Notes on Recurrent LIF Layers

March 13, 2024

Introduction

We introduce algorithms for calculating the forward and backward pass of a fully connected layer of LIF neurons with recurrent connections.

1 Forward pass

To aid in the exposition, we define the following variables:

Variable name	Variable shape	Variable description
N_{in}	1	Number of input spikes
N_{post}	1	Number of postsynaptic neurons
N_{pre}	1	Number of presynaptic neurons
$\mathbf{t}$	N_{in}	Chronologically sorted input spike times
$\mathbf{s}$	N_{in}	Presynaptic neuron index of each input spike
$\mathbf{w}^f$	$N_{\text{post}} \times N_{\text{pre}}$	Feedforward synaptic weights
$\mathbf{w}^r$	$N_{\text{post}} \times N_{\text{post}}$	Recurrent synaptic weights
N_{out}	1	Number of output spikes
$\mathbf{T}$	N_{out}	Chronologically sorted output spike times
$\mathbf{S}$	N_{out}	Postsynaptic neuron index of each output spike

Let

$$\underset{s}{\textsc{NextOutputSpikeTime}}(\mathbf{t}, \mathbf{s}, \mathbf{T}, \mathbf{S}, \mathbf{w}^f, \mathbf{w}^r, t_0, u_0, t_{\text{start}})$$

denote the function that, given a vector $\mathbf{t}$ of input spike times, a vector $\mathbf{s}$ of corresponding presynaptic neuron ids, a vector $\mathbf{T}$ of output spike times,[1] a vector $\mathbf{S}$ of corresponding postsynaptic neuron ids, the feedforward weight matrix $\mathbf{w}^f$, the recurrent weight matrix $\mathbf{w}^r$, an initial time t_0 and corresponding initial membrane potential u_0, and an initial search time $t_{\text{start}} \geq t_0$, returns the first time $T \geq t_{\text{start}}$ such that the free membrane potential of the s^{th} postsynaptic neuron is above the threshold potential ϑ, or ∞ if no such time exists. The details of how to evaluate NextOutputSpikeTime are not discussed here; we explain only how to calculate the forward pass of the LIF layer in terms of it.

The forward pass calculates the final $\mathbf{T}$ and $\mathbf{S}$ vectors given $\mathbf{t}$, $\mathbf{s}$, $\mathbf{w}^f$, and $\mathbf{w}^r$. The idea is to hop from output spike to output spike, and when an output spike is found, it

[1] $\mathbf{T}$ is a vector of postsynaptic spike times *so far*, and analogously for $\mathbf{S}$. As more output spikes are calculated, $\mathbf{T}$ and $\mathbf{S}$ are appended to.

gets propagated to all the neurons in the layer. NextOutputSpikeTime is then called for each neuron to find the earliest next output spike. This continues until all neurons are silent or a user-defined maximum number of output spikes is reached. The approach is described in detail in Algorithm 1.

Algorithm 1 Calculates the output spikes of a fully connected layer of LIF neurons with recurrent connections.

procedure RecurrentLIFLayerForward($\mathbf{t}, \mathbf{s}, \mathbf{w}^f, \mathbf{w}^r$)
  $\mathbf{T} \leftarrow [\,]^\top$ ▷ Output spike times
  $\mathbf{S} \leftarrow [\,]^\top$ ▷ Output spike neuron ids
  $\mathbf{T}_{\text{prev}} \leftarrow [\underbrace{-\infty, -\infty, \ldots, -\infty}_{N_{\text{post}} \text{ many}}]^\top$ ▷ Most recent spike time of each neuron
  $t \leftarrow \min(\mathbf{t})$ ▷ Start the search at the earliest input spike
  while `len`($\mathbf{T}$) < maximum output spikes **do**
    $T \leftarrow \infty$ ▷ Next output spike time
    $S \leftarrow -1$ ▷ Next output spike neuron id
    for $s = 1, 2, \ldots, N_{\text{post}}$ **do** ▷ Query each neuron to find the next output spike
      $T_{\text{prev}} \leftarrow \mathbf{T}_{\text{prev}}[s]$
      if `isinf`(T_{prev}) **then**
        $T_{\text{neuron}} \leftarrow \underset{s}{\text{NextOutputSpikeTime}}(\mathbf{t}, \mathbf{s}, \mathbf{T}, \mathbf{S}, \mathbf{w}^f, \mathbf{w}^r, \min(\mathbf{t}), 0, t)$
      else
        $t_{\text{remaining}} \leftarrow \tau_{\text{ref}} - (t - T_{\text{prev}})$ ▷ Calculate remaining refractory time
        if $t_{\text{remaining}} < 0$ **then**
          $t_{\text{remaining}} \leftarrow 0$
        end if
        $T_{\text{neuron}} \leftarrow \underset{s}{\text{NextOutputSpikeTime}}(\mathbf{t}, \mathbf{s},$
          $\mathbf{T}, \mathbf{S},$
          $\mathbf{w}^f, \mathbf{w}^r,$
          $T_{\text{prev}} + \tau_{\text{ref}}, \varrho,$
          $t + t_{\text{remaining}})$
      end if
      if `isinf`(T) **or** $T_{\text{neuron}} < T$ **then** ▷ Find the neuron that fires first
        $T \leftarrow T_{\text{neuron}}$
        $S \leftarrow s$
      end if
    end for
    if `isinf`(T) **then** ▷ Terminate if no neuron fired
      break
    end if
    $\mathbf{T} \leftarrow \mathbf{T} + [T]^\top$ ▷ Save the output spike time
    $\mathbf{S} \leftarrow \mathbf{S} + [S]^\top$ ▷ Save the output spike neuron id
    $\mathbf{T}_{\text{prev}}[S] \leftarrow T$ ▷ Update the most recent spike time of the neuron that fired
    $t \leftarrow T$ ▷ Continue the search starting at the output spike
  end while
  return $\mathbf{T}, \mathbf{S}$
end procedure

2 Backward pass

Todo: write this section.

3 Experiments

3.1 Neuron impulse response

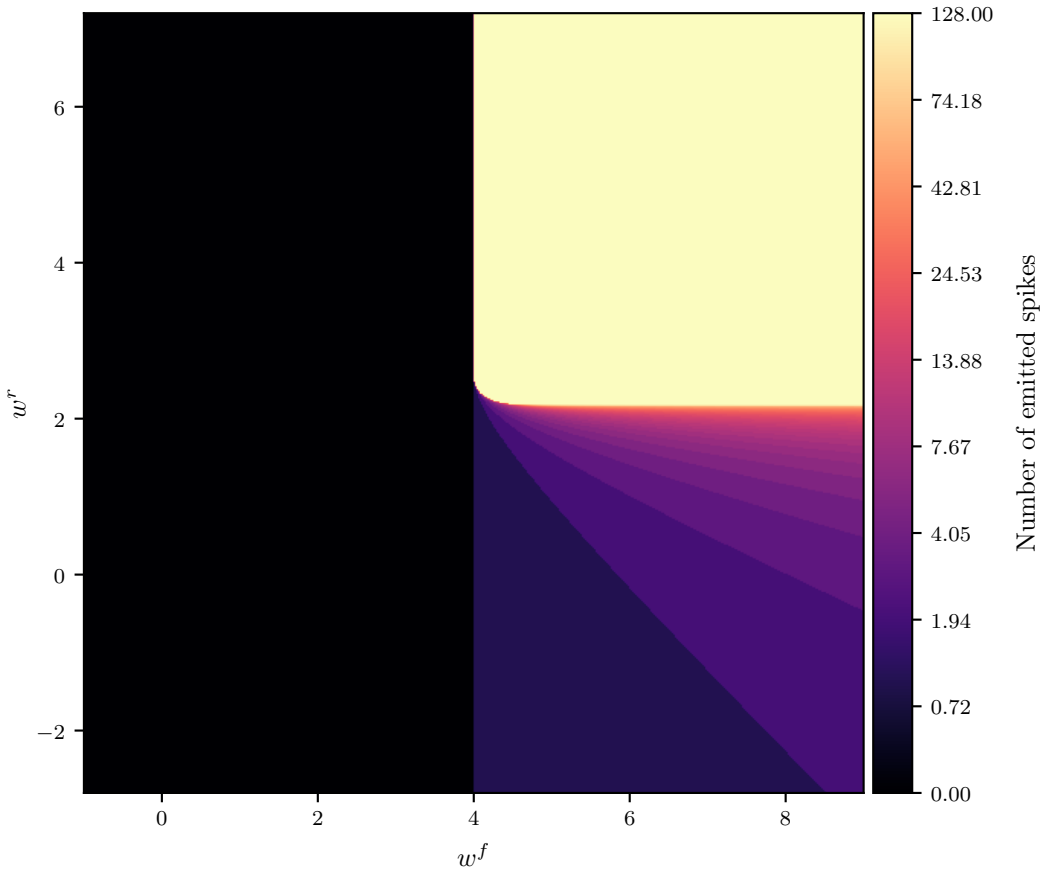

Figure 1: Number of emitted spikes of a LIF neuron in response to a single spike as a function of the feedforward weight w^r and the recurrent weight w^f.

3.2 Membrane dynamics in different regimes

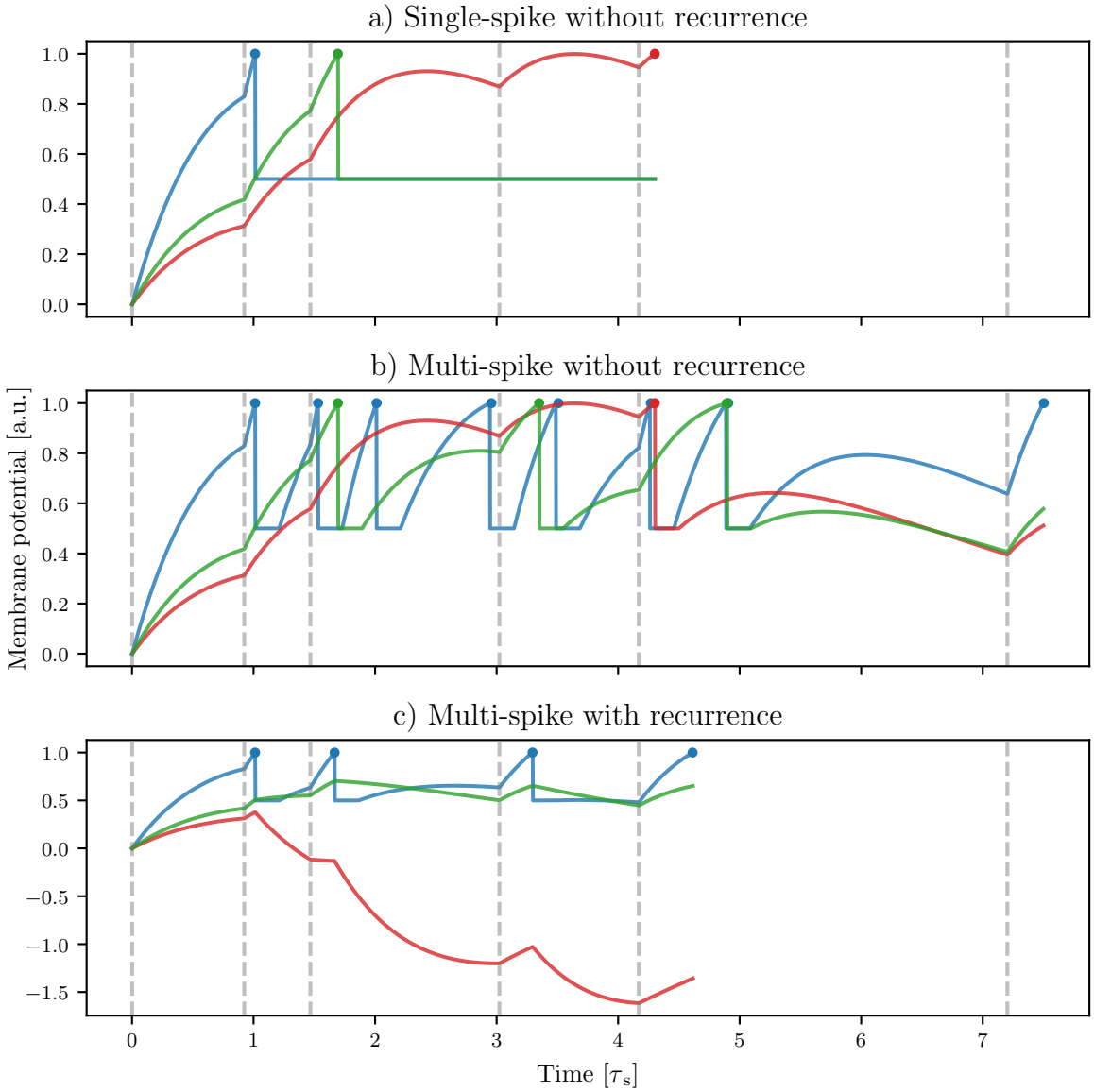

Figure 2: Membrane potentials of a 3-neuron SNN layer in response to the same stimulus for three different regimes. a) Neurons are allowed to fire only once, and the timing of the output spikes is not affected by previous output spikes. b) Neurons are allowed to fire arbitrarily many times, and the timing of the output spikes is not affected by previous output spikes. c) Neurons are allowed to fire arbitrarily many times, and the output spikes are fed back into the synapses. In this case, the recurrent weights are highly inhibitory, which causes the first-firing neuron (blue) to inhibit the other neurons (red and green) from firing.

3.3 Training on MNIST in different regimes

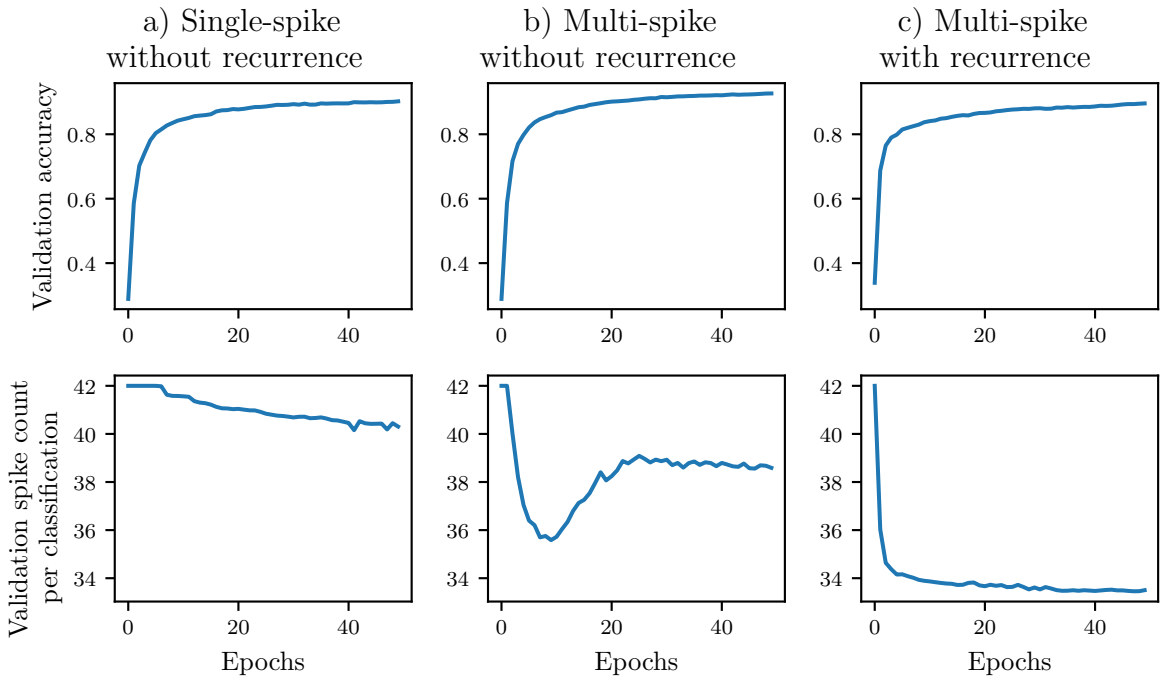

Figure 3: Training a 256-32-10 fully connected SNN on MNIST for 50 epochs.