DOCUMENTATION

From Python graph to spikes

exp074 · 31 July 2026 · Draft

Abstract

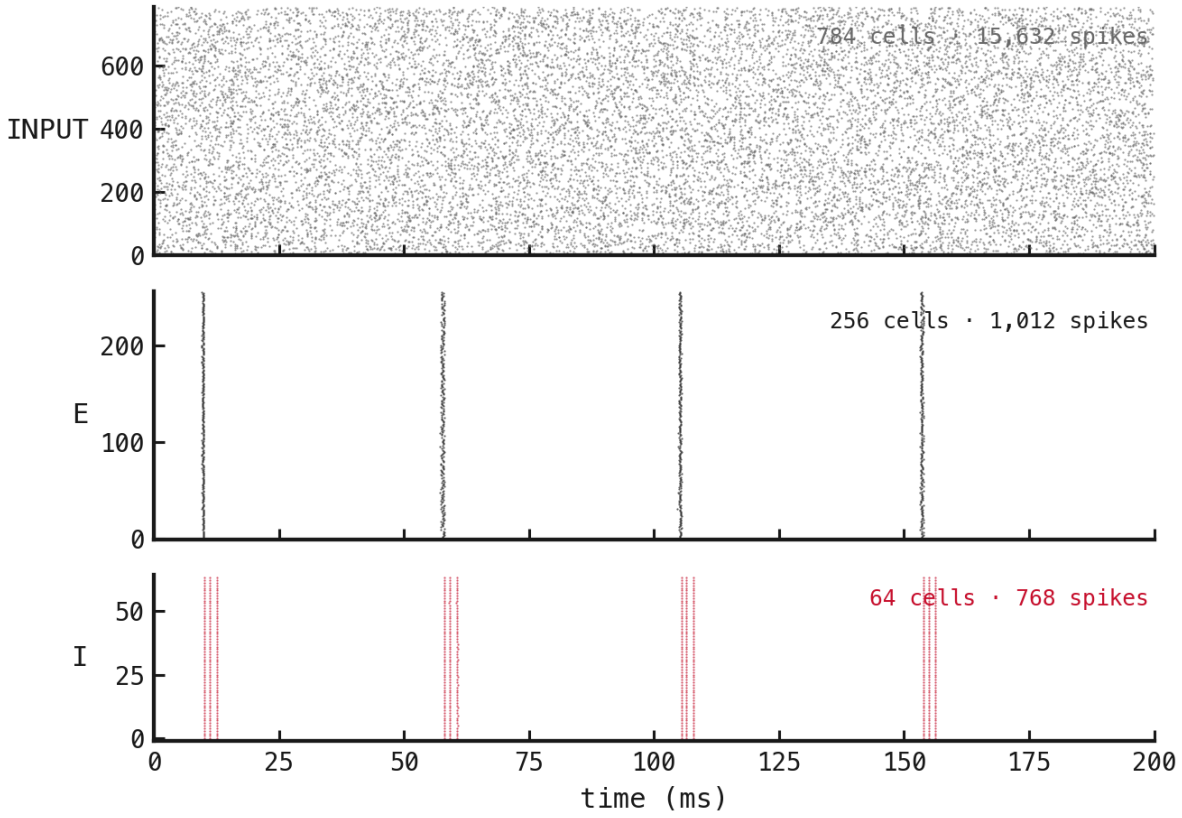
This is the smallest useful end-to-end demonstration of `snnlang`. The experiment runner defines a PING circuit using the Python authoring API, compiles it into a deterministic data-only bundle, and invokes `tools/snn` through its command-line interface. The simulator receives an explicit, saved Poisson 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.