



BindsNET: An ML-oriented spiking networks library built with PyTorch

Daniel J. Saunders, Hananel Hazan, Hassaan Khan, Hava T. Siegelmann, Robert Kozma

What is BindsNET?

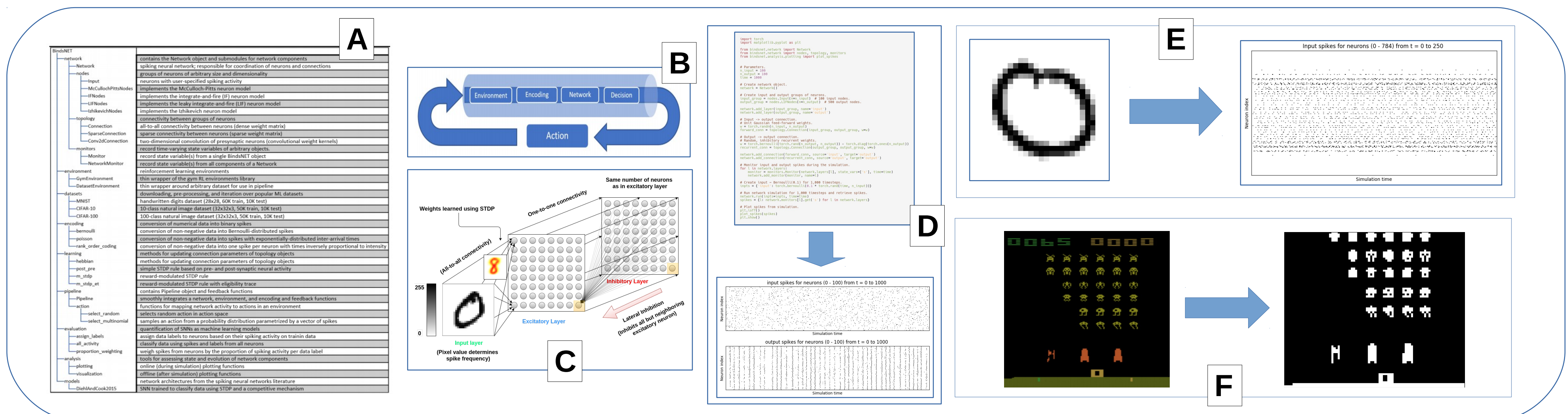
- Clock-driven *spiking neural networks* (SNN) simulation
- Oriented towards ML + RL
- User-friendly syntax + fast prototyping
- *Functional* (rather than *exact*) dynamics
- Run on CPUs, GPUs, or both
- Inherits performance + functionality of PyTorch

How is PyTorch used?

- **torch.Tensor** object: Linear algebra + tensor ops
- **torch.nn** module: Advanced network operations
- **torch.distributions** module: Generating spike data
- **torch.save, load**: Save / load params to / from disk
- **torchvision.datasets**: Planned integration!

What's in the library?

- **Network**: Coordinates simulation of network components
- **Nodes / Connections**: Groups of neurons and connections between them
- **Learning rules**: Hebbian learning, STDP, reward-modulated STDP, back-prop (?)
- **Datasets**: Popular machine learning datasets
- **Encoding**: Converts real-valued data into spikes
- **OpenAI gym integration**: Converts gym environment outputs into spiking inputs
- **Pipeline**: Coordinates a network, dataset / environment, encoding, and action function
- **Plotting**: Interactive plots of state variables during network simulation
- **Models**: Experimental SNN architectures



A: BindsNET package structure; B: Pipeline coordination object; C: Example spiking network architecture; D: Example network building / simulation script and spike outputs; E: Poisson encoding of MNIST digit for 250 timesteps; F: Pre-processed version of OpenAI gym's Space Invaders environment.

Why spiking neurons?

- More *biologically plausible* than ANN neurons
- Useful for modeling neuronal circuits + brains
- Speedup + power reduction on dedicated hardware
- Communicates with all-or-nothing *spikes*
- Naturally incorporates time by integrating input

TO-DO

•

