Convolutional Spiking Neural Networks

Daniel Saunders

July 25, 2018

This document tracks the progress of the BINDS lab on studying convolutional spiking neural networks (C-SNNs). All

Note: All `code` is assumed to be BindsNET syntax.

1 Locally-connected SNNs

TODO.

2 Two-layer convolutional SNNs

2.1 Basic structure

An layer of `Input` neurons (layer `X`) equal to the size of the input data (e.g., MNIST or CIFAR-10) is connected via a `Conv2dConnection` with STDP-modifiable [citation needed] synapses to a layer of `AdaptiveLIFNodes` or `DiehlAndCook2015Nodes` (layer `Y`). Layer `Y` is recurrently connected to itself with inhibitory synapses via a `Connection`.

Layer `Y` may be driven by `bernoulli()`- or `poisson()`-distributed spikes parametrized by the input data; since the networks are operating in an effectively rate-based regime, Bernoulli spikes may be preferred for efficiency reasons. **Note**: Additionally, the implementation of `poisson()` may be incorrect at this stage.

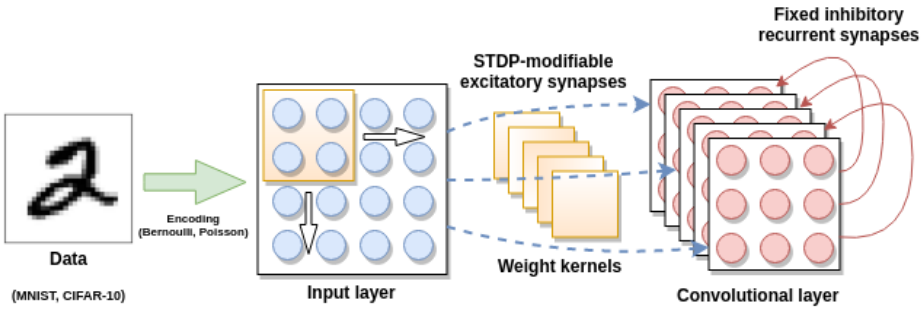

Figure 1: Schematic diagram of the structure of a two-layer convolutional spiking neural network. For ease of presentation, all data, layers, and weight kernels are depicted as two-dimensional (as in the case of gray-scale image processing). Input data encoded into spike trains is used for the activity of an input layer. Weight kernels are "slid" across the input space, multiplied by the input activations, and projected to unique neurons in the corresponding sub-population in the convolutional layer (one per weight kernel). The convolutional layer has inhibitory, recurrent connections, establishing a competition for spiking activity.

The goal of the convolutional layer is to learn a (compact) set of weight kernels (filters) which reliably detect salient features in the input space. The weight kernels "slide" across the input space

with a regular horizontal and vertical *stride*, and are multiplied by the activations at each input location to produce an activation value for a unique neuron in the next layer.

This technique has been successful in convolutional neural networks (CNNs) of deep learning; however, the neural model is much simplified, and the network can be efficiently trained using optimization techniques (stochastic gradient descent (SGD)). In the case of SNNs, the input activations are converted into spike trains, and the output of the convolution can be interpreted as current to be injected into the post-synaptic layer.

2.2 Recurrent connectivity

There are some variants of connectivity of the recurrent connection for layer `Y`.

2.2.1 Inhibit all neurons with same receptive field

2.2.2 Inhibit all other neurons

In the spirit of Diehl & Cook 2015 [citation needed], each neuron in layer `Y` is connected with an inhibitory synapse to all other neurons in the layer. This creates a simple competitive mechanism: when a neuron fires, all other neurons are damped, allowing the "winning" neuron to continue firing and to update its weights in the direction of the current input pattern. This is similar to the idea of a *winner-take-all (WTA) circuit* [citation needed].

The result of this training is improved clarity of weight kernels. Since (typically) only one neuron is able to fire consistently, that neuron can significantly update the weights of the filter (using STDP) in the direction of the stimulus that is firing it; i.e., the input data.

One downside of this approach is that, since we are using a WTA-like mechanism on the "output" layer (`Y`), its activation is very sparse and perhaps not useful for further processing. Since each neuron is only detecting a small feature of the input data ()

2.3 Architectural modifications

2.3.1 Duplicate layer `Y` without recurrent connectivity