Conversion of Single-Layer Spiking Neural Networks to Deep Neural Networks by Weight Matrix Factorization.

Bury Liam, Filali Ishak, Essien Daniel

2020-11-28

1 Introduction

In comparison to their artificial neural network (ANN) counterparts, spiking neural networks (SNNs) offer a more biologically plausible model of neural activities found in the brain. feature much richer temporal dynamics alongside a much simpler communication protocol. Two of the most important mechanisms of computation found in biological neural networks are neurons and the synaptic connections between them.

Neurons communicate through spikes, high voltage signals that typically persist for less than 1 ms in biological synapses. These high-energy but sparse and low-duration signals are believed to be one of the reasons why brains have such a high capacity for computation while consuming very little energy. Exploiting these qualities make SNNs a promising tool for low-energy computing in a large variety of problems including image classification, time-series prediction, robotic control and wearable technology. (Sorbaro, Liu, Bortone, & Sheik, 2020)

To present date, the training of deep spiking neural networks has proved a very difficult task due in part to a lack of portability of the tools used to train ANNs to SNN architectures. For example, gradient-descent based solutions are not longer effective due to the non-differentiability of spiketrains. In this paper, we present some of the ways in which matrix operations can be used to modify the structure of IF neural networks to produce deep SNNs that feature a higher degree of trainability due to an increased number of trainable parameters (synapses).

1.1 Synapse Model

Biological networks use synapses to transmit action potentials from a source neuron to one or more destination neurons. All transmit spikes of identical properties, but synapses modulate when and how each destination neuron is impacted by the spike.

The activity of a synapse is referred to as a spiketrain. Spiketrains can be defined as sequences of spike times

$$S = \{S_1, S_2, ..., S_n\} \tag{1}$$

or alternatively as a binary signal

$$V_i(t) = \sum_{k \in S} \delta(t - k) \tag{2}$$

where $V_i(t)$ is the voltage of the spiketrain from neuron i at time t. Spiketrains may only take one of two possible values, allowing for an extremely simplified communication system. (Drongelen, 2007) In current state-of-the-art neuromorphic computing hardware such as Intel's Loihi system, this communication is typically implemented with a simple packet router where the only information needed to transmit are the indices of target neurons.

Synapses compute a convolution over the spiketrain voltage, the output of which is integrated in the neuron. The alpha-function filter synapse was found by (Mainen & Sejnowski, 1995) to be a good basic model for synapses. It is defined by the impulse-response function given by

$$\frac{t}{\tau^2} e^{-t/\tau} \tag{3}$$

The postsynaptic potential caused by a spiketrain can then be interpreted as a continuous differentiable signal. (figure 2b)

$$V_i(t) = \sum_{k \in S} \frac{t-k}{\tau^2} e^{-(t-k)/\tau} \tag{4}$$

1.2 Integrate and Fire Neuron Model

The state of an IF neuron is its membrane potential. In biological neural networks, the membrane potential of a neuron is an electrical charge that is accumulated over time and released after passing a threshold condition, causing the neuron to generate a spike that is transmitted to as many as 100,000 other neurons. The binary nature of spikes is critical towards the efficiency of such elaborate networks.

Receiving a spike has the effect of increasing the membrane potential v of a neuron by an amount determined by weight and voltage of a synapse. In the IF neuron mode this the increase in membrane potential is linearly proportional to the voltage of the spike and can be expressed as

$$\frac{dv}{dt} = \sum_{i=0}^{n} I_i(t) = \sum_{i=0}^{n} w_i V_i(t) \tag{5}$$

where $I_i(t)$ is the postsynaptic current from synapse i at time t, $V_i(t)$ is the voltage of synapse i at time t, and w_i is the weight of synapse i. When crossing a voltage threshold V_T the membrane potential v is immediately set to some reset value V_R. This work will assume the commonly used values of $V_R = 0V$ and $V_T = 1V$.

For a constant input current $I > 0$, the amount of time required to raise the membrane potential v from V_R to V_T is referred to as the inter-spike interval $t_j(I)$ of a neuron j given a constant input current I. (Gerstner, Kistler, Naud, & Paninski, 2014) For IF neurons this can be expressed as

$$t_j(I) = \{t | \int_0^t dv = V^T - V^R\} = \frac{(V_T - V_R)}{I} \tag{6}$$

A similarly important statistic of a neuron is the firing rate f_j of a neuron j. Firing rate is inversely proportional the inter-spike interval and proportional to input current.

$$f_j(I) \approx \frac{I}{V_T - V_R} = I = \sum_{i=0}^{n} w_i V_i(t) \tag{7}$$

Therefore the firing rate of an IF neuron is proportional to its input current given a constant input. We can assume that this is also true for inputs that change slowly. (Gerstner et al.)

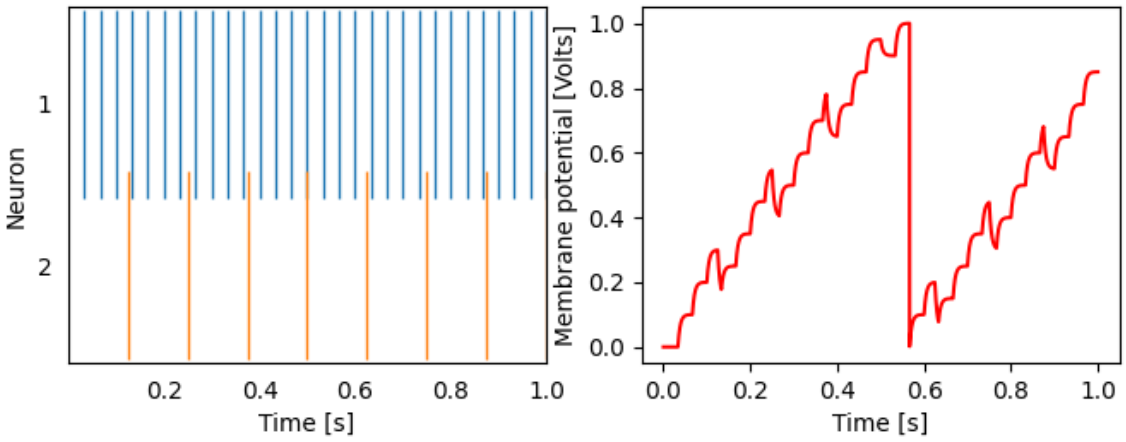

Figure 1: (left) Two spiketrains with frequencies 30 Hz and 8 Hz. (right) The membrane potential of an IF neuron after integrating the high and low frequency spiketrains with weight matrix $[1, -1.5]$. Upon crossing the threshold V_T the membrane potential of the neuron is reset to V_r.

2 Weight Matrices

Suppose we have two fully connected populations of neurons I and O containing N and M neurons respectively. The full set of synaptic connections from I to O can be described by a weight matrix $W \in \boldsymbol{R}^{N \times M}$, where W_{ij} is the weight of the synapse from I_i to O_j. Then we have,

$$\frac{dv_{O_j}}{dt} = \sum_{i=0}^{N} W_{ij} V_{I_i}(t) = W V_i \tag{8}$$

We let $v_O = \left[v_{O_1}, v_{O_2}, ..., v_{O_{N-1}}\right]$, then we have

$$\frac{dv_O}{dt} = W V_I \tag{9}$$

The firing rate of a neuron is proportional to the dot product of an input and connection strength. This reveals that IF neurons have the capability of computing dot products, and therefore matrix multiplication. The engineering applications of such computation are practically limitless.

The maximum firing rate of a neuron is an important property in the application of spiking neural networks. In rate-based models such as this, if a neuron fires more frequently then it carries a higher information density, yet consumes more energy. (Drongelen, 2007) The maximum firing rate f_j of neuron j given weight matrix W and scalar $k > 0$ can be estimated as

$$\max f_j = \frac{f_j(I_{max})}{V_T - V_R} = ||kWV_i(t)||_\infty = k||W||_\infty ||V_i(t)||_\infty \tag{10}$$

In the case that $0 \leq W_{ij} \leq 1$ and $0 \leq V_i(t) \leq 1$, the above formula reduces to

$$\max f_j = k \tag{11}$$

In general, we find that the firing rate of an IF neuron is proportional to the average of summed input currents. Now let $f_O = \left[f_{O_1}(WV_1), f_{O_2}(WV_2), ..., f_{O_{N-1}}(WV_{N-1})\right]$. Then we can say

$$f_O \approx V_O \approx WV_I \tag{12}$$

3 Conversion of Continuous Inputs to Spiketrains

Let x be an N-dimensional input signal defined as

$$x_n(t) = \frac{1 + sin(4t - n)}{2}, 0 \leq n \leq N, 0 \leq x_n(t) \leq 1 \tag{13}$$

We construct a population X of N IF neurons with parameters as described previously to encode the continuous values of x_n with spiketrains. Let $k = 100$ be the desired maximum firing rate of all neurons in X. When using a permutation matrix as a weight matrix, a one-to-one mapping from a neuron of x to X is constructed. We use the weight matrix kI_n to inject the current of each x_n into the corresponding neuron X_n with gain k. Then the firing rates of neurons in X approach

$$f_{X_j} \approx \sum_{i=0}^{N} kW_{ij}x_i(t) = kx_n(t)$$
$$f_X \approx kIx$$

The network was simulated with the python package Nengo over a period of two seconds (figure 2). The decoded output value of X was then obtained by normalizing the summed postsynaptic potentials of X. The total maximum firing rate of the population X is $k * N$, therefore the decoded of output of X should be normalized by scaling by a factor of $1/kN$. Observe how as total firing rates decrease, so does the accuracy of signal reconstruction. This is one shortcoming of rate-based encoding methods. (Sorbaro et al.)

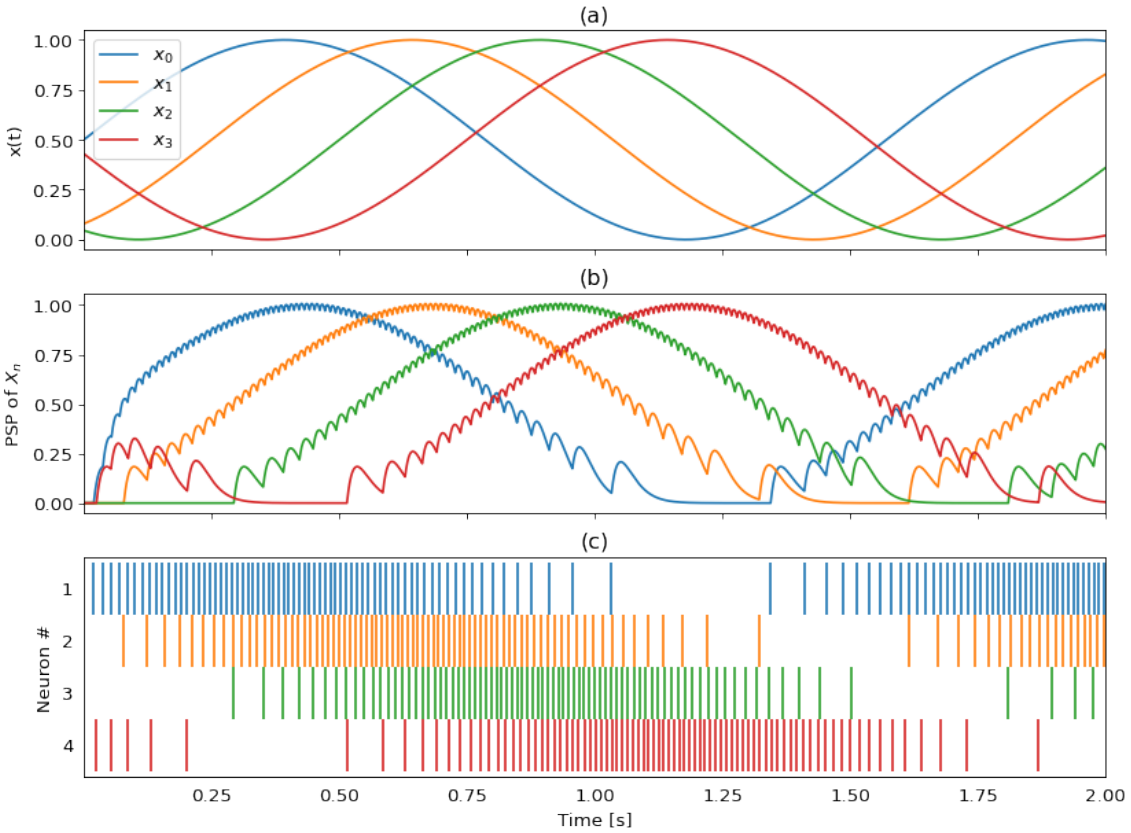

Figure 2: (a) The continuous input signal x. (b) The combined postsynaptic potential of the neurons in X approximating x after being passed through a 20ms alpha filter. (c) The spiketrains of neurons in X.

4 Multilayer SNN

Our objective is to apply some transformation to the continuous input signal with spiking IF neurons. To do so we generate a random weight matrix W_A to map the 4 input dimensions to a population A of N neurons, in addition to a random weight matrix W_B mapping A to a single output neuron Y_1.

$$W_A \in \mathbb{R}^{4\times N}, 0 \leq \mathrm{W_{Aij}} \leq 1$$

$$W_B \in \mathbb{R}^{N\times 1}, 0 \leq \mathrm{W_{Bij}} \leq 1$$

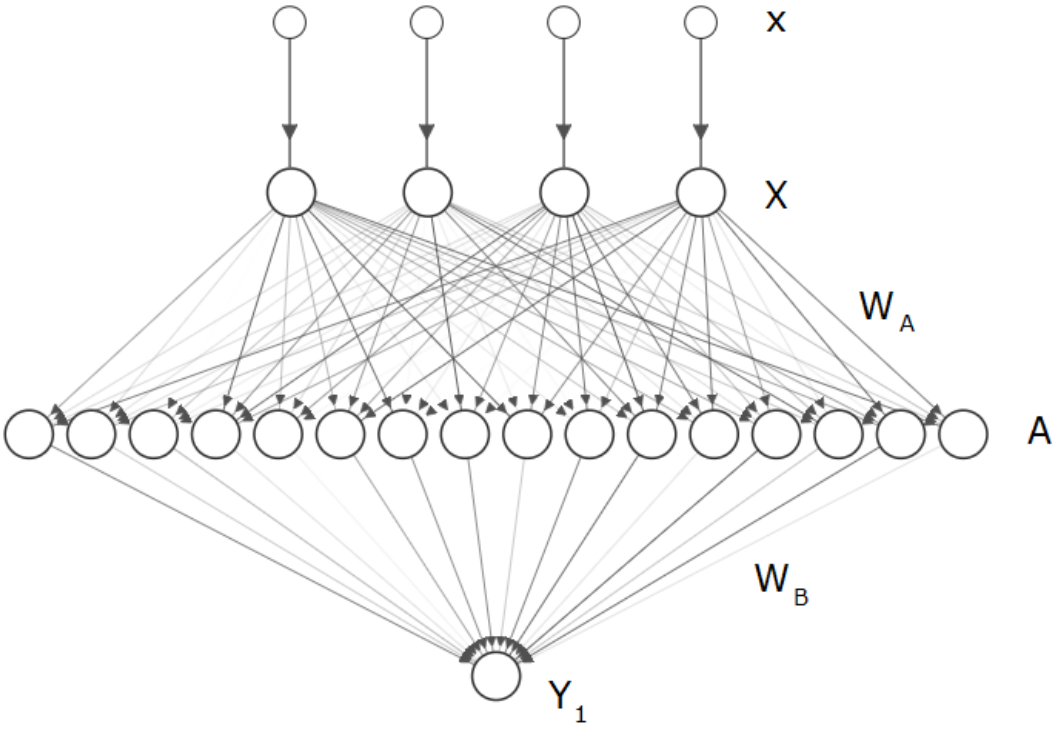

Figure 3: Diagram of the constructed network for N=16.

The output of this network can be described by the firing rate of its output layer. From equations 8 and 12, we see that

$$f_{Y_1} \approx W_B W_A X \approx W_B W_A I k x$$

5 Conversion to Deep SNN

Deep neural networks provide many advantages over their simplistic single-layer counterparts. Each additional layer of a neural network is capable of computing an additional layer of abstraction upon the state of the previous layer [**bibid**].

To convert the neural network to a deep neural network, we first decompose the weight matrix W_A into the product of several matrices representing successive layers in a feed-forward network. The PLU decomposition of a matrix is a simple example that allows us to express W_A as the product of several weight matrices W_P, W_L, and W_U, where W_P is a permutation matrix, W_L is a lower triangular matrix, and W_U is an upper triangular matrix. Note that a permutation matrices, when used as weight matrix, represents a redundant one-to-one remapping of dimensions. Therefore, in this network we premultiply W_P and W_L to make the weight matrix W_{PL}.

The dynamics of output Y_2 can be formulated as

$$f_{Y_2} \approx W_B(W_{PL}W_U)X \approx \frac{W_B W_{PL} W_U I k x}{k}$$
$$\approx W_B(W_A)X \approx f_{Y_2}$$

Thus, the original neural network and the newly constructed deep network are mathematically equivalent. To verify this we simulated both pathways in Nengo for a duration of 2 seconds and a simulation time-step of 0.1 ms (Figure 4).

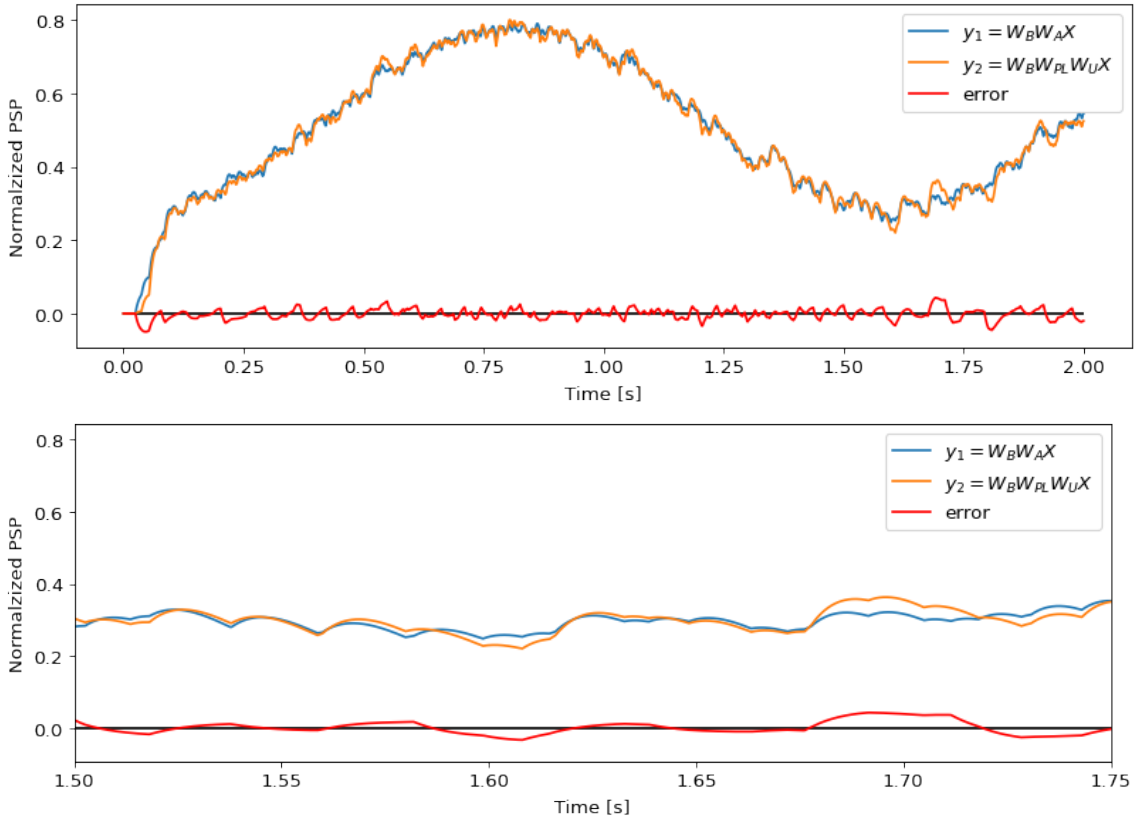

Figure 4: PSP of the two different transformation pathways with a 20ms alpha filter.

6 Results

	Single-Layer SNN	Deep SNN
# Neurons	37	69
# Synapses	164	612
Error	+0.00 %	-0.147 %

Figure 5. Parameters of the original and deep SNNs.

By adding a single additional fully connected layer, we have increased the number of trainable parameters in the network by 239% without introducing a significant source of error. After this step we may now apply learning algorithms designed for spiking neural networks, further reducing error between the input signal and a desired output.

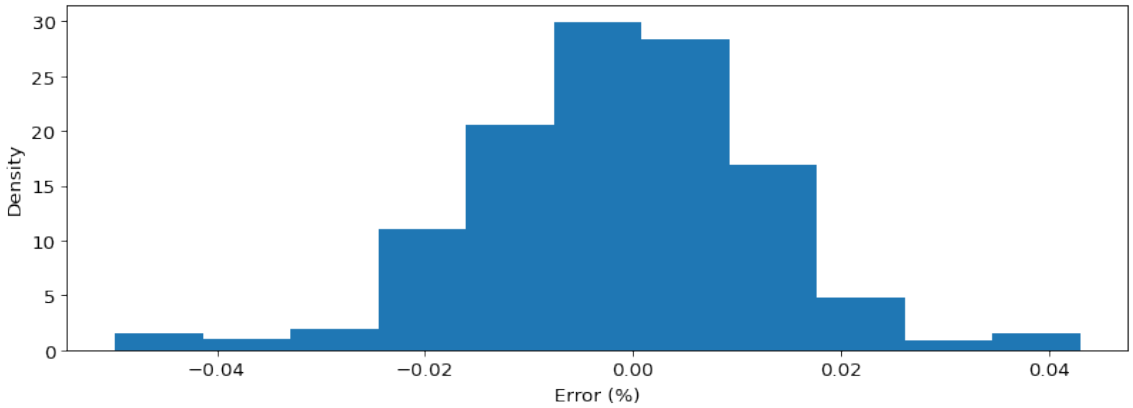

Figure 5: Distribution of error between the two output signals.

Percentile	1%	10%	50%	90%	99%
Error	-4.4%	-1.8%	-0.057%	1.42%	3.7%

Figure 6. Error parameters of the deep SNN.

7 Conclusion

The source of error in our experiment is most likely that we have assumed that the firing rate of a neuron is proportional to the dot product previous layer activity and the synaptic weight matrix. In our digital simulation of synapses, it is possible for a spike to cause a neuron's membrane potential to far exceed the firing threshold. This excess current will not contribute to the generation of a consecutive spike, causing the neurons firing rate to be slightly less than our linear equations would suggest.

One possible way to avoid this problem would be to integrate the current of a spike slowly over a short duration rather than instantaneously. This could be achieved through the use of more advanced synaptic models, such as the alpha synapse model. This would increase the accuracy of our model at the cost of increased latency in the form of delayed spikes.

8 Bibliography

[1] Sorbaro, M., Liu, Q., Bortone, M., & Sheik, S. (2020). Optimizing the Energy Consumption of Spiking Neural Networks for Neuromorphic Applications. Frontiers in Neuroscience, 14. doi:10.3389/fnins.2020.00662.

[2] Drongelen, Wim van. (2007) Signal Processing for Neuroscientists. Academic Press, 219-243. doi:0.1016/B978-012370867-0/50014-0.

[3] Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). Variability of spike trains and neural codes. Neuronal Dynamics, 168-201. doi:10.1017/cbo9781107447615.009

[4] Mainen, Z.F. and Sejnowski, T.J. (1995). Reliability of spike timing in neocortical neurons. Science (New York, NY), 268(5216):1503-6.