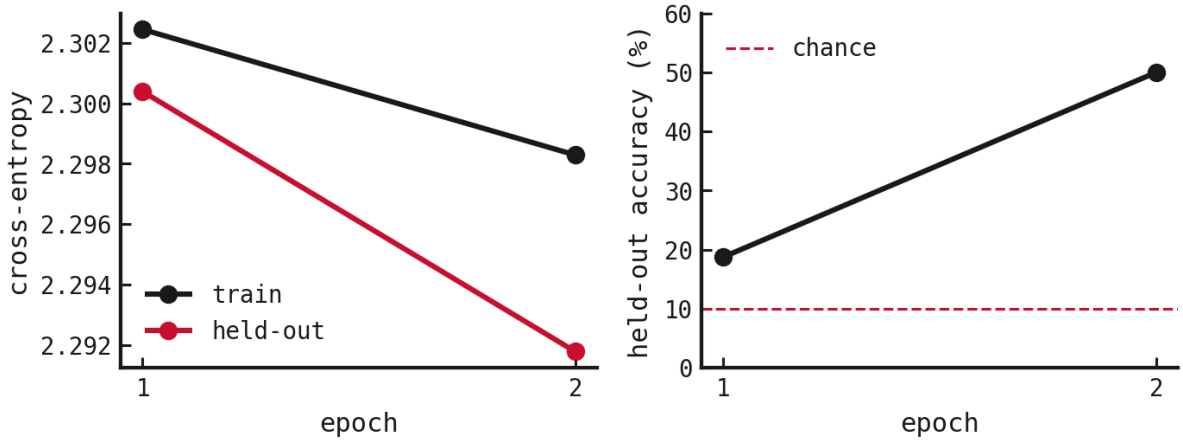
The selected bundle-trained checkpoint replayed through `tools/snn sim --bundle --infer --load-weights` at 50%, exactly matching the trainer's recorded best accuracy. The final checkpoint also replayed exactly. A focused deterministic unit gate separately shows exact bundle-vs-legacy equality for initial 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.