CSE8803-BMI Lab1 Report

Tristan Peat

September 2024

1 Introduction

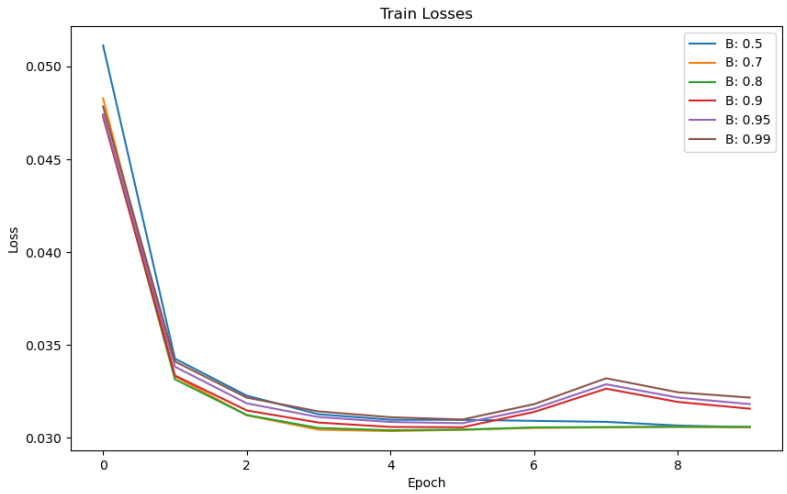
Spiking Neural Networks (SNNs) are a class of neural networks that more closely resemble biological processes by using discrete events, known as spikes, to control when information is propagated. In this study, we use the MNIST digit classification dataset to investigate how key hyperparameters affect SNN performance. Specifically, we examine the number of timesteps (T), which determines the temporal resolution; the surrogate gradient scale (z), which influences the approximation of gradients for backpropagation; the firing threshold (V_{th}), which dictates when a neuron fires; and the leakage factor (β), which controls the decay rate of the neuron's membrane potential. Each model was trained for 10 epochs, with identical random seeds to ensure reproducibility across experiments.

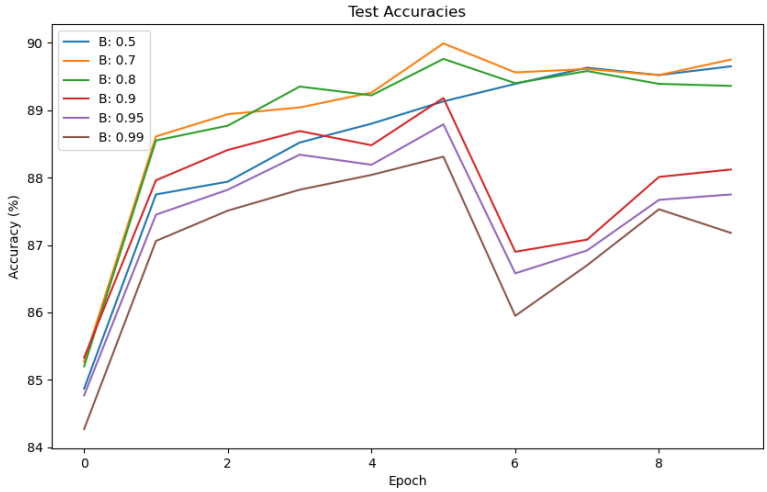
2 Comparison of ANN and SNN

ANN are trained using gradient descent such as SGD, Adam using algorithms, libraries, and hardware that are highly optimized. However, all of the aforementioned rely on differentiable functions. The spiking mechanism is inherently non-differentiable. Therefore training relies on a surrogate gradient method to smooth the non-differentiable spiking mechanism so that traditional backpropagation techniques can be used for optimization. Additionally, the incorporation of temporal dynamics in SNN increases computational complexity. The following hyperparameter experiments show the benefits of longer timesteps which introduces a tradeoff between performance and training time. Another complication introduced with the temporal dynamics is the idea of credit assignment; attributing model error to timing of spikes occurring over multiple timestamps in addition to the connection between neurons. The challenge is because spikes take time to through a network so it complicates the process of linking an error to specific spikes.

3 Tuning β

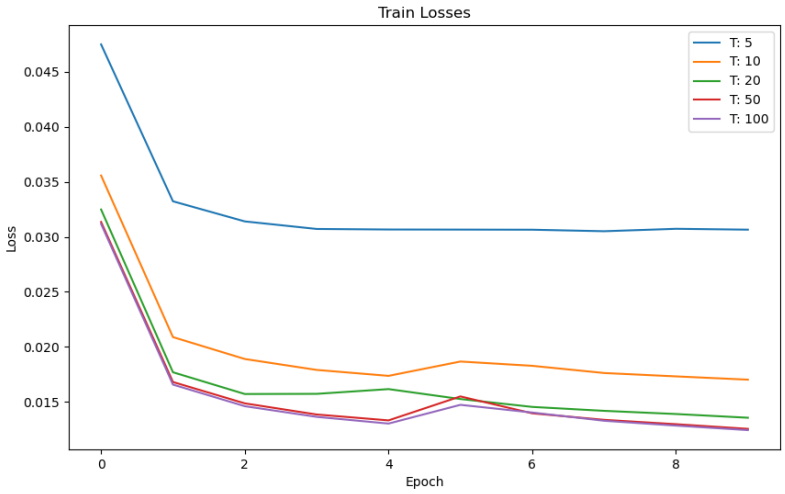
The leakage factor β controls the decay rate of membrane potential for neurons of SNN, effectively determining how long a neuron retains its past input. We tested β values of $[0.5, 0.7, 0.8, 0.9, 0.95, 0.99]$. As shown in the following figures, higher β values led to lower training losses and higher test accuracies in early epochs. This indicates that a slower decay rate allows neurons to build up valuable information quicker. However, as training continues, lower values of β catch up to the performance of higher β values. Setting β too close to 1 can cause the network to retain information for too long and have diminishing returns.

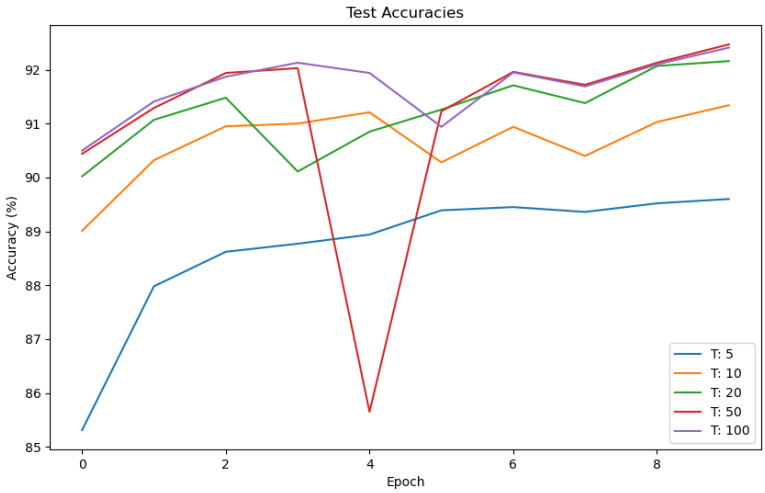




4 Tuning T

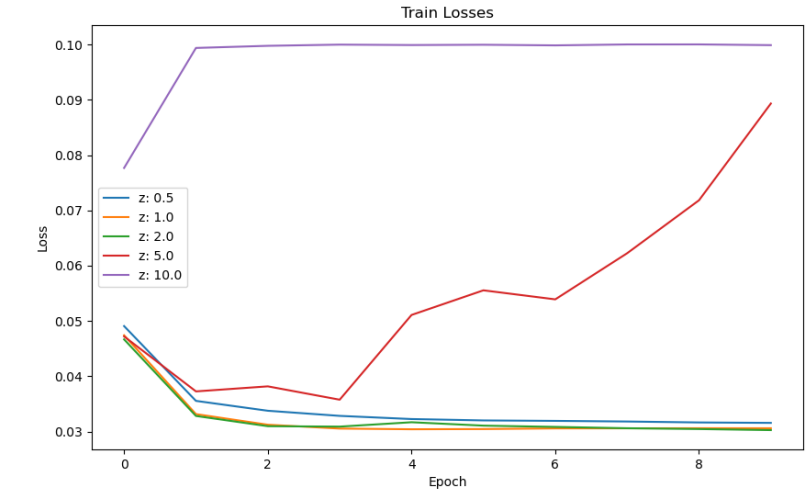
The number of timesteps T in a Spiking Neural Network (SNN) determines the temporal resolution and the duration over which the network processes input data. We tested T values of $[5, 10, 20, 50, 100]$. As illustrated in the following figures, increasing T led to a significant reduction in training losses and an improvement in test accuracies. This trend indicates that longer temporal windows allow the network to integrate information over more time steps, enhancing its ability to learn complex temporal patterns present in the input data. However, while higher T values improve performance, they also increase computational complexity and training time. Therefore, selecting an appropriate T involves balancing the benefits of improved accuracy against the additional computational overhead.

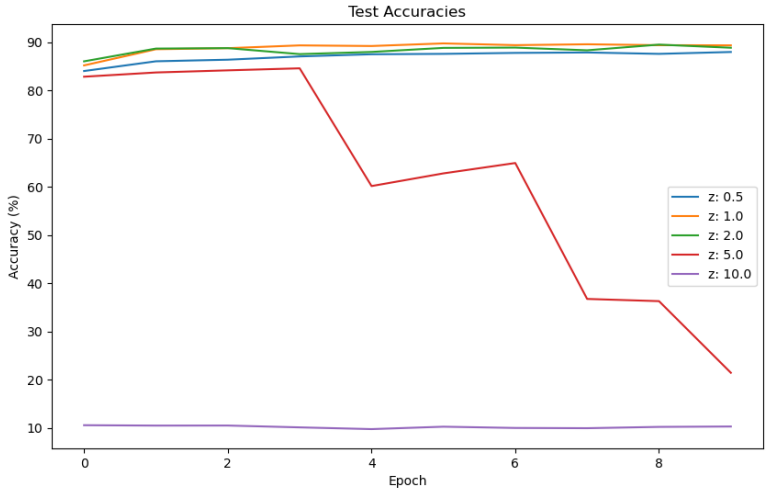




5 Tuning z

The surrogate gradient scale factor z influences the sharpness of the gradient used during backpropagation in SNN. A larger z results in steeper gradient approximations for spiking function, meaning it's closer to the exact value which should be zero everywhere exact at the threshold. Therefore, the magnitudes of gradients are much larger when the membrane potential approaches the threshold, which results in drastic updates during training. We suspect the larger values of z, as observed in the figures below, are causing instability in training and constantly overshooting the minima. We observe the best performance for moderate values of z, with the caveat that they may come with slower convergence speeds.

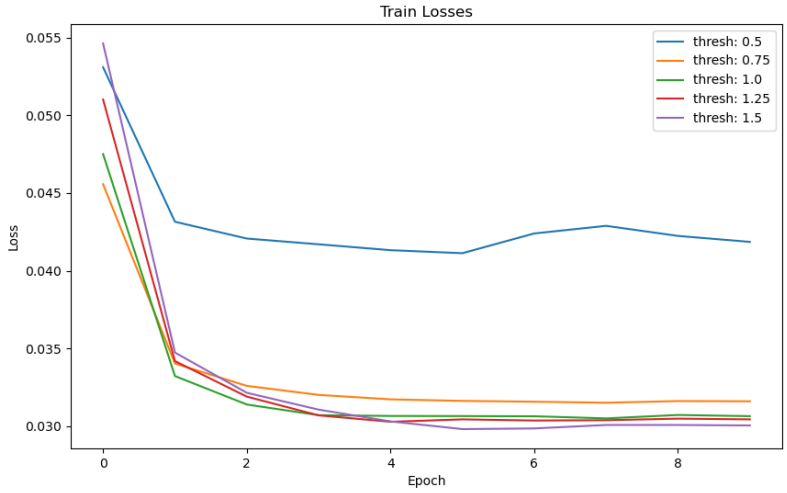


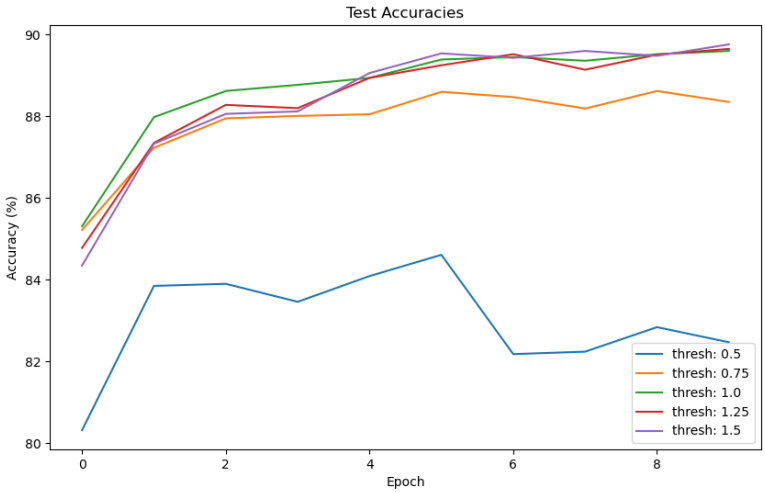


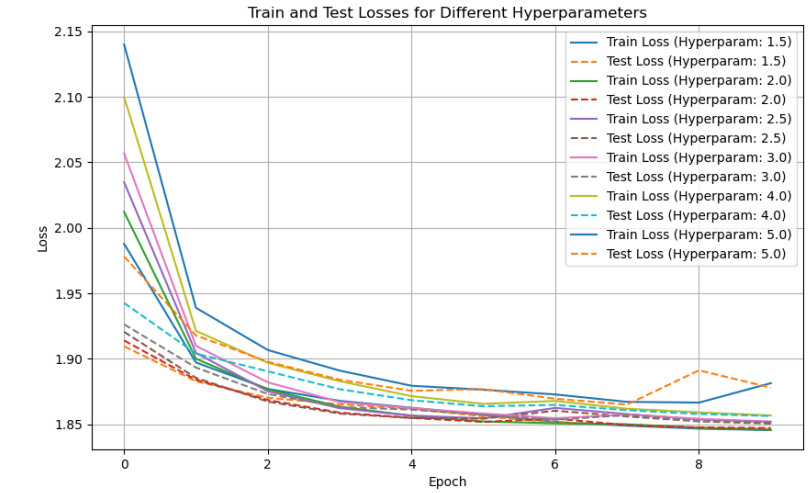
6 Tuning neuron threshold V_{th}

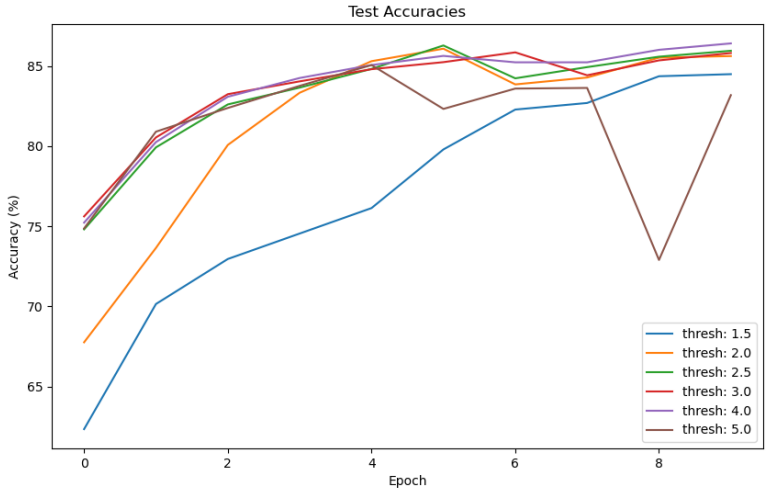
The firing threshold V_{th} in a SNN determines the membrane potential level at which a neuron emits a spike. Adjusting V_{th} affects the neuron's excitability; a lower threshold makes neurons more likely to fire, while a higher threshold requires stronger input to trigger a spike. We tested the neuron threshold with values $[0.5, 0.75, 1.0, 1.25, 1.5]$. As depicted in the figures below, moderate threshold values around 1.0 achieved a balance between sufficient network activity and training stability.

While threshold value of 0.5 had notably worse performance, performance seemed to be linearly proportional to threshold value. We re-ran the experiment with a wider spread of values and noticed that accuracy improves with a larger threshold, but both the train and test losses are higher. This might suggest that the reduced spiking in the neural network causes less information to be transmitted, impacting learning but also increasing generalization.









7 Comparing input encoding schemes

NOTE: A small bug was identified in the `test_snn` method, where the loss from the criterion needed to be multiplied by the batch size. While the training took too long to re-run for every hyperparameter experiment, the test loss may not be fully accurate.

Using the baseline hyperparameters outlined in Table 1, we evaluated three different input encoding strategies: `gen_train_data_img`, `gen_spike_data_static`,

Table 1: Baseline Model Hyperparameters

Hyperparameter	Value
Number of timesteps (T)	50
Surrogate gradient scale (z)	2.0
Threshold (θ)	1.5
Leakage factor (β)	0.7

and `gen_spike_data_bernoulli`.

The `gen_spike_data_bernoulli` function achieved the highest test accuracy among the encoding schemes. This method leverages stochastic Bernoulli sampling to generate spike trains, introducing temporal variability in the input data. The randomness in spike generation allows the Spiking Neural Network (SNN) to exploit temporal dynamics effectively, leading to improved learning and generalization on the MNIST classification task.

The `gen_train_data_img` function, which repeatedly presents the same static image across all timesteps, resulted in the lowest test accuracy. This approach does not utilize the temporal processing capabilities of SNNs, as there is no temporal variation in the input. Consequently, the network's ability to capture dynamic patterns over time is limited, affecting its overall performance.

Overall, temporal diversity seeems to improve SNN performance which explains the success of the `gen_spike_data_static` function.

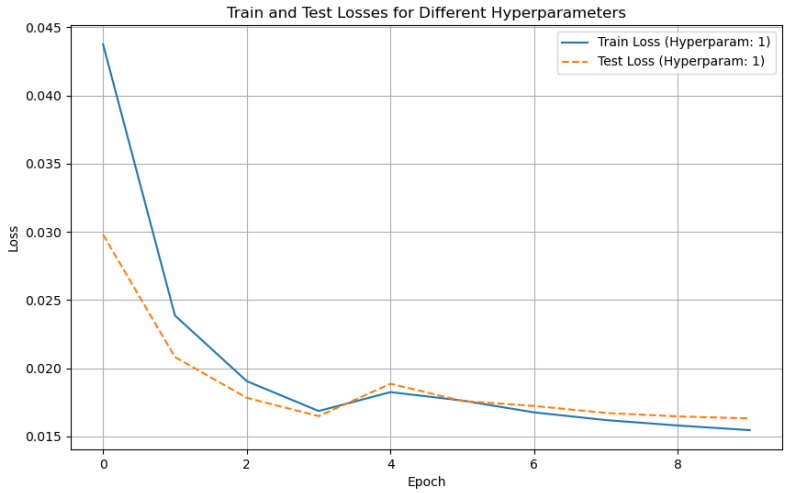
8 Comparing MSE to CE

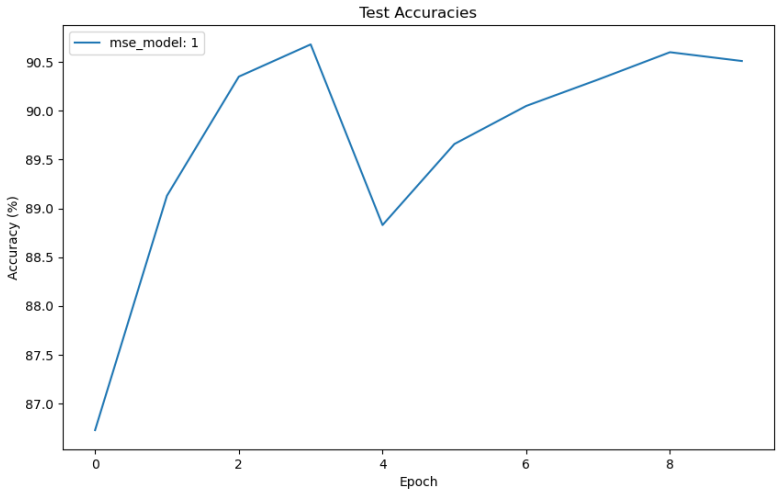
Using the baseline model from Table 1, we compare the use of MSE against CrossEntropyLoss. MSE is typically used for regression. To implement the change we removed the division by number of timesteps in the SNN output layer to ensure CE receives unnormalized scores because it applies softmax within the module. CE also expects indices of class $0 - 9$ for MNIST.

As seen in the following figures, CE performed 3 points higher than MSE. Additionally, the convergence was faster for CE than MSE with a much steeper loss curve for train in the first 2 epochs. The improvement in training dynamic offered by CE might be because it was designed specifically to measure the difference in predicted class probabilities and their associated class labels. Given MNIST is a classification task, perhaps CE offers more informative gradients for updating the model weights, which is especially important to the gradient approximate in the SNN.

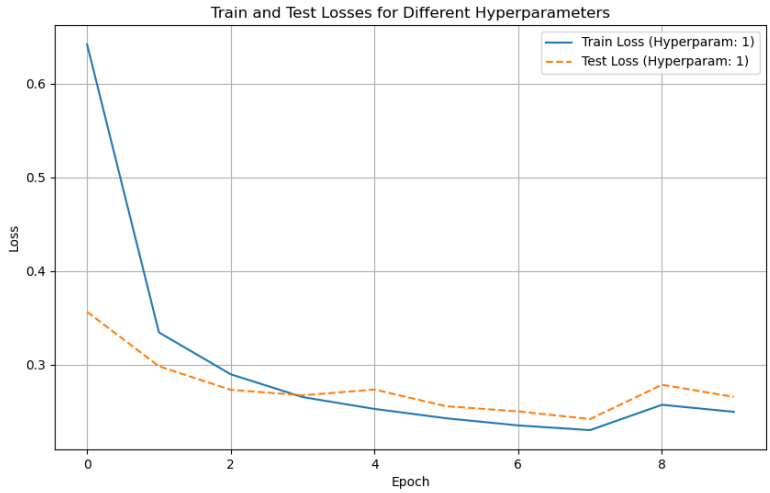
8.1 Baseline MSE Model

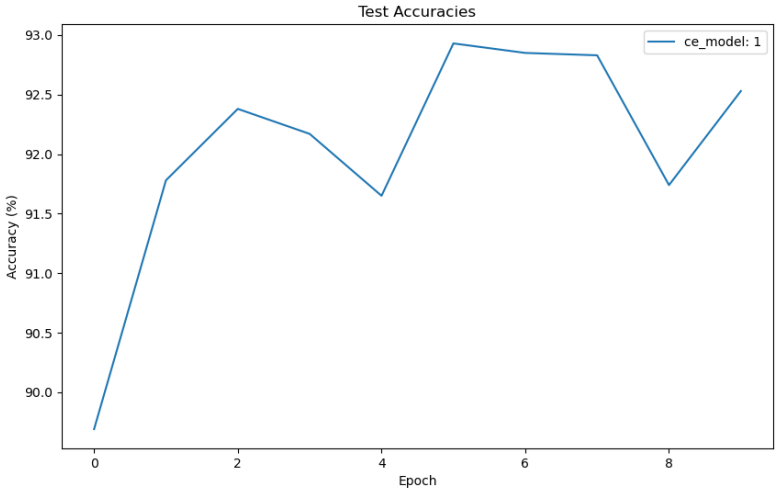
NOTE: the title is slightly misleading, these are just the train and test metrics for the baseline model optimized for MSE criterion with parameters form Table 1.





8.2 CE Model





9 FLOPs Calculation

One of the key advantages of SNNs is their potential for reduced computational complexity and energy efficiency, particularly due to sparse spikes. Therefore, we will compare the FLOPs between ANN and SNN.

9.1 Calculating ANN FLOPs

For Layer 1, with an input size of $M = 784$ and an output size of $N = 10$:

$$\text{FLOPs}_{\text{Layer 1}} = (2 \times 784 - 1) \times 10 + 2 \times 10 = 15,690 \text{ FLOPs}$$

For Layer 2, with an input size of $M = 10$ and output size of $N = 10$:

$$\text{FLOPs}_{\text{Layer 2}} = (2 \times 10 - 1) \times 10 + 2 \times 10 = 210 \text{ FLOPs}$$

Summing the FLOPs from both layers, we have:

$$\text{Total FLOPs}_{\text{ANN}} = 15,690 + 210 = 15,900 \text{ FLOPs}$$

9.2 Calculating SNN FLOPs

The FLOPs for the spiking neural network (SNN) depend on the average firing rates, which vary across layers. After computing the firing rates F_1 and F_2, the FLOPs per layer can be calculated. While we assume F_1, F_2 to be given we could could take ratio of the total spikes in a time window T divided by the length of the window and average this ratio over all neurons in the layer.

With input size $M = 784$, output size $N = 10$, and firing rate $F_0 = 1$ for the input layer, the FLOPs per timestep for Layer 1 are:

$$\text{FLOPs}_{\text{Layer 1}} = (1568 - 1) \times 10 + 20 = 15,690 \text{ FLOPs per timestep}$$

For Layer 2, with input size $M = 10$, output size $N = 10$, and firing rate arbitrarily chosen to be $F_1 = 0.25$, the FLOPs are:

$$\text{FLOPs}_{\text{Layer 2}} = (20 \times 0.25 - 1) \times 10 + 20 = 60 \text{ FLOPs per timestep}$$

We get the total FLOPs per timestep by summing over each layer:

$$\text{Total FLOPs}_{\text{SNN per timestep}} = 15,690 + 60 = 15,750 \text{ FLOPs per timestep}$$

For $T = 5$ timesteps, the total number of FLOPs is:

$$\text{Total FLOPs}_{\text{SNN}} = 5 \times 15,750 = 78,750 \text{ FLOPs}$$

9.3 Discussion

In summary, the total number of FLOPs for the artificial neural network (ANN) is $15,900$ FLOPs, while the total number of FLOPs for the SNN over $T = 5$

timesteps is $78,750$ FLOPs. This calculation highlights the importance of T in the computational complexity of SNN, as the total FLOP count is multiplied by number of timesteps. Additionally, the SNN seems like it has higher total FLOPs due to the temporal dimension introduced by multiple timesteps. However, when considering the sparsity induced by low firing rates, the actual number of operations can be significantly reduced, especially in deeper networks or larger layers.

However, in our MNIST simulation, we didn't use deeper or larger networks than a ANN so the difference was unnoticeable between the two, until a sizable scaling of the timestep resolution T. For any $T > 10$, we observed a large increase in the runtime required to train the model.