Spiking Neural Network Project

Marcus Williams

June 2022

1 Introduction

This project set out to simulate a spiking neural network using the Brian 2 python library and train it on the MNIST Numbers and MNIST Fashion datasets. We used Spike-timing-dependent plasticity (STDP) and explored the effect of changing the network and STDP parameters, as well as the differences between training on the Fashion and Numbers dataset.

2 Theory

2.1 Spiking neural networks

A spiking neural network is a network of artificial spiking neurons. SNNs more closely resemble biological neural networks than do other ANNs. Unlike most other ANNs, SNNs implement the concept of time, neurons do not constantly send information to each other but only when their activation threshold has been reached. When a neuron fires it sends a signal to other neurons which increase or decrease their probability of firing. SNNs can be used to perform many of the tasks typical ANNs do, although currently they often do so with worse performance. SNNs can theoretically consume very little power, only $\approx 10^{-5}$ [2] as much of a comparable standard network, if run on hardware made specifically to model neurons.

2.2 Model of a neuron

We model the neuron using the leaky-integrate-and-fire model. In this model a neuron has an internal counter, a number of input synapses with corresponding synaptic weights and output synapses with their own weights. Every time an input spike is received the counter is increased corresponding to the weight of the synapses the spike came from. When a certain activation threshold is reached the neuron will send a pulse along all its outputs and the counter is reset. The counter will also gradually decrease over time if no input pulses are received, and is therefore "leaky". This means that the neuron will only fire if enough pulses are received in a short enough time period. It is also possible to have inhibitory synapses i.e. negative synaptic weights, these will decrease the internal counter of other neurons rather than increase it.

2.3 Spike-timing-dependent plasticity

It is not possible to use classic back-propagation in a SNN. An alternative method of updating weights is spike-timing-dependent plasticity or STDP. In STDP the weight change of a synapse is dependent on the timing between the spikes of a pre- and postsynaptic neuron. If the presynaptic neuron spikes a short amount of time prior to the postsynaptic neuron that means that the pre-spike in some sense "predicted" the post-spike so it would typically be good to reinforce that behaviour by increasing the synaptic weight, while if the pre-spike comes after the post-spike it was clearly not instrumental to the post-spike happening and should therefore typically be depressed. Typically the weight change depends exponentially on the time between the pre- and post-spike though many possibilities are observed in different biological systems. This means that STDP is an unsupervised method which SNNs can be trained with. While STDP is often exponential in time it is important that it is linear or at least symmetric in conductance[1] since even a small non-linearity can cause catastrophic performance loss if the STDP function is asymmetric in conductance. This can be a problem when implementing SSNs with physical memristors which are often not linear.

2.4 Lateral inhibition

In a system which implements lateral inhibition, after a neuron fires, other neighbouring neurons in the same layer are inhibited from firing. There are two main types of lateral inhibition, "winner takes all" and "soft inhibition".

With "winner takes all" lateral inhibition, all other neurons in the same layer are completely inhibited. "Soft lateral inhibition" instead typically only reduces the membrane potential of neighbouring nodes.

Lateral inhibition is meant to help the neurons specialize at different things. It also reduces overtraining as the inhibition makes it less likely that the neurons collaborate in specific ways to get the right result.

2.5 Adaptive threshold

With adaptive threshold the neurons activation threshold goes up if the neuron spikes a lot and down if it spikes little. This leads to a more stable activity level in the network.

When manufacturing physical devices it is common to have small variations in properties such as conductance. This can lead to the neuron activation threshold effectively varying between neurons which could cause under- or over-activity in places in the network. Adaptive threshold should mitigate the issues this causes by equalising activity across the network.

2.6 MNIST Numbers dataset

The MNIST numbers dataset contains 70 000 28x28 pixel greyscale images of handwritten numbers 0-9. Each pixel has a single integer between 0 and 255 associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. The dataset is divided into 60

000 training images and 10 000 test images. An accuracy of about 99.8% has been achieved by [3] using a convolutional neural network.

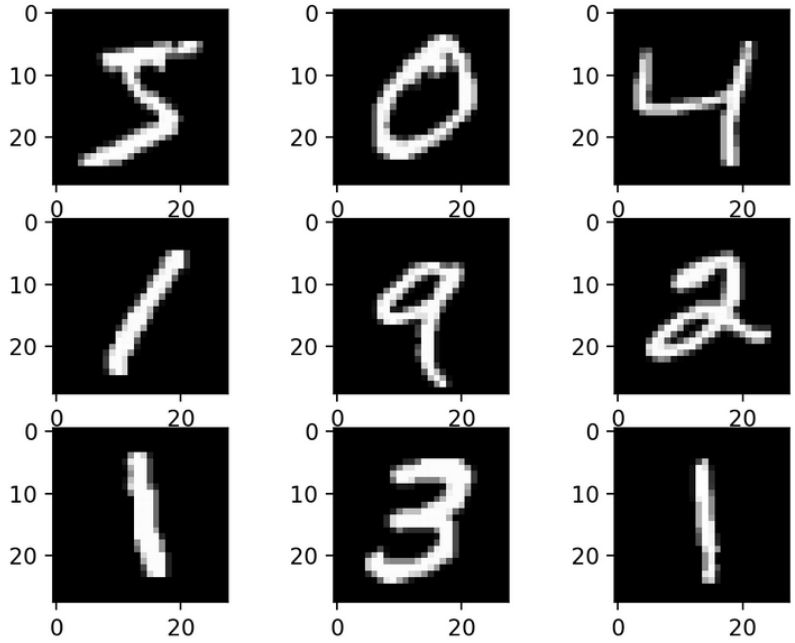

Figure 1: The first 9 images in the MNIST numbers dataset

2.7 MNIST Fashion dataset

The MNIST Fashion dataset contains 70 000 28x28 pixel greyscale training images of 10 different categories of articles of clothing based on zalandos (an online clothing retailer) catalogue, see table 1.

Table 1: Labels with corresponding article of clothing in the MNIST Fashion dataset

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

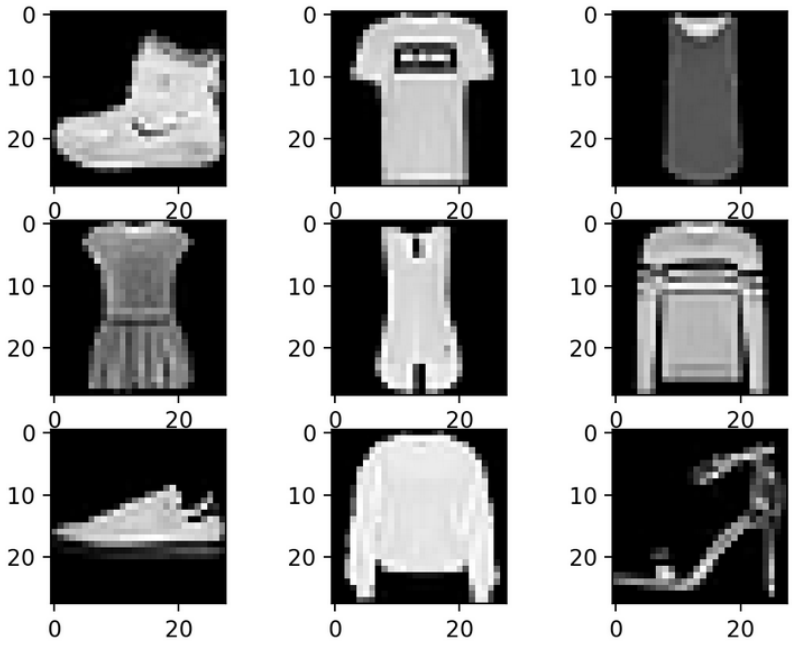

Figure 2: The first 9 images in the MNIST fashion dataset

Again the training set contains 60 000 images and the test set 10 000. This dataset is considered to be harder than the Numbers dataset with one of the best models only achieving an accuracy of 96.9% using a CNN [4].

3 Model

We implement an SNN using one 28x28 input layer and a varied amount, N_{out}, of output neurons (typically 196). All of the input neurons are connected to all output neurons via a synapse. The input neurons spikes were made to spike based on the input images using rate encoding, their spiking frequency set between 0 and 50 Hz relative to the value (colour) of the input pixel.

3.1 Training

To train the network we showed each of the 60 000 images to the input neurons for 250ms separated by a 150ms resting time in order for the previous images potentials to die down. STDP was used to update the synaptic weights continuously.

3.2 Classification

Because STDP is an unsupervised method we do not immediately know which neurons correspond to which label. In order to classify which number each output neuron corresponds, we showed the network 1 000 random images from the training set and recorded how many spikes each neuron had for each image, then each neuron was assigned to the number for which it spiked the most.

3.3 Testing

During testing 10 000 images were shown for 250ms, again with 150ms resting time. Using the classification previously generated, the prediction was then made as the label the most spiking neuron was assigned to. An accuracy measure was then generated by looking at the true label for the image, and calculating what percent of the time the prediction was correct.

4 Method

We started by investigating the effect of different forms of lateral inhibition, "winner takes all", soft lateral inhibition and no lateral inhibition all trained on 60 000 images with 196 output neurons. We then investigated the effect of reducing the network size to 36 output neurons. To test the effect of device variations we changed the neuron activation levels from a fixed 50 mV to a uniform distribution between 40 and 60 mV, and trained networks with and without adaptive threshold to see if this mitigates problems caused by variations. Another device non-ideality is non-linearity, so we tried replacing our STDP function with a non-linear one. Finally we trained a network on the MNIST Fashion dataset instead of numbers to see if it performed worse.

5 Results

5.1 Lateral inhibition

The network performance as a function of training images for different lateral inhibitions can be seen in figure 3.

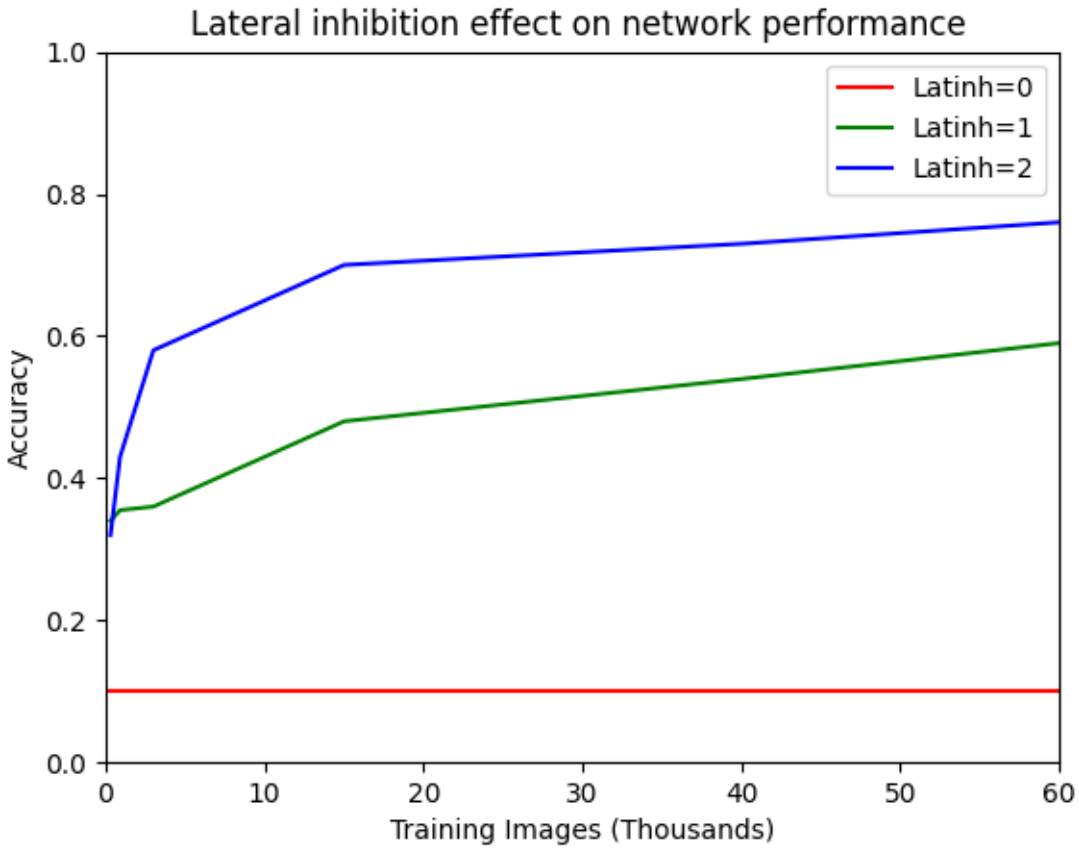

Figure 3: Network performance as a function of training images for "winner takes all" lateral inhibition (2, blue), soft lateral inhibition (1,green) and no lateral inhibition (0,red). All of the networks had N=196 neurons in the output layer and were otherwise the same.

Synapse maps for both "soft lateral inhibition" and "winner takes all" can be seen in Figure 4.

From Figure 3 we can see that the network with no lateral inhibition (0) performed no better than simply guessing the number. A synapse map for the network after 60000 iterations can be seen in Figure 5.

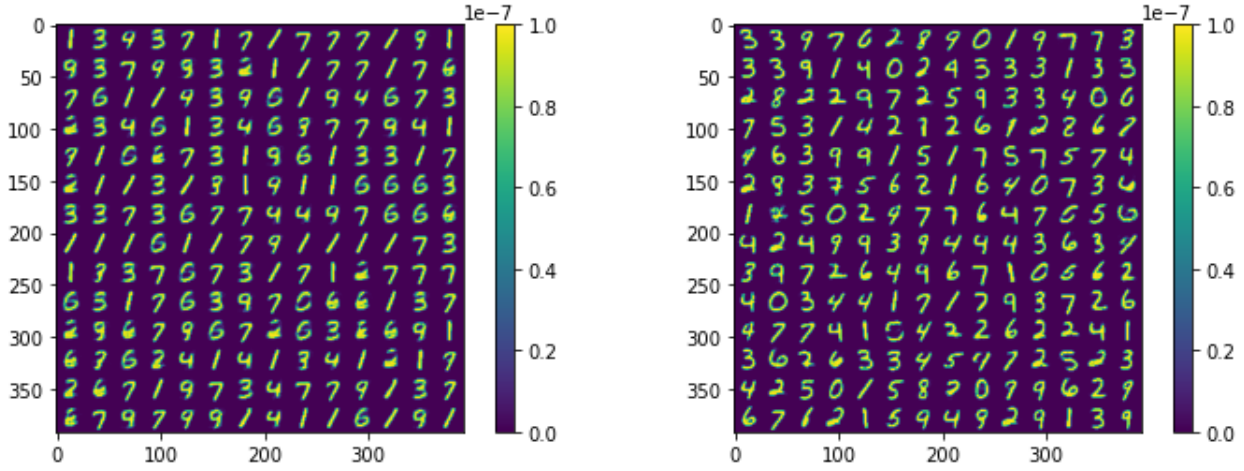

Figure 4: Synapse maps for "soft lateral inhibition" (left) and "winner takes all" lateral inhibition (right) after 60 000 training images. Both used $N = 196$ output neurons.

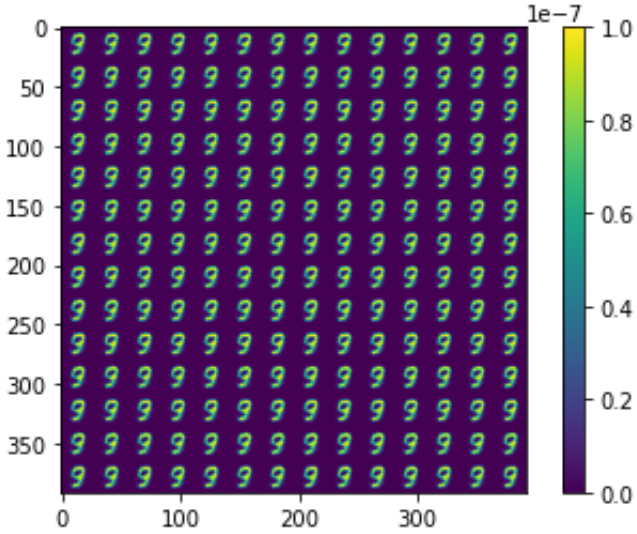

Figure 5: Synapse map for network with no lateral inhibition after 60000 training images with N=196 neurons in the output layer.

5.2 Network size

We trained one network with $N = 36$ output neurons to see the effect of a different network size, as compared to the reference with $N = 196$ output neurons. The performance as a function of training images for the networks can be seen in Figure 6.

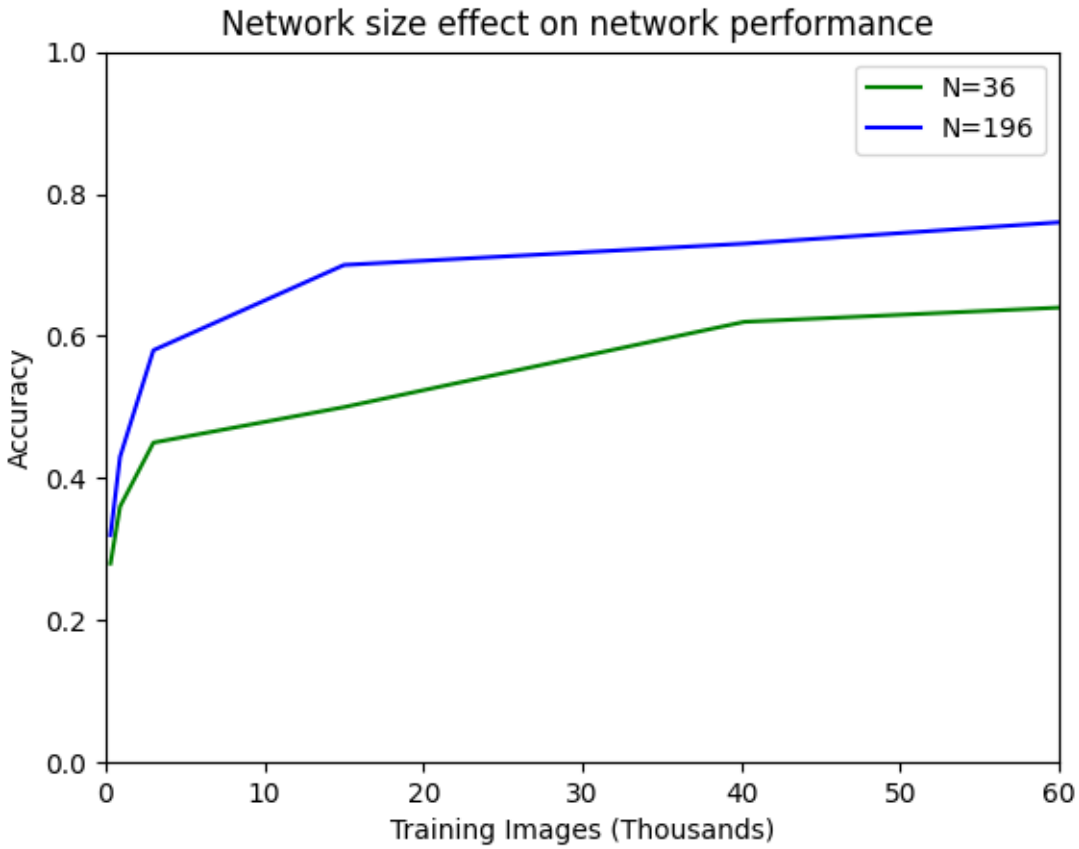

Figure 6: Network performance as a function of training images for two different output layer sizes, 36 (green) and 196 (blue). All of the networks had "winner takes all" lateral inhibition and were otherwise the same.

Figure 7 shows synapse maps for the different networks after 900 training images, and Figure **??** shows spike maps after 60 000 training images.

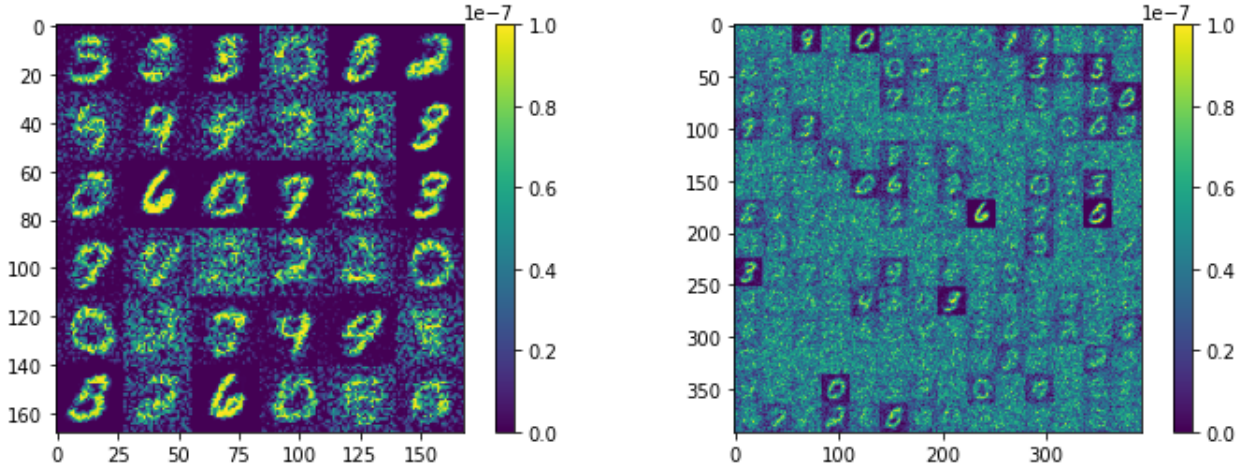

Figure 7: Synapse maps for N=36 (left) and N=196 (right) after 900 training images. Both used "winner takes all" lateral inhibition.

5.3 Adaptive threshold

Network performance as a function of training images with adaptive threshold and device variations can be seen in Figure 8.

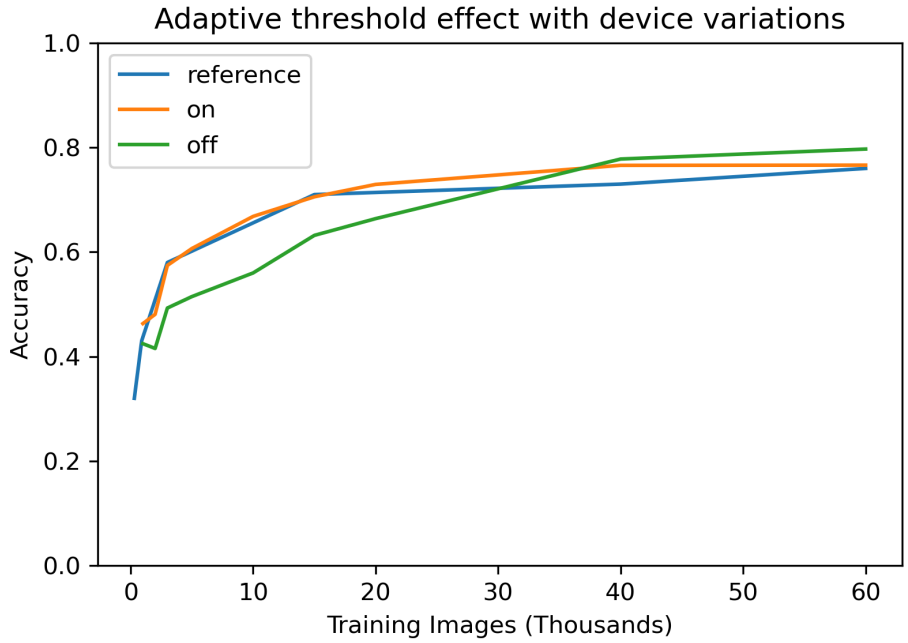

Figure 8: Network performance as a function of training images for adaptive threshold on (orange), off (green) with device variations and a reference network (blue) which has no device variations. All of the networks had N=196 and "winner takes all" lateral inhibition

Figure 9 shows synapse maps with and without adaptive threshold (with device variations).

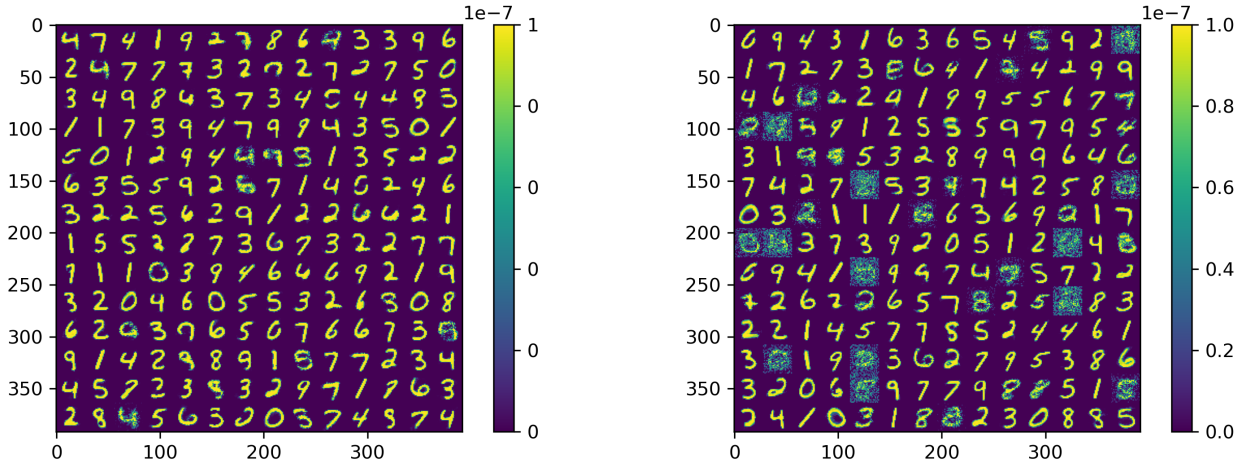

Figure 9: Synapse maps for the adaptive threshold on (left) and adaptive threshold off (right) networks with devices variations after 60 000 training images. Both of the networks had N=196 and "winner takes all" lateral inhibition.

5.4 Non-Linear STDP

Figure 10 shows the network performance when using a non-linear STDP function, and synapse maps for the network after 1000 and 60000 training images can be seen in Figure 11.

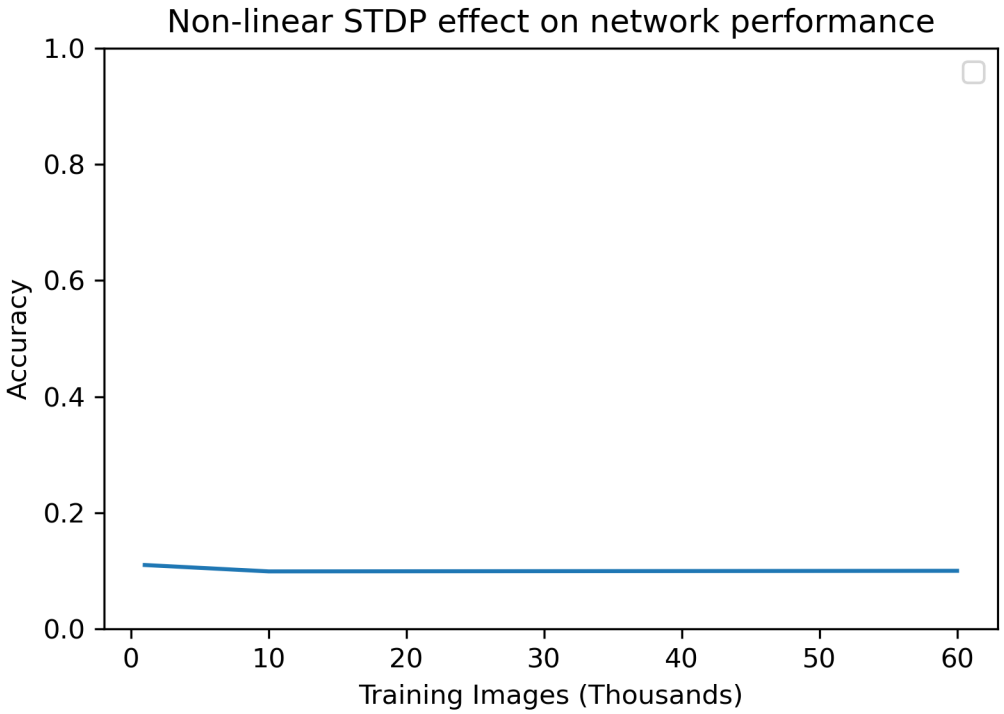

Figure 10: Network performance as a function of training images with non-linear STDP, N=196 and "winner takes all" lateral inhibition

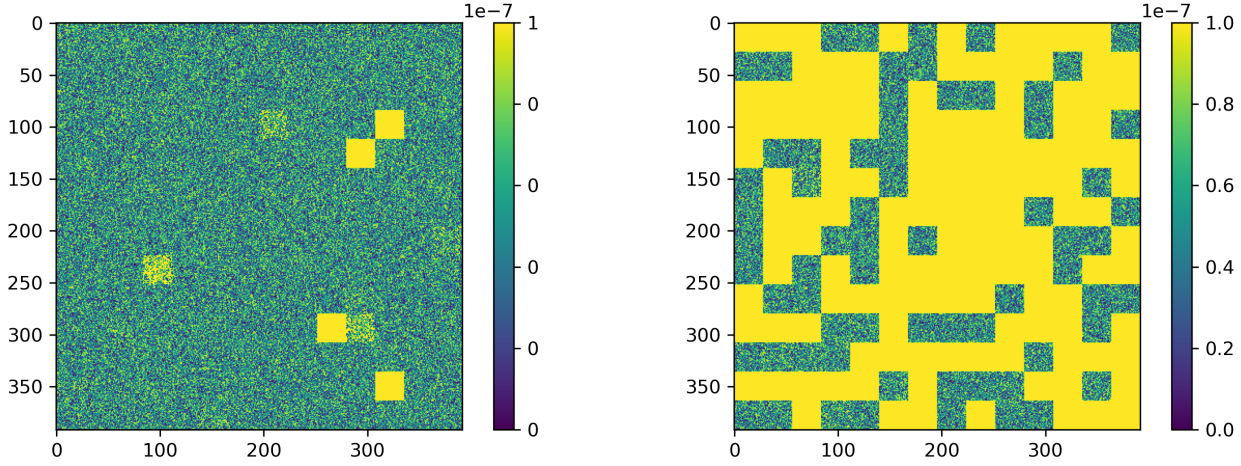

Figure 11: Synapse maps for the non-linear STDP network after 1000 training images (left) and 60 000 (right). N=196 and "winner takes all" lateral inhibition.

5.5 MNIST Fashion dataset

The accuracy as a function of training images can be seen in figure 12 and synapse maps for the MNIST Fashion dataset as well as the MNIST Numbers (for comparison) can be seen in Figure 13.

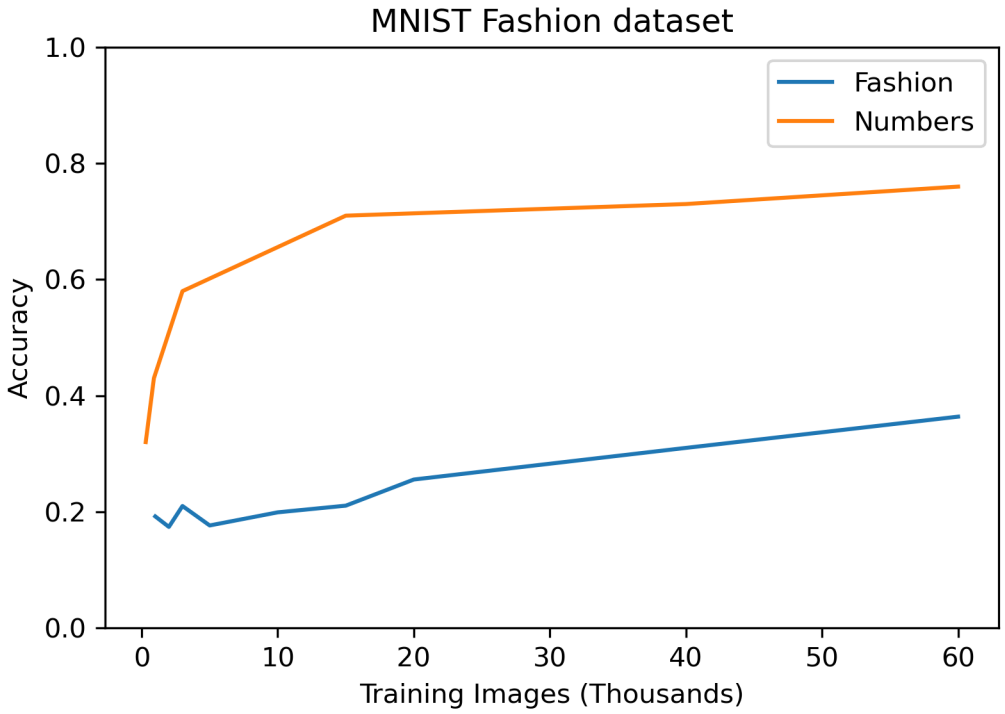

Figure 12: Network performance as a function of training images for the MNIST Fashion dataset (blue) and for comparison the MNIST numbers dataset (orange). Both of the networks had N=196 and "winner takes all" lateral inhibition

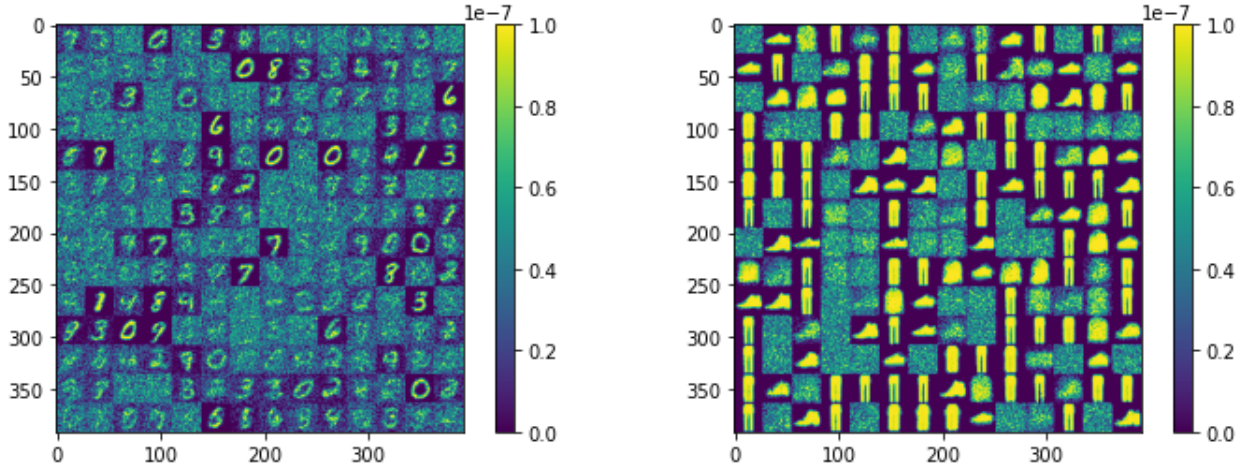

Figure 13: Synapse maps for the MNIST Numbers dataset and the MNIST Fashion dataset both after 2000 images. N=196 and "winner takes all" lateral inhibition for both networks.

A synapse map for the network after 60 000 images can be seen in Figure 14.

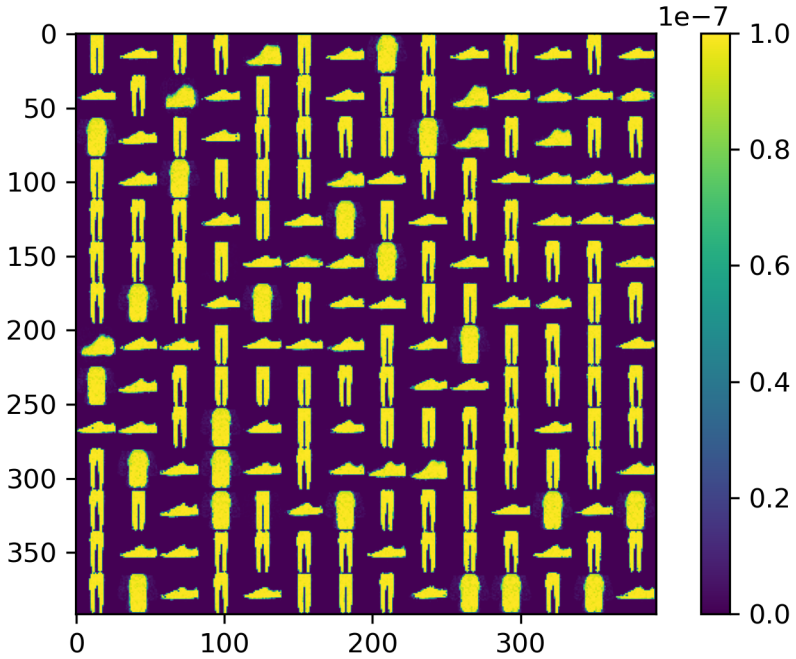

Figure 14: Synapse map for the MNIST Fashion dataset both after 60 000 images. N=196 and "winner takes all" lateral inhibition.

6 Discussion

6.1 Lateral inhibition

Figure 3 shows that "winner takes all" lateral inhibition performed significantly better for this problem than soft lateral inhibition for all datapoints. It appears that using lateral inhibition helped the neurons specialize at different things and thus improved performance. We can also see that not using any lateral inhibition the nodes all became focused on the same thing, see figure 5. This meant the network was no better than just guessing '9' on every input - giving an accuracy of 10%.

At first glance the synapse maps in figure 4 look similar, both "winner takes all" and soft inhibition seem to have low noise and recognisable digits. Looking closer at the soft inhibition map however one can find that 48 neurons (almost 25%) decided to specialise in the number 1. By contrast their appear to only be 3 neurons corresponding to 0 and these seem to partially detect 6 as well. The "winner takes all" network seems far more balanced with each number corresponding to roughly 1/10 of the output neurons. Since handwritten digits probably have a similar level of variation within a class having such an unbalance is probably a poor use of output neurons.

6.2 Network size

Figure 6 shows the network with 196 output neurons outperforming the much smaller network with only 36 output neurons. With fewer neurons we can represent fewer "versions" of numbers. With 196 neurons we could in theory have almost 20 different types of each number corresponding to say slightly, different curvatures, angles and positions, thereby increasing the probability that a number from the test set closely resembles one of them.

From Figure 7 we can see that already after 900 training images the smaller network does not have that many noisy areas left, compared to the larger network for which most neurons show random noise.

When training, we found it odd that the training speed did not seem to depend on the number of neurons, training a network with 36 output neurons took about as long as with 196 despite the network only containing 1/5 as many synapses. This is most likely due to unoptimized code rather than anything else as we had many other strange performance issues. If it were the case that it took 1/5 as long to train a network with 36 output neurons, as perhaps one would expect, one could debate whether the 36 neuron network would perform better than the 196 neuron network if trained 5 times as long(5 epochs) or if performance is fundamentally limited by output neuron count.

6.3 Device variations

Figure 8 shows the effect device variations had on performance as a function of images trained on. Initially the network with device variations but no adaptive threshold performs significantly worse than the network without variations and the network with variations and adaptive threshold. The

network with adaptive threshold performs closely in line with the network without variations which suggests adaptive threshold significantly mitigates issues the variations might cause. Somewhat surprisingly however is that both the network with and and without adaptive threshold perform better than the reference network after about 30000 images and that the network without adaptive threshold actually performs the best at 60 000 images. Looking at figure 9 which shows two synapse maps corresponding to adaptive threshold on or off at 60 000 images the one for adaptive threshold on looks "better" than the one for off. The one for on has barely any "noisy" areas i.e. neurons firing based off things which do not look like numbers, whereas the adaptive threshold off has several (around 15) neurons which seem very noisy. It could therefore be the case that the off network just happened to perform better this time and that if we trained the network many times the adaptive threshold on network would on average do better,

6.4 Non-Linear STDP

Having a non-linear (in conductance) STDP function, which could correspond to having a non-linear memristor, completely destroys network performance as can be seen in figure 10. The performance graph is pretty much just a flat line at 0.1, since there are 10 balanced classes this corresponds to just guessing at random. Looking at the synapse map in figure 11 we cannot identify any numbers, only completely yellow squares or noisy neurons. Comparing the map at 1000 images, which has a few fully yellow neurons, to the one at 60 000, which primarily has yellow squares, it seems like neurons are slowly trained to be activated by all pixels maximally as the training runs. It is quite typical that asymmetric non-linearities destroy performance. While non-linearities don't cause a huge reduction in performance on their own unless very large, assymetric non-linearities (which is what we have) cause catastrophic loss of performance even for relatively small non-linearities according to [1].

6.5 MNIST Fashion dataset

Figure 12 compares the performance of a size 196, "winner takes all" network on the MNIST Numbers and the MNIST Fashion dataset. While the network reaches 76 % accuracy after 60 000 images on the numbers it only reaches 0.36 % accuracy on the fashion dataset, very poor performance. While the numbers performance increases quickly at first before flattening out the fashion performance seems to increase fairly linearly throughout, indicating that we could perhaps improve performance by training for more epochs. That the network performed worse on the fashion dataset is not that suprising as it is generally considered to be a harder problem than numbers, both because there is more variation in each class and that some of the classes are quite similar.

Something we noticed was that for the fashion dataset the neurons seemed to reach "finished" states much faster. In figure 13 we can see the synapse matrices for both datasets after only 2000 images. While the numbers dataset has some identifiable numbers most of the output neurons still specialise in random noise, by contrast over half of the fashion dataset neurons have decided they want more from life. This could be reflective of the decreased performance, since images more

often trigger the wrong neuron these neurons will "denoise" faster than otherwise.

Figure 14 shows the synapse map for the fashion dataset after 60 000 images. From eyeballing it does not seem possible to differentiate 10 different types of neuron activations, instead it seems that several similar classes have blurred together.

7 Summary

It was found that "winner takes all" lateral inhibition worked better than softer inhibition (76% vs 60% accuracy after 60 000 images) and that the network with no inhibition was useless (10% accuracy). The network with 36 output nodes performed worse (64% accuracy) but that reducing the network size surprisingly did not appear to speed up training. Device variations decreased performance at a low number of trained images which adaptive threshold remedied but actually performed better than without variations after 60 000 images, though this is probably due to chance. A non-linear STDP function completely obliterated performance (10% accuracy), which is in line with results from [1]. Switching to the Fashion dataset significantly (36% accuracy) decreased performance but performance did seem to be steadily rising so simply training for more epochs might improve it considerably.

References

[1] Mattias Borg. *Lecture 13 - Neuromorphic device in reality*. 2022.

[2] Mattias Borg. *Lecture 7 – Spiking Neural Networks Part 2*. 2022.

[3] Jie Hu **andothers**. "Squeeze-and-Excitation Networks". **in***IEEE Transactions on Pattern Analysis and Machine Intelligence*: 42.8 (2020), **pages** 2011–2023. DOI: 10.1109/TPAMI.2019.2913372.

[4] Muhammad Tanveer, Muhammad Umar Karim Khan **and** Chong-Min Kyung. "Fine-Tuning DARTS for Image Classification". **in**(**june** 2020).