Internship Report

Author: Peter Lake
Supervisor: Julian Göltz

November 15, 2023

Background

This project demonstrates an alternative interpretation for the output of the spiking neural network (SNN) classification model introduced by Göltz et al. in [1]. Unlike the original formulation, in which the network's class prediction and loss function are based on the differences between the output spike times of the label neurons, the new approach is based on absolute spike times and is trained using a mean squared error (MSE) loss function. This is shown to produce a slightly better classification accuracy than the original method when trained on the Yin-Yang benchmark classification task [2].

The model and the framework used to train it were both introduced in [1]. The framework, called *Fast and deep*, is a technique for training networks of leaky integrate-and-fire (LIF) neurons. The LIF neuron model roughly captures the dynamics of a biological neuron by modeling the neuron as a simple resistor-capacitor circuit [3] [4]. The neuron's membrane is represented as a capacitor with capacitance C_m connected in series with a resistor with resistance $1/g_l$, which models the membrane's leakiness, and a battery with voltage E_l, which determines the membrane's rest potential. An additional current term, $I(t)$, corresponds to the synaptic current caused by spikes received from presynaptic neurons. The equation governing the voltage $u(t)$ across the capacitor is

$$C_\mathrm{m}\dot{u}(t) = g_l\left[E_l - u(t)\right] + I(t).$$

Since E_l can be set arbitrarily,[1] it's convenient to set $E_l = 0$ so that

$$C_\mathrm{m}\dot{u}(t) = -g_l u(t) + I(t). \tag{1}$$

Fast and deep uses current-based synapses with an exponential synaptic interaction kernel. The synaptic current $I(t)$ is given by

$$I(t) = \sum_i w_i \sum_{t_i} \theta(t - t_i) \exp\left(-\frac{t - t_i}{\tau_\mathrm{s}}\right)$$

where w_i are the presynaptic weights, t_i are the input spike times, τ_s is the synaptic time constant, and $\theta(t)$ is the Heaviside step function. The first sum runs over all presynaptic neurons, and the second sum runs over all spikes for each presynaptic neuron. To simulate action potentials, while integrating Equation (1), as soon as the membrane potential $u(t)$

[1] Changing E_l is equivalent to adding a constant bias to $u(t)$.

exceeds the threshold potential ϑ, an output spike is emitted, and the membrane potential is instantaneously clamped to a reset potential ϱ for a time period τ_{ref}. The solution to Equation (1) is reported in [1] and repeated in Equation (4).

The Fast and deep framework provides methods for training networks of LIF neurons using exact spike times and conventional backpropagation. This is achieved by deriving a closed form solution for the output spike time of a LIF neuron as a function of the input spike times and synaptic weights. All the derivative formulas needed for backpropagation can then be readily derived.

To introduce our new interpretation for the output of the SNN classification model from [1], we denote by t_n the first output spike time of the n^{th} neuron of the output layer where $n \in \{1, 2, \ldots, C\}$ corresponds to each of the C classes. In the original formulation, the model's class prediction is taken to be the index of the first-spiking label neuron. This behavior is trained by minimizing the following cross-entropy loss function:

$$L_{\text{XE}} = \log \left[\sum_{n=1}^{C} \exp \left(-\frac{t_n - t_{n^*}}{\xi \tau_{\text{s}}} \right) \right] \tag{2}$$

where n^* is the correct class, $\xi > 0$ is a scaling parameter, and τ_{s} is the synaptic time constant of the LIF neuron. Minimizing L_{XE} encourages the label neuron associated with the correct class to spike earlier than the remaining label neurons.

In the new formulation, the model's class prediction is taken to be the index of the label neuron that spikes closest in time to t_{correct}, which is a fixed hyperparameter. This behavior is trained by minimizing the following MSE loss function:

$$L_{\text{MSE}} = (t_{n^*} - t_{\text{correct}})^2 + \sum_{n \neq n^*} (t_n - t_{\text{incorrect}})^2 \tag{3}$$

where $t_{\text{correct}}, t_{\text{incorrect}} > 0$ are hyperparameters. Minimizing L_{MSE} encourages the correct label neuron to spike close in time to t_{correct} and the remaining label neurons to spike close to $t_{\text{incorrect}}$. Note that while L_{XE} is shift-invariant in the sense that it depends only on the differences between the label neuron spike times t_n, L_{MSE} does not have this property and depends explicitly on the individual spike times.

Experiments

In this section, we tune the hyperparameters t_{correct} and $t_{\text{incorrect}}$ of the MSE loss function (Equation (3)) on the Yin-Yang benchmark classification task [2], then compare the performance of the model when trained using the MSE loss versus the original cross-entropy loss (Equation (2)).

MSE loss hyperparameter tuning

As a first experiment, we tune the hyperparameters t_{correct} and $t_{\text{incorrect}}$ of the MSE loss function when training the model on the Yin-Yang classification task with a hidden layer size of 120 neurons, which is the value assumed everywhere in this report unless otherwise indicated. The model is trained multiple times with random weight initializations for each cell in a grid of $(t_{\text{correct}}, t_{\text{incorrect}})$ values. As shown in Figure 1, the classification error is lowest around $(t_{\text{correct}}, t_{\text{incorrect}}) = (1.7\tau_{\text{s}}, \tau_{\text{s}})$, which is the tuned value used

for all subsequent experiments. In general, the classification error appears lowest when t_{correct} and $t_{\text{incorrect}}$ are both close to $1.7\tau_{\text{s}}$, which is the average output spike time of the untrained, randomly initialized network, as can be seen in the left panel of Figure 3 b. Figure 1 shows the classification error to be approximately invariant under swapping of t_{correct} and $t_{\text{incorrect}}$ with a slight preference for $t_{\text{correct}} > t_{\text{incorrect}}$. This indicates that the MSE loss function appears to work slightly better when the correct label neuron is trained to fire after the incorrect label neurons. This is reversed from the original paper [1] in which the correct label neuron is trained to fire first.

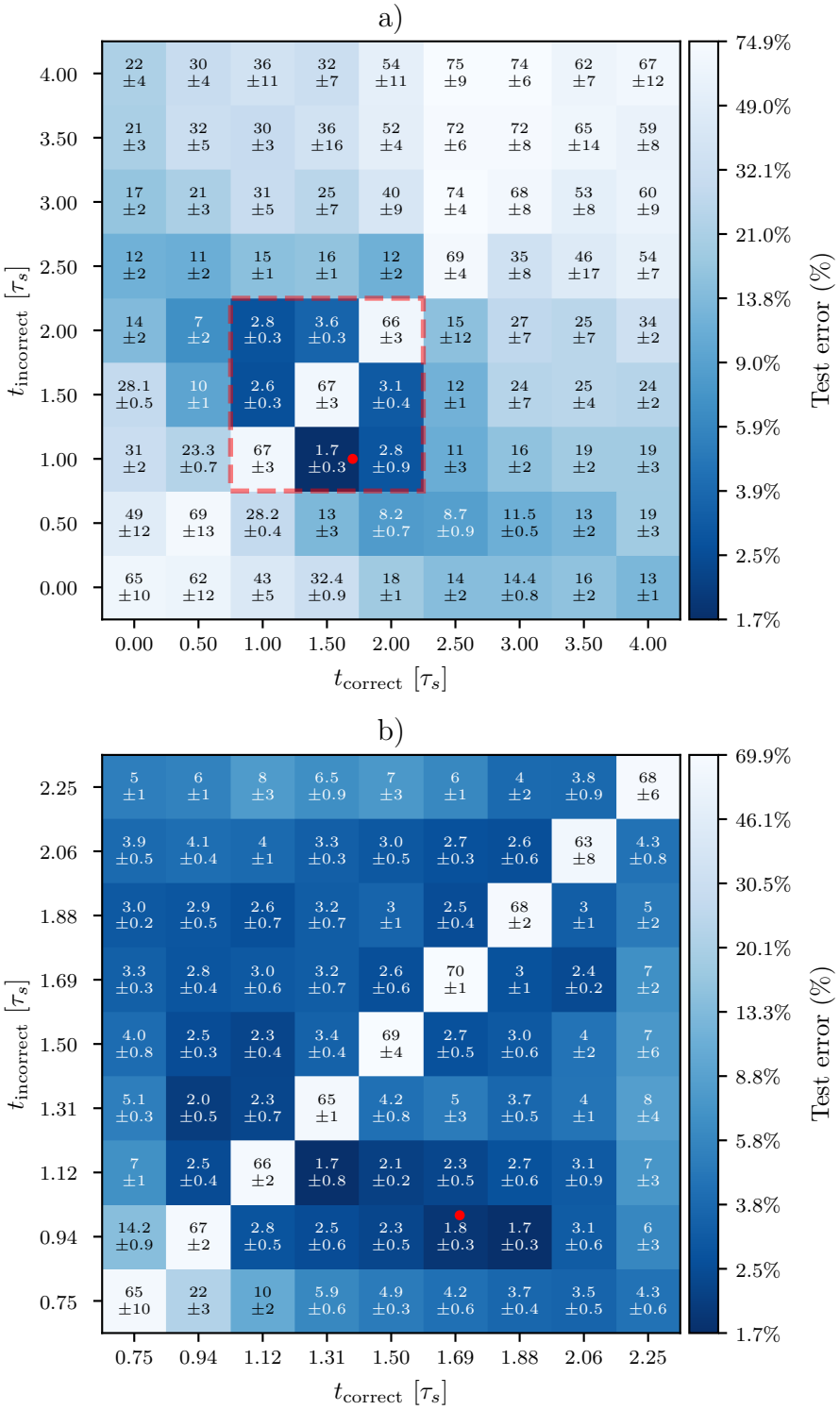
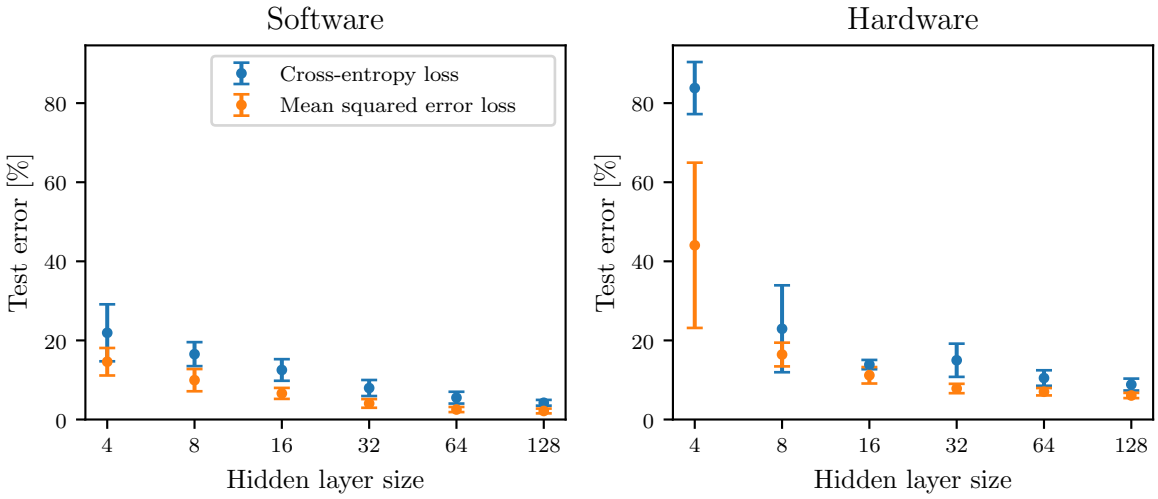


Figure 1: Mean and standard deviation of the final test error on the Ying-Yang dataset when using the MSE loss with various values of the hyperparameters $t_{\text{correct}}, t_{\text{incorrect}}$. The red dot (•) marks the coordinate $(t_{\text{correct}}, t_{\text{incorrect}}) = (1.7\tau_s, \tau_s)$, which is used in the rest of this report. For each cell, the results from 5 training runs with random weight initializations are shown. **(a)** Test errors for $(t_{\text{correct}}, t_{\text{incorrect}}) \in [0, 4\tau_s]^2$. **(b)** Test errors for the smaller region of hyperparameter space highlighted in (a).

Comparing cross-entropy loss versus MSE loss

Next, we compare the accuracy of the tuned MSE loss function against the original cross-entropy loss function when training on the Yin-Yang task. The comparison is performed for several hidden layer sizes ranging from 4 to 128 neurons. As shown in Figure 2, the MSE loss outperforms the cross-entropy loss both in software simulations and hardware emulations for hidden layer sizes of 32, 64, and 128 neurons.

(a) Full comparison



(b) Zoomed in

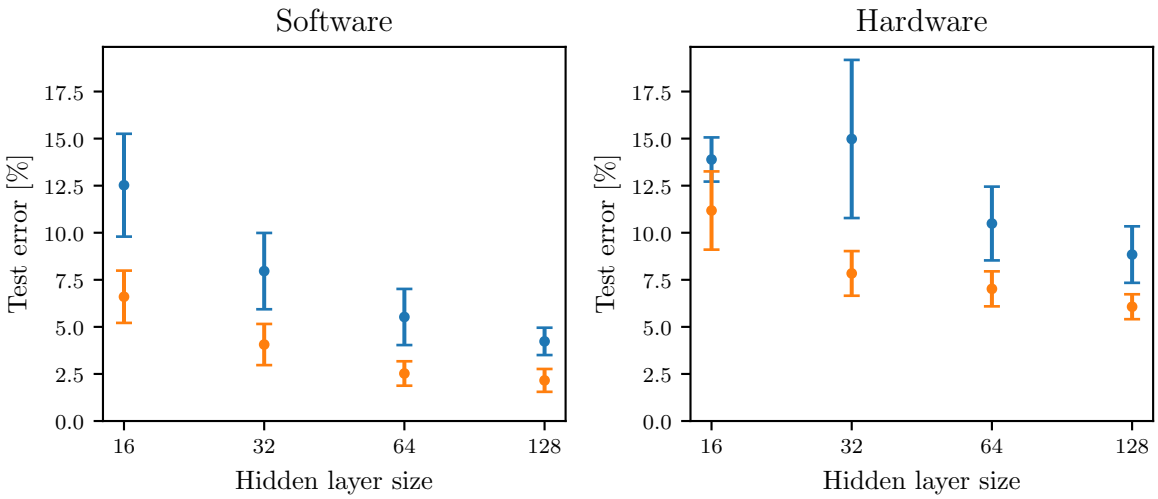


Figure 2: Mean and standard deviation of the final test error on the Yin-Yang task versus the hidden layer size of the network. The two loss functions, cross-entropy and MSE, are compared when training using software simulation versus hardware emulation. For each loss function and hidden layer size, the results from 40 (software) or 10 (hardware) training runs with random weight initializations are shown. The top and bottom rows show the same data with different x and y-axis scales.

Histograms of label neuron spike times

Both the cross-entropy and MSE loss functions encourage a time separation between the spike times of the correct and incorrect label neurons. To better understand the differing behaviors that these two loss functions produce, we inspect the distributions of the label neuron spike times of the network when trained under each loss function, again using the Yin-Yang task.

In Figure 3, histograms of the spike times of the correct and incorrect label neurons over the training dataset are shown separately, highlighting the model's ability to separate the correct class from the incorrect classes under each loss function. Both loss functions appear to encourage a strong separation, but the MSE loss achieves a noticeably clearer separation. Note that for the cross-entropy loss, the correct label neuron is trained to fire first as per Equation (2), but for the MSE loss, the correct label neuron is trained to fire last, which was found to be optimal during the hyperparameter tuning above.

Another quantity whose distribution over the training dataset is of interest is the difference in spike time between the correct label neuron and the closest-to-correct incorrect label neuron, which is defined as[2]

$$\Delta t_{\mathrm{XE}} := t_{n_*} - \min_{n \neq n^*} \{t_n\}$$

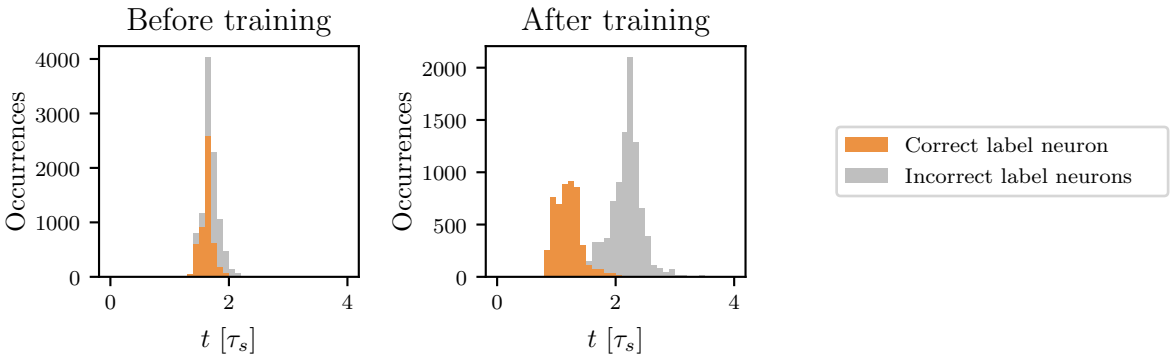
for the cross-entropy loss and

$$\Delta t_{\mathrm{MSE}} := t_{n^*} - \max_{n \neq n^*} \{t_n\}$$

for the MSE loss. This distribution gives a more explicit idea of how well the model separates the correct from incorrect classes across the examples. Again, as shown in Figure 4, both loss functions produce a good separation, but the MSE loss produces a clearer separation.

[2]The difference in definitions is due to the fact that for the cross-entropy loss, the correct label neuron is trained to fire first, while for the MSE loss, the correct label neuron is trained to fire last.

(a) Training using cross-entropy loss



(b) Training using MSE loss

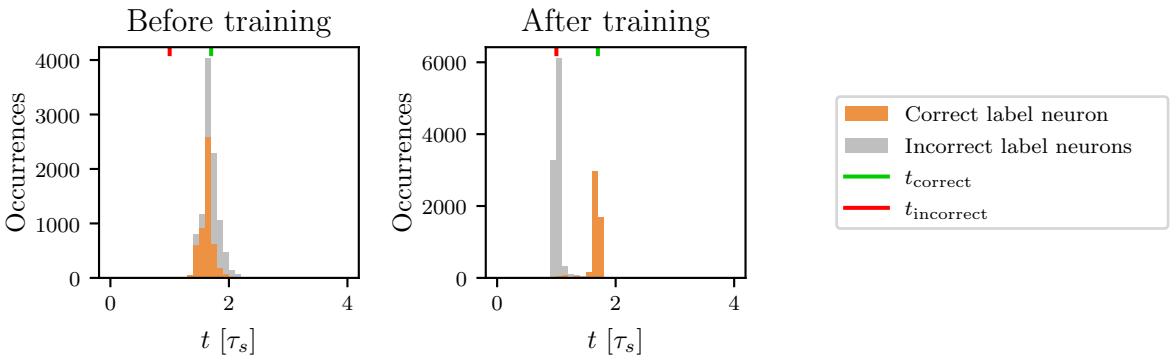
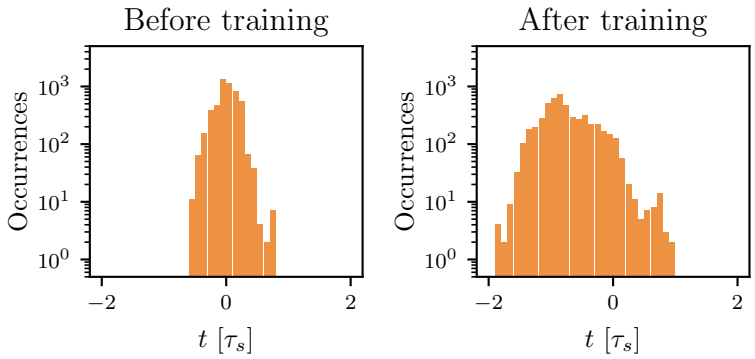


Figure 3: Histograms of the spike times produced by the label neurons over the entire Yin-Yang training set. The spike times from the correct (orange) and incorrect (grey) label neurons are shown separately. The experiment is repeated for the cross-entropy (top row) and MSE (bottom row) loss functions both before (left column) and after (right column) training. The model weights for both networks are initialized using the same seed.

(a) Training using cross-entropy loss



(b) Training using MSE loss

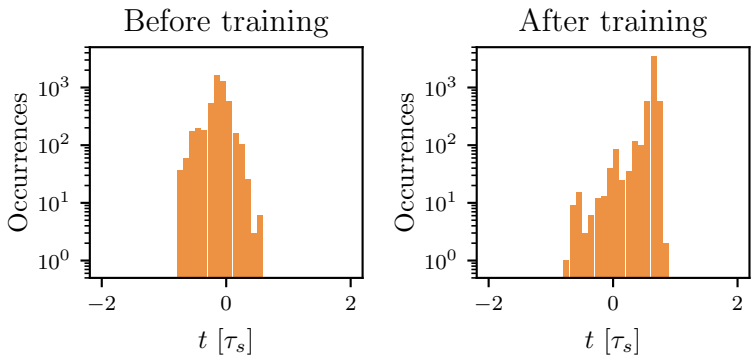


Figure 4: Histograms of the difference in output spike time between the correct label neuron and the closest-to-correct incorrect label neuron over the entire Yin-Yang training set. The experiment is repeated for the cross-entropy (top row) and MSE (bottom row) loss functions both before (left column) and after (right column) training. The model weights for both networks are initialized using the same seed.

Conclusion

In this report, an alternative output interpretation and corresponding loss function for the Fast and deep classification model [1] based on the mean squared error was tuned and compared against the approach used in the original paper. The new approach was shown to produce a small but clear improvement in accuracy on the Yin-Yang classification task both in software simulations and hardware emulations for several hidden layer sizes.

Appendix

Deriving spike time derivatives using implicit differentiation

Here we derive formulas for the derivatives of the output spike time T of a LIF neuron with respect to the input spike times t_i and synaptic weights w_i.

The solution to the LIF equation (Equation (1)) with initial data $u(0) = 0$ is

$$u(t) = \frac{1}{C_\mathrm{m}} \frac{\tau_\mathrm{m}\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \sum_i w_i \sum_{t_i} \kappa(t - t_i) \tag{4}$$

where

$$\kappa(t) = \theta(t) \left[\exp\left(-\frac{t}{\tau_\mathrm{m}}\right) - \exp\left(-\frac{t}{\tau_\mathrm{s}}\right) \right]$$

is the PSP kernel. Assuming each presynaptic synapse receives exactly one spike, this can be written more succinctly as

$$u(t) = \frac{1}{C_\mathrm{m}} \frac{\tau_\mathrm{m}\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \sum_{\text{spikes } t_j} w_j \kappa(t - t_j). \tag{5}$$

Let T be the first output spike time and ϑ be the threshold potential. Then $u(T) = \vartheta$, so by Equation (5), we have

$$\vartheta = u(T) = \frac{1}{C_\mathrm{m}} \frac{\tau_\mathrm{m}\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \sum_{\text{spikes } t_j} w_j \kappa(T - t_j). \tag{6}$$

Differentiating both sides of Equation (6) with respect to t_i gives

$$\begin{aligned}
0 &= \frac{1}{C_\mathrm{m}} \frac{\tau_\mathrm{m}\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \sum_{\text{spikes } t_j} \frac{\partial}{\partial t_i} \left[w_j \kappa(T - t_j) \right] \\
\Rightarrow \quad 0 &= \sum_{\text{spikes } t_j} \frac{\partial}{\partial t_i} \left[w_j \kappa(T - t_j) \right] \\
\Rightarrow \quad 0 &= \sum_{\text{spikes } t_j} \left[w_j \kappa'(T - t_j) \left(\frac{\partial T}{\partial t_i} - \delta_{ij} \right) \right] \\
\Rightarrow \quad \frac{\partial T}{\partial t_i} &= \frac{w_i \kappa'(T - t_i)}{\sum_{\text{spikes } t_j} w_j \kappa'(T - t_j)}.
\end{aligned} \tag{7}$$

Similarly, differentiating both sides of Equation (6) with respect to w_i gives

$$\begin{aligned}
0 &= \frac{1}{C_\mathrm{m}} \frac{\tau_\mathrm{m}\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \sum_{\text{spikes } t_j} \frac{\partial}{\partial w_i} \left[w_j \kappa(T - t_j) \right] \\
\Rightarrow \quad 0 &= \sum_{\text{spikes } t_j} \frac{\partial}{\partial w_i} \left[w_j \kappa(T - t_j) \right] \\
\Rightarrow \quad 0 &= \sum_{\text{spikes } t_j} \left[\delta_{ij} \kappa(T - t_j) + w_j \kappa'(T - t_j) \frac{\partial T}{\partial w_i} \right] \\
\Rightarrow \quad \frac{\partial T}{\partial w_i} &= \frac{-\kappa(T - t_i)}{\sum_{\text{spikes } t_j} w_j \kappa'(T - t_j)}.
\end{aligned} \tag{8}$$

Equations (7) and (8) are formulas for the derivatives of the output spike time T with respect to the input spike times t_i and synaptic weights w_i. They are particularly convenient because they are independent of the method used to calculate T, and because they explicitly incorporate T. This has been argued in [1] to make training more robust in scenarios where the exact neuron parameters are unknown and the real output spike time may therefore differ from the one calculated under ideal assumptions.

References

[1] J. Göltz, L. Kriener, A. Baumbach, S. Billaudelle, O. Breitwieser, B. Cramer, D. Dold, A. F. Kungl, W. Senn, J. Schemmel, K. Meier, and M. A. Petrovici. "Fast and energy-efficient neuromorphic deep learning with first-spike times". In: *Nature Machine Intelligence* 3.9 (Sept. 2021), pp. 823–835. ISSN: 2522-5839. DOI: 10.1038/s42256-021-00388-x. URL: https://doi.org/10.1038/s42256-021-00388-x.

[2] L. Kriener, J. Göltz, and M. A. Petrovici. "The Yin-Yang Dataset". In: *Proceedings of the 2022 Annual Neuro-Inspired Computational Elements Conference*. NICE '22. Virtual Event, USA: Association for Computing Machinery, 2022, pp. 107–111. ISBN: 9781450395595. DOI: 10.1145/3517343.3517380. URL: https://doi.org/10.1145/3517343.3517380.

[3] M. A. Petrovici. "Form vs. function: theory and models for neuronal substrates". PhD thesis. University of Heidelberg, Germany, 2015. URL: http://www.ub.uni-heidelberg.de/archiv/21402.

[4] P. Dayan and L. F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2005. ISBN: 0262541858.