Accepted for publication in Proceedings of SPIE, published by SPIE. Please cite as: Patrick Maier, James Rainey, Elena Gheorghiu, Kofi Appiah, and Deepayan Bhowmik "Digit classification using biologically plausible neuromorphic vision", Proc. SPIE 13137, Applications of Digital Image Processing XLVII, 131370I (30 September 2024); https://doi.org/10.1117/12.3031280

Digit Classification using Biologically Plausible Neuromorphic Vision

Patrick Maier[a], James Rainey[b], Elena Gheorghiu[c], Kofi Appiah[d], and Deepayan Bhowmik[b]

[a]Division of Computing Science and Mathematics, University of Stirling, Stirling, UK
[b]School of Computing, Newcastle University, Newcastle upon Tyne, UK
[c]Division of Psychology, University of Stirling, Stirling, UK
[d]Department of Computer Science, University of York, York, UK

ABSTRACT

Despite tremendous advancement in computer vision, especially with deep learning, understanding scenes in the wild remains challenging. Even modern image classification models often misclassify when presented with out-of-distribution inputs despite having been trained on tens of millions of images or more. Moreover, training modern deep-learning classifiers requires a lot of energy due to the need to iterate many times over the training set, constantly updating billions of model parameters. Owing to problems with generalisability and robustness as well as efficiency, there is growing interest in computer vision to mimic biological vision (*e.g.*, human vision) in the hope that doing so will require fewer resources for training both in terms of energy and in terms of data sets while increasing robustness and generalisability. This paper proposes a biologically plausible neuromorphic vision system that is based on a spiking neural network and is evaluated on the classification of hand-written digits from the MNIST dataset. The experimental outcome indicates improved robustness of the proposed approach over state-of-the-art considering non-digit detection.

Keywords: Neuromorphic vision, digit classification, spiking neural network, human vision system.

1. INTRODUCTION

Prompt reactions to adversarial conditions are critical features for safe and reliable interaction with the environment. Computer vision, which plays a significant role in such interaction processes, requires a higher level of scene understanding with fast response capabilities. State-of-the-art deep learning-based approaches are not adaptive to unseen changes in the environment, computationally intensive and prone to fail in complex scenarios. This paper proposes a biologically plausible neuromorphic computational vision approach that mimics the computational neuroscience of human vision and examines its application in a prototypical digit classification problem.

Computational neuroscience has produced decades of insights into the principles and structure of neural networks that realise vision in animals, including humans. Biological neural networks are massively parallel analogue computers that communicate by transmitting electrical spikes, hence they are termed *spiking neural networks* (SNNs). A characteristic property of SNNs is that their energy consumption is directly related to spiking frequency, and that the energy consumption of neurons that are not spiking is very low. This explains the comparatively low energy consumption of the human brain, despite having billions of neurons and trillions of synapses.

In this paper, we investigate and extend a previously studied SNN by Diehl and Cook[1] designed for classifying hand-written digits from the MNIST dataset. This SNN consists of a simple two-layer architecture of excitatory

Further author information: (Send correspondence to Deepayan Bhowmik)
Patrick Maier: E-mail: patrick.maier@stir.ac.uk
James Rainey: E-mail: James.Rainey@newcastle.ac.uk
Elena Gheorghiu: E-mail: elena.gheorghiu@stir.ac.uk
Kofi Appiah: E-mail: kofi.appiah@york.ac.uk
Deepayan Bhowmik: E-mail: deepayan.bhowmik@newcastle.ac.uk

and inhibitory neurons and implements a biologically plausible unsupervised competitive learning approach, similar to self-organising maps, that learns representations of digits by adapting the input weights of excitatory neurons using biologically plausible learning rules.

While the learning algorithm of Diehl and Cook is biologically plausible, the learned representations, maps of unoriented pixel intensities, are poor. It is well known that neurons in the visual cortex respond not just to the intensity of light but to simple visual features such as oriented edges and bars. Hence, we propose an improvement on the state-of-the-art by coupling the SNN by Diehl and Cook with convolutional filters that are selective to orientation. We investigate how using such filters affects the training behaviour of the network as well as its robustness when confronted with unclassifiable inputs.

Our contributions are as follows:

- A comparison of the effectiveness of pixel-based vs. convolution-based representations of digits in the context of unsupervised competitive learning. We find that, while pixel-based representations tend to reach the final accuracy more rapidly, convolution-based representations improve more steadily over time, avoiding big fluctuations in accuracy as the network trains.
- An investigation of the robustness of pixel-based vs convolution-based representations. We find that pixel-based representations are not robust when exposed to unclassifiable inputs, misclassifying almost all such inputs as digits. By contrast, convolution-based representations, when trained long enough, misclassify less than 10% of such inputs as digits.
- A novel and biologically plausible means of improving robustness for competitive SNNs by deliberately *untraining* (i.e. resetting to a random initial state) a subset of the neuron population post training. In competitive SNNs, these untrained neurons act as representations of unclassifiable inputs. Untraining 10% of the population lowers the rate of misclassified error inputs to between 3% and 5%, for both pixel-based and convolution-based representations. Thanks to a careful selection of the sub-population, untraining has virtually no impact on the digit classification accuracy of the network.

2. RELATED WORK

2.1 Biological Vision

2.1.1 Object and shape processing

Psychophysical and neurophysiological studies have shown that object processing occurs across multiple stages in the visual cortex, with different visual areas processing a wide variety of visual features. The visual analysis of objects begins with the detection of the orientations and positions of local parts of contours. This is accomplished by orientation-selective V1 neurons[2–4] which respond selectively to local/specific features (e.g. oriented edges and bars) of complex stimuli. The outputs of a small subset of these V1 neurons are then combined to form parts-of-shapes (curves, angles)[5] that are encoded by neurons in intermediate visual areas V2[6–8] and V4.[9–13] These parts-of-shapes are further combined to form whole-shapes which are encoded by neurons in higher visual areas IT and LOC[14–18] that are selective to global simple shapes (e.g., circle, square, etc.).

2.1.2 Texture processing

Computational models of texture processing often refer to different processes: texture detection, discrimination and segmentation. It is well established that texture-modulation detection is mediated by relatively low-to-intermediate level processes that detect variations in contrast energy within narrow-band spatial-frequency and orientation-selective channels, as modelled by the Filter-Rectify-Filter cascade,[19–22] which indiscriminately summate signals across different features/channels.

A standard model of texture processing is the filter-rectify-filter (FRF) model,[23] also known as 'non-Fourier', 'back-pocket' or 'complex channel' model. The model consists of two stages. The first stage computes a generic image representation that resembles spatial processing characteristics such as spatial frequency and orientation tuning of neurons in early visual areas. The input image is represented at a range of different spatial resolutions or scales (this involves low-pass filtering/blurring the image and subsampling), with each resolution level being

decomposed into a set of four small orientation-selective components/filters. Thus, the first-stage filters are linear and narrow-band and convolve the input image at several spatial scales and orientations. The outputs of the first-stage filters are then subjected to a non-linearity, such as full-wave rectification or squaring, which involves summing of the half-wave rectified responses of pairs of filters. The purpose of this non-linearity is to make all excursions from the mean level positive such that first-stage filter responses do not cancel each other out when linearly pooled at the second stage. The second stage of the model involves the computation of more specific properties which are done by second-stage filters that have larger receptive fields than their first-stage counterparts, but are otherwise close analogues. The second-stage filters sum the non-linearly transformed first-stage outputs.

Different FRF models differ mainly in the form of their intermediate non-linearity (e.g., full-wave rectification, half-wave rectification, squaring, etc.) and in the manner in which the outputs of the second-stage filters are combined prior to the decision stage. The choice of different types of non-linearities is mainly related to the task (e.g., texture modulation detection, texture segmentation). For example, second-stage filters with orthogonally tuned first-stage inputs are subtracted to produce an orientation-opponent signal, resulting in a representation at the second stage of normalised opponent measures.[24]

Our digit-classification application uses four orientation-selective filters in stage 1 followed by a set of fully-connected neurons. Outputs of stage 1 are combined before they feed into stage 2, consisting of a set of neurons representing ten digit classes (0-9).

2.2 Digit classification using SNNs

Classification of handwritten digits has been a benchmark problem in machine learning and neural network for decades.[25] Traditionally, artificial neural networks (ANNs) have been employed to achieve high accuracy in digit recognition tasks, but more recently, a more powerful alternative known as spiking neural networks (SNNs) have emerged. SNNs, which are inspired by the biological neurons' spiking behaviour and attempt to emulate the remarkable energy efficiency of the brain in processing information.[26] Unlike ANNs, which use continuous activation function, SNNs use spikes or discrete events to transmit information.[27]

Several neuron models are used in SNNs, with the most prominent being the Leaky Integrate-and-Fire (LIF) model, the Hodgkin-Huxley model and Izhikevich model.[28] The LIF model is particularly popular in digit classification tasks due to its simplicity and efficiency in simulating neuron behaviour. Directly training SNNs is more challenging due to the non-differentiable nature of spike events. However, several methods have been proposed to overcome this. Spike-Timing-Dependent Plasticity (STDP) is a biologically inspired learning rule that adjusts synaptic weights based on the timing of spikes.[29] One of the prevalent approaches in employing SNNs for digit classification is converting pre-trained ANNs into SNNs. This method involves training a traditional ANN and then mapping its weights and structure to an SNN.

Vision datasets have been used to compare SNNs and recurrent neural networks (RNNs). A series of experiments were conducted on two types of neuromorphic datasets (N-MNIST and DVS-Gesture) to conclude that SNNs generally achieve better accuracy than RNNs, even though accuracy of RNNs increases with rate-coding-inspired loss functions.[30] Li and Meng (2023)[31] presented a deep spiking neural network with improved back-propagation, and the experiments conducted achieved recognition accuracies of 98.43%, 96.70%, 98.81% and 93.61% respectively on the image classification datasets MNIST, Fashion-MNIST, Kuzushiji-MNIST and CIFAR-10.

SNNs represent a promising frontier in the classification of handwritten digits, offering advantages in terms of biological plausibility, energy efficiency, and potentially faster processing times.[27] While challenges remain, particularly in the areas of training complexity and hardware implementation, ongoing research is paving the way for SNNs to become a viable alternative to traditional ANNs in various applications.

3. METHODOLOGY

We investigate the behaviour of the MNIST digit-classifying SNN of Diehl and Cook (2015)[1] by evaluating its learning speed, accuracy and ability to detect erroneous inputs. We will contrast their original network with our proposed variant by modifying the retina layer to a primary visual cortex area V1 layer which consists of neurons

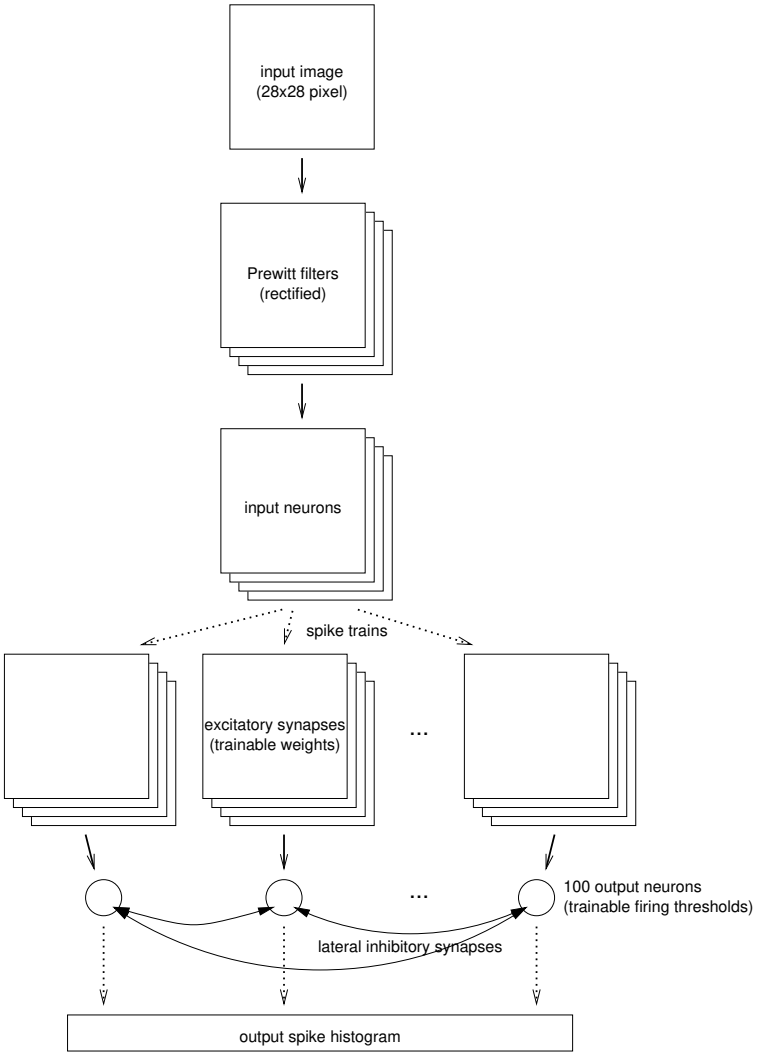

Figure 1. Proposed network architecture, extending the architecture of Diehl and Cook (2015)[1] by a filter layer.

sensitive to oriented features. In addition, we will contrast fully trained networks with networks that deliberately "untrain" a subset of neurons and use untrained neurons for detecting erroneous (i.e. non-digit) input images.

3.1 Background: Unsupervised learning of digits using SNN

This section reviews the SNN architecture and training method proposed by Diehl and Cook (2015)[1] and implemented in the Brian 2 SNN simulator.[32]

Figure 1 depicts the structure of our proposed network, largely following the architecture of Diehl and Cook (2015)[1] except for the addition of a filter layer before the input neurons. Our network applies edge-detecting filters (see Section 3.2) to the 28x28-pixel input image. A layer of input neurons converts the filter outputs into spike trains. The input layer is fully connected via excitatory synapses to a layer of 100 output neurons. The synaptic weights are initialised at random and adjusted during training to learn specific input patterns. Each of the output neurons is also connected to all other output neurons by inhibitory synapses. Thus, the firing of an output neuron will, after a short delay, laterally inhibit the firing of other output neurons. This structure creates a competition between output neurons. Those neurons responding strongest to a given input will fire fastest and therefore drive their inhibitory synapses to suppress the firing of output neurons that responded less strongly to the input. We refer the reader to Diehl and Cook (2015)[1] for details on the dynamics of neurons and synapses.

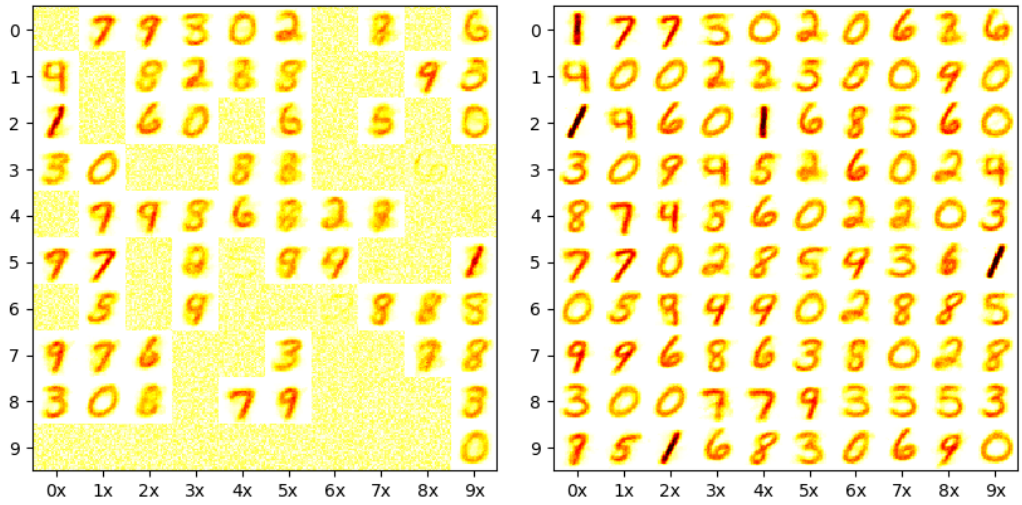

Figure 2. 100 neurons trained for 1000 images (left) and 3000 images (right).

Training the network proceeds in two phases. During a first, unsupervised training run, the weights of the excitatory synapses and the firing thresholds of the output neurons are adjusted using an STDP-based learning rule. Then, synaptic weights and firing thresholds are fixed and a second, supervised training run learns a probabilistic digit classification for each output neuron. Finally, prediction is based on mixing the probabilistic classifications, weighted by the output spike histogram for the given input.

Diehl and Cook (2015) argue that their network is based on biologically plausible principles, including the use of unsupervised STDP for learning synaptic weights and firing thresholds and the use of lateral inhibition to reduce the number of output spikes.[1] The latter is very effective; a trained network will see only a small fraction (typically between 2 and 4) of the output neurons spike on an given input. This sparsification of high-dimensional sensory inputs is a main factor for the the superior energy-efficiency of visual signal processing in biological systems.

3.1.1 Training phase 1: Unsupervised clustering of input patterns

The first training phase exposes the network to images from the MNIST training dataset without considering their labels. Each image is exposed for 350 milliseconds (in simulation time), then the inputs are darkened for 150 milliseconds to quieten the network, before the cycle repeats with the next training image.

Exposing the input image will cause some of the output neurons to spike (which will drive their inhibitory synapses to subsequently suppress the firing of other output neurons). An STDP-based learning rule is used to adjust the excitatory synaptic weights of only those output neurons that did spike, and they are adjusted to become a little more similar to the current input, thus strengthening the spiking neurons' future responses to similar inputs. Over time, each neuron's excitatory synaptic weights will come to resemble a handwritten digit, and will act as centroid of the cluster of digits to which the neuron responds.

To illustrate, Figure 2 shows heat maps of the excitatory synaptic weights of all 100 output neurons after having being trained on 1000 and 3000 MNIST images, respectively. After 1000 training samples, the synaptic weights of just over half of the neurons have been adjusted to resemble handwritten digits (some more accurately than others) whereas the synaptic weights of the other half remain untrained because those neurons have spiked rarely or never. After 3000 training samples, synaptic weights of all 100 neurons have been trained, and they all resemble handwritten digits (though some appear ambiguous and could represent multiple digits, e.g. 4s as well as 9s).

Besides excitatory synaptic weights, STDP is also used to adjust the firing thresholds of output neurons, raising the thresholds of neurons that spike often, thereby reducing their firing rate. This is also biologically motivated, and ultimately makes firing rates across the neuron population more uniform.

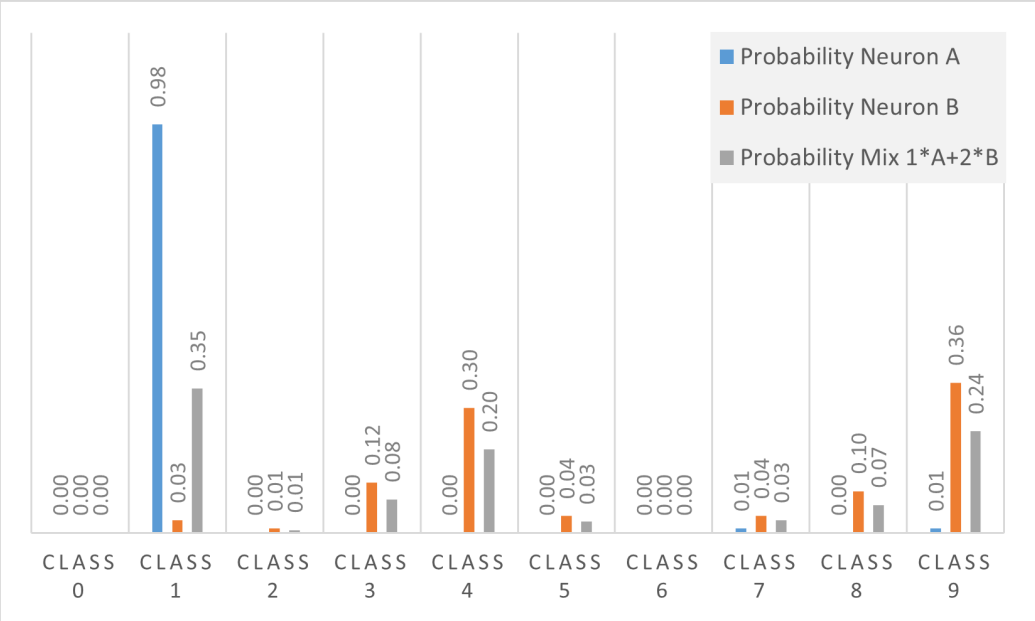

Figure 3. Probability distributions of two neurons (A and B) derived from spike count histograms. The (hypothetical) third distribution is a mix of A and B.

3.1.2 Training phase 2: Supervised labelling of output neurons

After training phase 1, synaptic weights and firing thresholds are fixed. In order to assign meaning to the learned input patterns, the training dataset is exposed to the network again, image by image, for the same durations as before, while gathering spike count histograms, indexed by the training images' labels (the classes 0 to 9), for each output neuron. After completing the training run, each neuron's spike count histogram defines a discrete probability distribution over the classes 0 to 9. We call the probability of the mode of each neuron's distribution the neuron's *confidence*. The higher the confidence, the more selective the neuron, that is, the more likely it responds to a single digit only.

Figure 3 shows examples of two such probability distributions. Neuron A recognises digit 1 with very high confidence of 0.98. On the other hand, neuron B recognises digit 9 with very low confidence, as the probability of class 9 is barely higher than that of class 4, causing neuron B to mix up 4s and 9s often.

3.1.3 Prediction: Classification of digits

The procedure for classifying unseen inputs is similar to the supervised training step. The fixed-weights network is exposed to the input image for 350 milliseconds, followed by 150 milliseconds of darkness. Spike counts are recorded for all 100 output neurons. The output probability distribution is computed as the spike-count-weighted mixture the neuron distributions that were calculated during the supervised training phase. The *predicted class* is the mode of the output distribution and the *confidence* of the prediction is the probability of the mode.

To illustrate this with a hypothetical example, suppose that neurons A and B in Figure 3 spike once and twice, respectively, on a given hypothetical input, and no other neuron spikes. The output distribution is the mix 1*A+2*B shown in Figure 3, with mode class 1 and confidence 0.35. That is, the network classifies the input as a digit 1 based on the high confidence of neuron A despite the low-confidence neuron B having reacted more strongly to the input.

As an aside, Diehl and Cook (2015) opted for a simpler majority-vote classifier. After the supervised training phase, they label each neuron with the mode of its probability distribution. When classifying, they count the spikes per class, and the class with most spikes wins.[1] The downside of this method is that it is more difficult to define a meaningful measure of prediction confidence. Moreover, low confidence neurons are more likely to

Table 1. Four network variants, representing pixels or vertical/horizontal edges.

Convolutions	Operators	Kernel size	Stride	Represented features
28x28	identity	1x1	1	28x28 pixels; the network of Diehl and Cook (2015)
14x14	smoothing	2x2	2	14x14 smoothed pixels
28x28x2	Prewitt	3x3	1	28x28x4 rectified vertical and horizontal edges
14x14x2	Prewitt	3x3	2	14x14x4 rectified vertical and horizontal edges

cause mispredictions. For instance, the majority-vote classifier would have predicted class 9 instead of 1 for the hypothetical scenario depicted in Figure 3.

3.2 Sensitivity to oriented features

The input layer of the network by Diehl and Cook (2015) is sensitive to 28x28 pixel intensities.[1] To examine whether an orientation-sensitive input layer improves the performance of the network, we apply convolutional filters to the input image, so that the resulting network learns representations of horizontally or vertically oriented features rather than pixel intensities. As there are multiple orientations to consider, learning oriented features increases the number of input neurons and synaptic weights and thus the computational resources required by the network. For a fair comparison in terms of computational resources, we also investigate variants of the network that reduce the image resolution such that the resulting network requires the same number of input neurons and synaptic weights for learning oriented features as the original network by Diehl and Cook (2015).

Table 1 lists the four network variants that will be investigated in Section 4. Each variant is obtained from the original network of Diehl and Cook (2015)[1] by applying different convolutional filters to the 28x28-pixel input image.

- The first network (row 1) applies 28x28 identity filters (i.e. it does not apply any filters at all) resulting in the original network of Diehl and Cook (2015) that learns pixel intensities.
- The second network (row 2) applies the following 2x2 smoothing kernel

$$\begin{bmatrix} 1/4 & 1/4 \\ 1/4 & 1/4 \end{bmatrix}$$

  with a stride of 2 pixels, effectively reducing the image resolution to 14x14 pixels (with intensity ranging from 0 to 255). Compared to Diehl and Cook (2015), this network requires only a quarter of the input neurons and synaptic weights.
- The third network (row 3) applies (normalised) Prewitt operators[33] for detecting vertical and horizontal edges. The operators are represented by the following 3x3 kernels

$$\begin{bmatrix} 1/3 & 0 & -1/3 \\ 1/3 & 0 & -1/3 \\ 1/3 & 0 & -1/3 \end{bmatrix} \qquad \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 \\ -1/3 & -1/3 & -1/3 \end{bmatrix}$$

  The Prewitt filter outputs range from -255 to 255, yet the input neuron expect non-negative values. Hence, we duplicate the number of filters by including the inverses of the above Prewitt kernels, and rectify filter outputs with a ramp function (see Figure 4 for an example of applying the edge detecting filters to an image of the digit 6). Thus, the filters yield 28x28x4 edge features (ranging from 0 to 255), requiring four times as many input neurons and synaptic weights as the network of Diehl and Cook (2015).
- The final network (row 4) applies the same edge-detecting rectified Prewitt operators with a stride of 2 pixels, resulting in a lower resolution of 14x14x4 edge features. Consequently, this network requires the same amount of input neurons and synaptic weights as the original network of Diehl and Cook (2015).

Figure 4. Left: image of digit 6. Right: Prewitt kernels applied to the image; vertical/horizontal (top row) and their inverses (bottom row).

Section 4 investigates how the different filters affect training speed of the network as well as the accuracy of the fully trained network (both for classifying digits and discriminating digits from non-digits). Note that the cost of the convolutional filters is negligible, as the kernels are simple (3 additions, 3 subtractions and 1 division for a Prewitt kernel) and the filters are computed only once per input image. In fact, the computational cost of these SNNs is roughly proportional to the number of input neurons, as more input neurons tend to generate a higher total number of spikes.

3.3 Detecting erroneous inputs

An important property of robust classifiers is their ability to detect when classification is impossible because the input is too far outside the distribution of training samples. Experiments with applying the classifiers to random images from the CIFAR-100 dataset demonstrate that the original network by Diehl and Cook (2015) is not a robust classifier, misclassifying almost all CIFAR-100 images as digits (see Section 4.3).

There are two ways in which an SNN-based classifier can be made more robust.

1. By not responding to erroneous inputs. That is, no neuron should spike on inputs that are not handwritten digits. Section 4.3 shows that sensitivity to oriented features improves robustness by suppressing most responses to erroneous inputs.

2. By introducing an additional class for recognising error inputs. That is, dedicated neurons responsible for detecting erroneous inputs should spike.

Here, we outline how to realise the latter approach by dedicating a subset of neurons to detecting errors. Importantly, this group of neurons is not trained on samples of error images, as we assume that there are no representative samples of erroneous inputs — by definition, erroneous inputs are those that fall far outside the distribution of training samples.

The idea is to use "untrained" neurons to detect erroneous inputs. The synaptic weights of untrained neurons are random, and therefore these neurons will respond to random input patterns. On regular MNIST inputs, the firing of these untrained neurons will be suppressed by the stronger responses of trained neurons. However, on error inputs the response of some untrained neurons is likely stronger than the responses of trained neurons. In that case, the spiking untrained neurons will suppress the firing of trained neurons via lateral inhibition, thereby preventing misclassification of the input as a digit.

Technically, we achieve the effect of including untrained neurons in the population by resetting a subset (10 out of 100) of output neurons. Resetting a neuron means resetting its synaptic weights to its initial random values, and resetting the firing threshold to the median of the population. The effect of "untraining" 10 neurons by resetting their synaptic weights is visualised in Figure 5.

Resetting the firing threshold to the population median will make the reset neuron roughly as sensitive as trained neurons. That is, the firing threshold is neither so high that the reset neuron never spikes, nor so low that

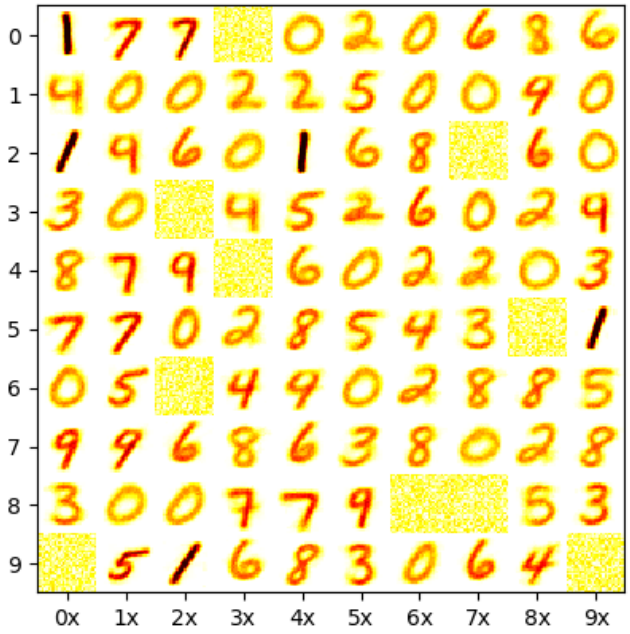

Figure 5. 100 neurons trained on 6000 digits, then 10 neurons have their synaptic weights reset to random values.

it spikes all the time. Resetting a neuron's synaptic weights to random values will make the neuron unlikely to respond strongly to any image in the training set. However, some reset neurons may respond more strongly than trained neurons to images that are *not* part of the training set; if the response is strong enough to trigger the reset neuron to spike, lateral inhibition will prevent trained neurons spiking, thereby preventing misclassification.

Which 10 neurons should be reset? It makes sense to reset neurons that are unlikely to contribute much to the accuracy of digit classification. We use the confidence of each neuron's probability distribution (determined after training phase 2, see Section 3.1.2) as a proxy for the quality of that neuron's contribution to classification. Therefore, we rank all 100 neurons by their confidence and reset the 10 neurons with lowest confidence values. Section 4 investigates how resetting those 10 neurons affects the accuracy of classifying digits and discriminating digits from non-digits.

4. RESULTS AND DISCUSSION

In this section, we evaluate 4 different networks of 100 output neurons. Two of the networks are sensitive to pixel intensities, the original one by Diehl and Cook (2015) with a resolution of 28x28 pixels, and a variant with reduced resolution of 14x14 pixels. The other two networks are sensitive to horizontal and vertical edges, one with a resolution of 28x28x2 Prewitt filters, and one with a reduced resolution of 14x14x2 Prewitt filters. Due to the representation of Prewitt filters (Section 3.2), the networks based on Prewitt filters use four times as many computational resources as the corresponding pixel-based networks of the same image resolution.

We find that all four networks eventually converge to same digit classification accuracy of between 82 and 83%, which is the same accuracy that was reported by Diehl and Cook (2015) for a network of 100 output neurons.[1] Our experiments are concerned with how the networks differ in terms of learning speed and in terms of robustness.

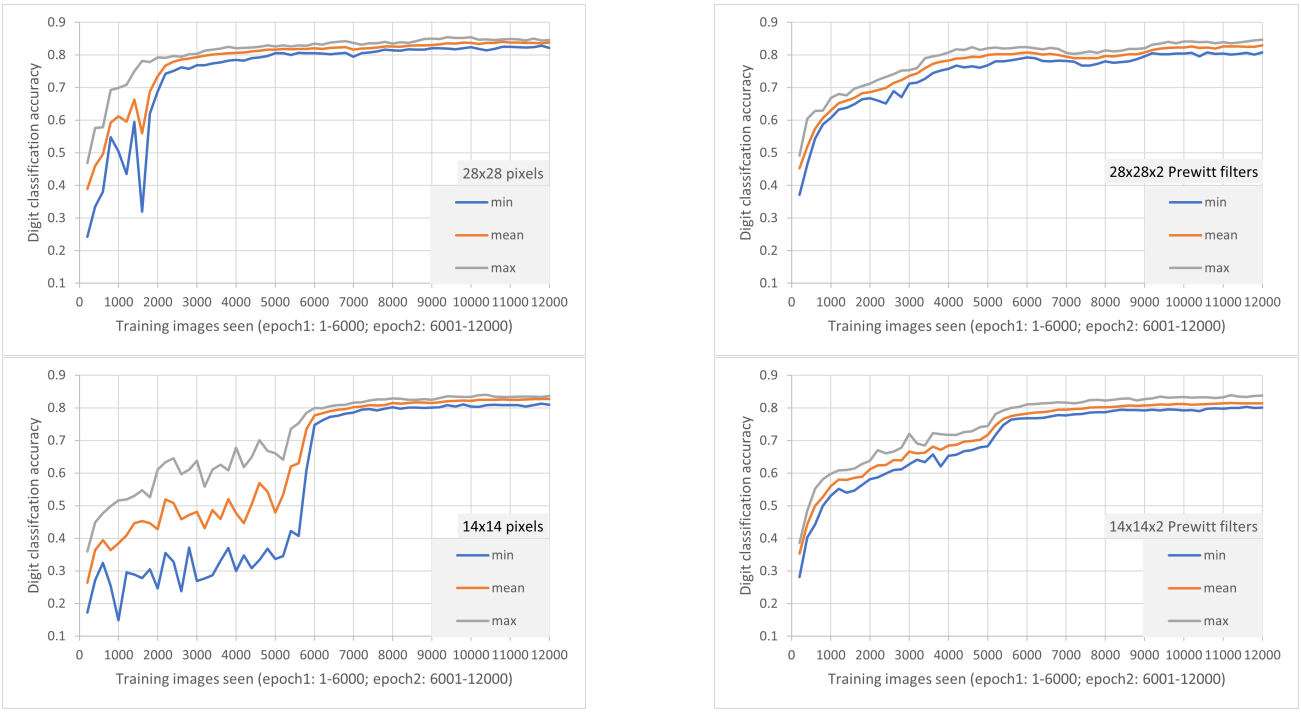

Figure 6. 10-fold cross validation of digit classification accuracy of 4 networks as training progresses. The left column shows pixel-based SNNs, the right column SNNs based on oriented features. The top row shows high-resolution SNNs; the bottom row shows low-resolution SNNs.

4.1 Speed of training convergence

We first investigate how fast these networks learn, that is how accuracy depends on the number of training images seen. We trained each network for 2 epochs on 6000 images* of the MNIST training set. Every 200 training images, the network was frozen and run to classify all 10000 MNIST test images. The process was repeated 10 times with 10 different sets of 6000 MNIST training images.

Figure 6 reports the minimum, mean and maximum classification accuracy (i.e. the fraction of test images classified correctly) over time for each of the four networks. The data shows that all networks approach peak accuracy before the end of the first epoch. However, both the speed of convergence and the variability of accuracy vary between networks.

- The high resolution pixel-based network approaches peak accuracy earliest, after training on about a third of the images. The other networks approach peak accuracy only after training on about 80 to 100% of the images. Three of the networks show a marked jump in accuracy when approaching the peak. This jump coincides with the point where all neurons have had their synaptic weights sufficiently trained to resemble some images from the training set.

- The pixel-based networks show very high variability in accuracy prior to converging to peak accuracy. By contrast, the networks sensitive to oriented features show much less varibility as training progresses. Thus, only networks based on oriented features offer a predictable trade-off between traininig set size and accuracy.

4.2 Impact of resetting neurons on accuracy and confidence of classifying MNIST digits

In order to dedicate some of the neuron population to detecting erroneous inputs that are not digits, we "untrain" 10 of 100 neurons by resetting their synaptic weights to their initial random state. To limit the impact on

*In the interest of faster training, we increased the STDP learning rates of all networks by a factor of 10 compared to Diehl and Cook (2015).[1]

Table 2. Accuracy of classifying MNIST digits.

SNN	Accuracy	Confidence		
		Q1	Q2	Q3
28x28	0.867	0.60	0.80	0.89
28x28 (reset)	0.865	0.63	0.79	0.86
28x28x2	0.834	0.56	0.75	0.85
28x28x2 (reset)	0.835	0.57	0.75	0.85
14x14	0.848	0.54	0.75	0.89
14x14 (reset)	0.860	0.56	0.75	0.86
14x14x2	0.853	0.55	0.76	0.89
14x14x2 (reset)	0.840	0.57	0.74	0.89

classification accuracy, we pick the 10 neurons with least confidence. In this section, we evaluate the impact of resetting those 10 neurons on the accuracy and the confidence of digit classifcation.

We train each of the four networks on the same set of 6000 images from the MNIST training set, for 2 epochs. To benchmark accuracy, we classify 1000 images from the MNIST test set, and we repeat the experiment after resetting the 10 least confident neurons. Both the training set and the testing set are balanced, that is, each class makes up exactly 1/10th of the training and the testing set.

Table 2 reports the overall accuracy (the fraction of correctly classified digits) as well as the distribution of confidence levels of the classification (first, second and third quantile of confidence levels). We find that resetting the 10 least confident neurons has very little impact on overall accuracy. For each network, overall accuracy stays within about 1 percentage point either way.

Resetting the 10 least confident neurons may have a small impact on the confidence levels of the classification. More precisely, while there is little change in the median and the upper quartile of the confidence levels, the lower quartile of the confidence levels is raised slightly for all networks. This implies that confidence levels at the lower end are influenced by the low-confidence neurons, and that resetting those low-confidence neurons removes some negative influence on the confidence of classifications.

Figure 7 shows confusion matrices for the high-resolution networks, without (top row) and with (bottom row) resetting neurons, demonstrating that resetting the 10 least confident neurons barely alters the classification. In particular, all confusion matrices add to 1000, i.e. all 1000 input images were classified as digits.

These confusion matrices highlight some of the reasons for the limited digit classification accuracy of the networks, showing that 4s and 9s are often mixed up and 2s are often misclassified as 8s. The matrices also reveal differences between the pixel-based and the Prewitt-filter-based SNN: the latter is better at recognising 4s but worse at recognising 0s, 3s and 9s.

4.3 Detecting non-digits

Next, we compare how the four networks react when presented with non-digits. That is, we 'classify' 1000 randomly selected images from the CIFAR-100 dataset.[†] To measure accuracy, we report the fraction of CIFAR-100 images that are correctly not classified. We repeat the experiment after resetting the 10 least confident neurons and interpret the reset neurons as detectors of the non-digit class. To measure accuracy in the second set of experiments, we count the number of CIFAR-100 images that are correctly not classifed and the number of CIFAR-100 images that are classified as non-digits. Table 3 reports the overall accuracy, and the breakdown into the fraction of unclassifed images (no neuron spiked) and images classifed as non-digits.

We find that the pixel-based networks are unable to detect non-digits, misclassifying almost all CIFAR-100 images as digits. An analysis of the confusion matrix shows that they misclassify about two thirds of the images

[†]The CIFAR-100 images are 32x32 pixel RGB images. In order to fit the dimensions of the MNIST dataset, we convert the CIFAR-100 images to grayscale and crop the central 28x28 pixels.

28x28 pixels

Actual digit \ *Predicted digit*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	98	1	0	0	0	0	1	0	0	0
1	0	99	0	0	0	0	1	0	0	0
2	8	0	75	3	0	0	1	1	12	0
3	0	0	1	91	0	3	0	1	3	1
4	1	4	2	1	76	0	2	2	0	12
5	7	0	0	3	1	83	2	0	3	1
6	7	1	0	0	0	0	92	0	0	0
7	2	1	0	0	0	0	0	90	0	7
8	0	1	2	3	0	3	2	0	89	0
9	2	0	0	1	16	1	0	5	0	75

28x28x2 Prewitt filters

Actual digit \ *Predicted digit*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	89	1	3	0	0	3	4	0	0	0
1	0	100	0	0	0	0	0	0	0	0
2	6	3	72	1	0	2	1	1	14	0
3	1	1	7	77	0	5	0	0	7	2
4	0	4	2	0	87	1	0	0	0	6
5	4	3	0	2	1	78	4	0	7	1
6	6	1	0	0	0	1	92	0	0	0
7	0	1	0	0	3	0	0	92	0	4
8	0	4	1	5	1	3	0	0	84	2
9	0	0	0	0	29	1	0	7	0	63

28x28 pixels (10 neurons reset)

Actual digit \ *Predicted digit*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	98	1	0	0	0	0	1	0	0	0
1	1	98	0	0	1	0	0	0	0	0
2	10	1	73	1	0	0	1	1	13	0
3	1	0	5	85	0	2	0	0	6	1
4	1	2	2	0	79	0	2	1	2	11
5	8	0	0	2	1	84	2	0	2	1
6	5	0	0	1	0	0	94	0	0	0
7	1	1	0	0	0	0	1	89	1	7
8	0	2	2	4	0	5	0	0	87	0
9	0	0	1	0	19	2	0	2	1	75

28x28x2 Prewitt filters (10 neurons reset)

Actual digit \ *Predicted digit*	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	88	1	3	0	0	4	4	0	0	0
1	0	100	0	0	0	0	0	0	0	0
2	6	5	72	1	0	2	0	1	13	0
3	1	0	5	81	0	3	0	0	8	2
4	0	2	4	0	88	1	0	1	0	4
5	4	2	0	2	1	72	5	1	9	4
6	6	1	0	0	0	2	91	0	0	0
7	0	1	0	0	3	0	0	91	0	5
8	0	4	1	3	1	2	0	0	88	1
9	0	0	0	0	27	1	0	8	0	64

Figure 7. Confusion matrices. Left column: 28x28 pixels. Right column: 28x28x2 Prewitt filters. Top row: without resetting neurons. Bottom row: resetting the 10 least confident neurons.

Table 3. Digit/non-digit classification accuracy of 4 networks, with and without resetting the 10 least confident neurons.

SNN	**Accuracy**		
	Overall	**No spike**	**Non-digit**
28x28	0.003	0.003	
28x28 (reset)	0.969	0.001	0.968
28x28x2	0.927	0.927	
28x28x2 (reset)	0.961	0.805	0.156
14x14	0.001	0.001	
14x14 (reset)	0.972	0.000	0.972
14x14x2	0.915	0.915	
14x14x2 (reset)	0.960	0.714	0.246

as digit 0. By contrast, the Prewitt-filter-based networks do not spike for about 92% of the CIFAR-100 images, making them reliable detectors of non-digits. We note that when the networks are trained for only 1 epoch, the error detection accuracy of the Prewitt-filter-based networks deteriorates to between 40 and 60%. Thus, error detection accuracy benefits more from longer training than digit classification accuracy.

Resetting the 10 least confident neurons dramatically improves the non-digit detection for pixel-based networks, resulting in a non-digit detection accuracy of about 97%. Resetting neurons also marginally improves non-digit detection for Prewitt-filter-based networks, raising the overall non-digit detection accuracy from 92% to about 96%.

We also note that discriminating between digits and non-digits based on prediction confidence thresholds will not yield a reliable error dectector for the pixel-based SNNs. For example, for the 28x28 pixel network the 75th percentile of the prediction confidence when classifying 1000 non-digits is 0.59. Thus, using a confidence of 0.59 or lower as the threshold for detecting non-digits would result in misclassifying 25% of non-digits as digits. And yet, the same threshold would also result in approximately 25% of digits being misclassified as non-digits since the 25th percentile of the prediction confidence when classifying 1000 digits is 0.60 (Table 2). This shows that there is significant overlap between the distributions of confidence levels for digits and non-digits, and no threshold will be able to separate the distributions without large errors.

5. CONCLUSIONS

The aim of this work was to investigate biologically plausible ways to improve the training speed, classification accuracy and robustness of a digit classifier based on a known spiking neural network architecture.[1] We found that processing simple oriented features rather than pixels had little impact on training speed (though it does have an impact on the variability of training speed, Section 4.1) and accuracy (Section 4.2) but dramatically improved robustness (Section 4.3). We also found that "resetting" some neurons post-training improved robustness without compromising accuracy (Section 4.3). Processing-oriented features (vertical and horizontal edges) increase the model size (by 4x) over a pixel-based model, though the increase can be offset by sampling input images at a lower resolution (without significant loss of accuracy for MNIST digit classification) as it is in the case of our study.

Biological vision involves a hierarchy of processing stages, from detecting local features (oriented bars and edges), to connecting these features to form contours and textures, and eventually shapes and surfaces. This rich hierarchy of features is used to assign meaning to images. Yet, the model by Diehl and Cook (2015) omits the entire hierarchy of feature detectors, assigning meaning directly to patterns of input pixels.[1] Our work contrasts the pixel-based models with models based on just two oriented features. Restricting to two orientations does cause a small loss in accuracy, however, oriented features turn out to naturally suppress responses to error inputs, making the networks more selective for inputs that are similar to the training data. This increased selectivity may be an advantage in applications where error inputs are common and should just be ignored.

Biological vision is robust in the sense that it will rarely misclassify objects. Biologically, this is achieved with a number of different means, e.g. by using context information. Our work investigates two possible ways of responding to error inputs, both avoiding the use of context information. (1) Error suppression rests on the increased selectivity of neurons trained on oriented features instead of pixels. This is a biologically plausible mechanism. (2) Error detection uses dedicated "untrained" neurons that respond to "random" inputs. It is an open question whether this is a biologically plausible mechanism. While actively "resetting" neurons isn't biologically plausible, we note that the reset neurons resemble neurons that aren't trained yet. Thanks to STDP, neurons are trained selectively rather than at a uniform rate. If the neuron population is big enough (i.e., significantly bigger than the 100 output neurons in our experiments, as is the case in biological vision), it is plausible that some neurons will never be trained and will therefore be able to perform the same error detection function as neurons that were reset.

Our model still falls short of biological vision in restricting to two orientations and directly assigning meaning to patterns of oriented features instead of assembling complex shapes (e.g. angles, segments of curves) from oriented features. Future work will investigate whether increasing the number of orientations and adding layers to represent angles and curve segments can increase the accuracy and robustness of digit classification.

ACKNOWLEDGEMENTS

This research was supported by DASA, grant #ACC6036789, awarded to D.B., E.G., P.M. and K.A.

REFERENCES

[1] Diehl, P. U. and Cook, M., "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers Comput. Neurosci.* **9**, 99 (2015).
[2] Hubel, D. H. and Wiesel, T. N., "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology* **195**(1), 215–243 (1968).
[3] Kapadia, M. K., Westheimer, G., and Gilbert, C. D., "Spatial distribution of contextual interactions in primary visual cortex and in visual perception," *J Neurophysiol* **84**(4), 2048–62 (2000).
[4] Carandini, M. and Heeger, D. J., "Summation and division by neurons in primate visual cortex," *Science* **264**(5163), 1333–6 (1994).
[5] Gheorghiu, E. and Kingdom, F. A., "Multiplication in curvature processing," *J Vision* **9**(2), 20–23 (2009).
[6] Anzai, A., Peng, X., and Van Essen, D. C., "Neurons in monkey visual area V2 encode combinations of orientations," *Nat Neurosci* **10**(10), 1313–21 (2007).
[7] Dobbins, A., Zucker, S. W., and Cynader, M. S., "Endstopped neurons in the visual cortex as a substrate for calculating curvature," *Nature* **329**(6138), 438–41 (1987).
[8] Hegde, J. and Van Essen, D. C., "Selectivity for complex shapes in primate visual area V2," *J Neurosci* **20**(5), RC61 (2000).
[9] Pasupathy, A. and Connor, C. E., "Shape representation in area V4: Position specific tuning for boundary conformation," *Journal of Neurophysiology* **86**, 2505–2519 (2001).
[10] Pasupathy, A. and Connor, C. E., "Population coding of shape in area V4," *Nature Neuroscience* **5**(12), 1332–8 (2002).
[11] Felleman, D. J. and Van Essen, D. C., "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex* **1**(1), 1–47 (1991).
[12] Merigan, W. H., "Basic visual capacities and shape discrimination after lesions of extrastriate area V4 in macaques," *Visual Neuroscience* **13**(1), 51–60 (1996).
[13] Connor, C. E., Brincat, S. L., and Pasupathy, A., "Transformation of shape information in the ventral pathway," *Curr Opin Neurobiol* **17**(2), 140–7 (2007).
[14] Brincat, S. L. and Connor, C. E., "Underlying principles of visual shape selectivity in posterior inferotemporal cortex," *Nat Neurosci* **7**(8), 880–6 (2004).
[15] Ito, M. et al., "Size and position invariance of neuronal responses in monkey inferotemporal cortex," *J Neurophysiol* **73**(1), 218–26 (1995).
[16] Tanaka, K., "Inferotemporal cortex and object vision," *Annu Rev Neurosci* **19**, 109–39 (1996).
[17] Gross, C. G., "Representation of visual stimuli in inferior temporal cortex," *Philos Trans R Soc Lond B Biol Sci* **335**(1273), 3–10 (1992).
[18] Fujita, I. et al., "Columns for visual features of objects in monkey inferotemporal cortex," *Nature* **360**(6402), 343–6 (1992).
[19] Graham, N. V., "Beyond multiple pattern analyzers modeled as linear filters (as classical V1 simple cells): useful additions of the last 25 years," *Vision Res* **51**(13), 1397–430 (2011).
[20] Landy, M. S., "Texture analysis and perception," in [*The New Visual Neurosciences*], Werner, J. S. and Chalupa, L. M., eds., ch. 45, 639–652, MIT Press, Cambridge, Mass. (2013).
[21] Malik, J. and Perona, P., "Preattentive texture discrimination with early vision mechanisms," *J Opt Soc Am A* **7**(5), 923–32 (1990).
[22] Thielscher, A. and Neumann, H., "Neural mechanisms of human texture processing: texture boundary detection and visual search," *Spat Vis* **18**(2), 227–57 (2005).
[23] Wilson, H. R., "Non-fourier cortical processes in texture, form, and motion perception," in [*Models of Cortical Circuitry*], Ulinski, P. S. and Jones, E. G., eds., **13**, 445–477, Plenum, New York (1999).
[24] Landy, M. S. and Bergen, J. R., "Texture segregation and orientation gradient," *Vision Res* **31**(4), 679–91 (1991).
[25] HAMIDA, S., CHERRADI, B., RAIHANI, A., and OUAJJI, H., "Performance evaluation of machine learning algorithms in handwritten digits recognition," in [*2019 1st International Conference on Smart Systems and Data Science (ICSSD)*], 1–6 (2019).
[26] Datta, G. and Beerel, P. A., "Can deep neural networks be converted to ultra low-latency spiking neural networks?," in [*2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*], 718–723 (2022).
[27] Yi, Z., Lian, J., Liu, Q., Zhu, H., Liang, D., and Liu, J., "Learning rules in spiking neural networks: A survey," *Neurocomputing* **531**, 163–179 (2023).
[28] Izhikevich, E., "Which model to use for cortical spiking neurons?," *IEEE Transactions on Neural Networks* **15**(5), 1063–1070 (2004).

[29] Vaila, R., Chiasson, J., and Saxena, V., "Deep convolutional spiking neural networks for image classification." https://arxiv.org/abs/1903.12272 (2019).
[30] He, W., Wu, Y., Deng, L., Li, G., Wang, H., Tian, Y., Ding, W., Wang, W., and Xie, Y., "Comparing SNNs and RNNs on neuromorphic vision datasets: Similarities and differences," *Neural Networks* **132**, 108–120 (2020).
[31] Li, Z. and Meng, L., "Deep spiking neural networks for image classification," *International Journal of Human Factors Modelling and Simulation* **8**(1), 21–35 (2023).
[32] Stimberg, M., Brette, R., and Goodman, D. F., "Brian 2, an intuitive and efficient neural simulator," *eLife* **8**, e47314 (Aug. 2019).
[33] Prewitt, J., "Object enhancement and extraction," in [*Picture Processing and Psychopictorics*], Lipkin, B. and Rosenfeld, A., eds., 75–150, Academic Press (1970).

APPENDIX A. MORE RESULTS

Here we report on repeats of the experiments of sections 4.1 and 4.2 when training on the full MNIST dataset, instead of training on just 10% of the dataset.

A.1 Speed of training convergence

We first investigate how the networks learn when trained on the full MNIST training dataset. We trained each network for 2 epochs on all 60,000 MNIST training images. At intervals, the networks were frozen and and their accuracy evaluated by classifying all 10,000 MNIST test images. Up to 10,000 training images, the networks were evaluated every 1000 images. For the remainder of epoch 1, they were evaluated every 5000 images. For epoch 2, the networks were evaluated every 10000 images.

Figure 8. Digit classification accuracy of 4 networks as training progresses (fulll MNIST training dataset).

Figure 8 shows the classification accuracy over the training time. While the 4 networks differ on the behaviour at the start of training (as was observed in section 4.1), they converge from about 10,000 training images, and appear to reach peak accuracy before the end of epoch 1, at around 40,000 images.[‡] From that point on, accuracy remains relatively stable, with an observable dip twards the end of each epoch for all networks; only the 28x28x2 Prewitt filter network shows more pronounced drops in accuracy at the end of the epochs. The table below summarises the accuracy of each network as the geometric mean over the accuracy of epoch 2. The lower mean

[‡]This is consistent with the findings of Diehl and Cook (2015) which reports an average accuracy of 0.829 for a 28x28 pixel network of 100 neurons trained on 40,000 MNIST images.[1]

Table 4. Accuracy of classifying MNIST digits (full MNIST training and testing datasets).

SNN	Accuracy	Confidence		
		Q1	Q2	Q3
28x28	0.827	0.49	0.66	0.86
28x28 (reset)	0.829	0.52	0.68	0.85
28x28x2	0.780	0.40	0.55	0.83
28x28x2 (reset)	0.800	0.45	0.58	0.82
14x14	0.845	0.54	0.73	0.88
14x14 (reset)	0.835	0.55	0.73	0.86
14x14x2	0.842	0.52	0.73	0.88
14x14x2 (reset)	0.841	0.55	0.74	0.86

accuracy of the 28x28x2 Prewitt filter network is mainly due its pronounced drop in accuracy at the start and end of the epoch; the mean accuracy of the other networks is comparable.

	28x28	28x28x2	14x14	14x14x2
Mean accuracy	0.847	0.817	0.858	0.849

A.2 Impact of resetting neurons on accuracy and confidence

Next, we take the above networks, trained over the full two epochs, and "untrain" the 10 least confident neurons of each of the four networks by resetting their input weights to their initial random state. We compare digit classification accuracy on the full set of 10,000 MNIST test images before and after resetting neurons.

Table 4 shows the classification accuracy without and with resetting neurons, together with the quartiles for the prediction confidence. The observations are similar to section 4.2. That is, resetting the 10 least confident neurons has only a small effect on the digit classification accuracy. The largest effect is observed for the high-resolution Prewitt filter network whose accuracy improves by 2 percentage points when resetting the 10 least confident neurons; this may be due to the fact that resetting those 10 neurons undoes some of the damage that was caused by overfitting at the end of the epoch.

The observations for prediction confidence are similar to before: Resetting the least confident neurons lifts the lower quartile, and in some cases the median, due to removing the low-confidence contributions of the reset neurons. Resetting also slightly lowers the upper confidence quartile.