Competitive Learning with Spiking Nets and Spike Timing Dependent Plasticity

Christian Huyck[1] and Orume Erekpaine[1]

Middlesex University, London NW4 4BT UK
c.huyck@mdx.ac.uk
http://www.cwa.mdx.ac.uk/chris/chrisroot.html

Abstract. This paper explores machine learning using biologically plausible neurons and learning rules. Two systems are developed. The first, for student performance categorisation, uses a two layer system and explores data encoding mechanisms. The second, for digit categorisation, explores competitive behaviour between categorisation neurons using a three layer system with an inhibitory layer. Both are successful. The competitive mechanism from the second system is more plausible biologically, and, by using one neuron per input feature, uses fewer neurons.

Keywords: Spiking Neurons · Spike Timing Dependent Plasticity · Categorisation · MNIST

1 Introduction

The authors have proposed, and still believe, that the best way to develop a full-fledged, Turing test passing, general AI is to follow the human model (Huyck, 2017). This means developing embodied agents that persist for a significant amount of time (e.g. years), are good cognitive models of most of the things that humans do, and are based on simulated neurons that are relatively close approximations to biology. Progress can be made by using commonly used spiking neurons, and commonly used Hebbian learning rules. These models have strong support from biological evidence, and many of their limitations are understood and are being actively explored (Taherkhani et al, 2020).

While there is interest in passing the Turing test, there is appreciably more interest in machine learning. Deep nets, such as BERT (Devlin et al, 2018) and Alpha Go (Silver et al, 2017), are widely used in modern industrial tasks. These make use of neuron like nodes, and synapse like connections; they take advantage of biologically implausible gradient descent methods, and vast data sets to learn the connection weights. These "neural nets" have impressive results and it is not surprising that they spark a great deal of interest.

They are not, however, particularly closely connected to biology. One way to bridge the gap is to use more biologically plausible systems to solve machine learning tasks. The systems described in this paper use simple spiking neurons, based on point models. They learn by spike timing dependent plasticity (STDP), a model with biological support (see section 2.1). It uses the firing times of the pre and post-synaptic neurons to select the weight change with the weight increasing if the pre-synaptic neuron fires first, and decreasing if the post-synaptic neuron fires first.

This paper describes two machine learning systems, one for student performance categorisation (section 4), and one for digit categorisation (section 5). The student performance system is based on the authors' earlier work (Huyck, 2020) and uses a two layer topology. The digit categorisation system makes use of a three layer topology inspired by Diehl and Cook (2015) (see section 2.2). Both make use of standard neural models, middleware and a neuron simulator (see section 2.1), so that the systems can be readily used and modified[1] with an explanation of how to run the simulations to reproduce the data reported below.

2 Literature Review

The work reported in this paper is the third in a series of papers using biologically motivated simulated neurons and learning rules. The earlier papers (Huyck, 2020; Huyck and Samey, 2021) used a feed forward topology with input neurons connected to category neurons.

Like the systems introduced in the rest of the paper, these earlier papers used standard neural models, standard learning models, a widely used neural simulator, and commonly used middleware. These are described in section 2.1.

Those earlier papers make use of a two layer architecture with an input layer connected to an output layer. The neurons in the output layer act as the categoriser, and the connections from input to output are plastic during training. During training, an input is presented by causing the appropriate input neurons to fire, followed by causing the appropriate categorisation neuron to fire. This leads to a mechanism where the synaptic weight reflects the co-occurrence value between the feature value neuron and the category neuron. The system from section 4 also makes use of this mechanism.

Section 5 uses a different three layer topology that allows the categorisation neurons to compete with each other. This is a modification of the work of Diehl and Cook (2015), described more fully in section 2.2.

2.1 Standard Models and Commonly Used Systems

The simulations in the earlier papers (Huyck, 2020; Huyck and Samey, 2021) and those described below make use of commonly used computational neuroscience platforms. In particular, one commonly used mechanism is to use PyNN (Davison et al, 2007) as middleware to specify the neural topology, synaptic modification rules, neural stimulation, and recording. This acts as middleware between the developer and existing neural simulators; there are many simulators, and the simulations below use NEST (Gewaltig and Diesmann, 2007).

[1] The code can be found on http://www.cwa.mdx.ac.uk/spikeLearn/spikeLearn.html.

In this paper, the leaky integrate and fire neural models with fixed threshold and exponentially decaying conductance are from Brette and Gerstner (2005) (*IF_cond_exp* in PyNN). The model is based on equations 1 and 2.

$$C\frac{dV}{dt} = f(V) - w + I \tag{1}$$

$$f(V) = -g_L(V - E_L) + g_L \varDelta_T exp(\frac{V - V_T}{\varDelta_T}) \tag{2}$$

V is the variable that represents the voltage of the neuron, and T is time. C is the membrane capacitance constant, and I is the input current. w is an adaptation variable that is 0 for these neurons. g_L is the leak conductance constant and E_L is the resting potential constant. $\varDelta_T$ is the slope factor constant and V_T is the spike threshold constant.

In one set of simulations, neurons with adaptation are used. In this case, every time a neuron fires, its adaptation variable w is increased by a constant b. This then reverts to 0 depending on another constant a, and time as shown in equation 3. τ_w is the adaptation time constant. This in effect makes it more difficult for neurons to fire frequently.

$$\tau_w \frac{dw}{dt} = a(V - E_L) - w \tag{3}$$

Equations 1 and 2 determine the change in V (the voltage variable), and equation 3 manages the change in w (the adaptation variable). These variables are also changed when $V > V_T$ and the neuron spikes; V is reset to E_L, and w is increased by a constant b for spike triggered adaptation. The simulations described below in this paper use the default constants described by (Brette and Gerstner, 2005).

In the brain, most if not all learning is Hebbian (Hebb, 1949). If the presynaptic neuron tends to cause the post-synaptic neuron to fire, the weight will tend to increase. There are many variations of this rule, but a great deal of biological evidence supports STDP (Bi and Poo, 1998). Bi and Poo (1998) have perhaps the first published example that shows the performance of the changing efficiency of biological synapses. Song et al (2000) have developed an idealised curve that fits the biological data.

The simulations below use the standard spike pair STDP rule described by equation 4. The presynaptic neuron fires at t_r and the post-synaptic neuron fires at t_o. If the presynaptic neuron fires first, the weight is increased (modulated by a constant c_+) otherwise it decreases (modulated by the constant c_-).

$$\varDelta_w = \begin{cases} c_+ * e^{t_r - t_o} & t_r <= t_o \\ c_- * e^{t_o - t_r} & t_o > t_r \end{cases} \tag{4}$$

2.2 Unsupervised learning using STDP

Perhaps the best working example of spiking nets learning using Hebbian rules is by Diehl and Cook (2015). This has several main differences from the authors' earlier work in this area (Huyck, 2020; Huyck and Samey, 2021). The first is that there are not single neurons for each category but instead a group of neurons that are not explicitly associated with any category. Unlike Huyck and Samey (2021), these category neurons are never explicitly fired during training, but are only activated from the input. Additionally, there is a set of inhibitory neurons that take input from the category neurons and in turn inhibits them. This enables the layer of category neurons to compete. The only plastic synapses are from the input to the category neurons. When the system is learning properly, the neurons compete for the synaptic strength, and each fires only when a particular type of input is presented.

This is unsupervised behaviour, and to this point the category labels have not been presented. Once this training is complete, the category neurons are assigned labels based on the categories of the inputs to which they respond.

This behaviour resembles that of a self organising map (SOM) (Kohonen, 1997). The nodes of the SOM are like a category neuron. The SOM nodes move to a place that responds to particular inputs, and moves away from other nodes. When the category neurons are learning properly, they also respond to particular inputs and not others. If one wants to categorise with a SOM network, each of the nodes can be assigned a particular category. Novel input can then be categorised by the category of the node that responds.

The second difference is input. In Huyck and Samey (2021), each value for each feature had its own neuron. In Diehl and Cook (2015), each feature has a single neuron, and higher values lead to more activation, which leads to earlier firing or earlier firing along with more spikes. This allows fewer neurons to be used.

The third difference is an enforced balancing of firing for these categorisation neurons. Category neurons have to fire roughly the same amount over several data items. Diehl and Cook (2015) enforce this by a dynamic firing threshold that responds to how often it has recently fired. This relates to adaptation. See sections 3.2 and 6 for more on balanced firing.

3 Methods

Biological neuron simulations run for a period of simulated time with the neuron behaving throughout the period. When a neuron fires, activation spreads from it to other neurons that have synapses from it.

The systems can learn via a Hebbian learning rule, and the simulations in this paper use one. Synaptic weights change via STDP, increasing when pre-synaptic neurons fire before post-synaptic neurons, and decreasing when pre-synaptic neurons fire after the post-synaptic neurons.

Both of the systems introduced in this paper work in a layered fashion, with input neurons in the first layer and category neurons in the second. There are no synaptic connections within layers, only between layers.

Neurons used in these simulations are leaky integrate and fire neurons. All use the default parameters. In both systems, training is performed by a system with plastic synapses. Data items are presented in sequence. In the Digit Categorisation task, one system variant uses leaky integrate and fire neurons with adaptation for the categorisation neurons. This uses all the default parameters except the adaptation increase rate b (see equation 3).

3.1 Student Performance Categorisation

The student performance system categorises based on data from the UCI machine Learning Repository (Cortez and Silva, 2014). The dataset includes performances for mathematics and Portuguese separately but only the performance for mathematics is used. The dataset contained 33 columns and 395 rows with 32 features and a numeric target. The data was originally intended for a regression task, but classification labels were derived from the initial target column for the categorisation task used in this paper.

Input to the student performance system is performed by a spike source array. A spike source is specified (based on the input data) and a particular time. The static synaptic weight from the source to the neuron is 0.1. This causes the neuron to spike exactly once. For testing there are no spike sources for the output layer.

Several input data encoding mechanisms are used. These and their translate to into input neuron spikes are explained in section 4.

There are four output categories. The initial data has 21 categories, but these have been binned: 0-5 *Fail*; 6-10 *Pass*; 11-15 *Credit*; and 16-20 *Distinction.*

The input neurons are well connected to the category neurons. Each learns using the additive form of STDP.

3.2 Digit Categorisation

The digit categorisation experiments are based on a version of the widely used MNIST digit categorisation benchmark (Bache and Lichman, 2013). In this version, the digits are represented by a vector of 64 inputs (an 8x8 box) with values between 0 and 16, a transformation of the initial 28x28 bitmap of the digit.

The input layer consists of 64 neurons, one for each vector item. Input instances are given 100 ms of simulated time with neurons associated with non-zero items given clamped input (DC current or DCSource in PyNN) from 20 to 80 ms of that 100 ms. Simulations were run with a .1 ms time step. Higher value inputs are given a larger amount of current. The firing times of inputs are shown in table 1.

Digit Topology The basic topology of the digit recognition system shown in figure 1 is, like Diehl and Cook (2015), three layers. All versions of the system have 64 inputs. The input layer is well connected to the categorisation layer (all to all); when plastic, the synapses are STDP synapses using the multiplicative form of the rule. During the STDP phases of training they are plastic, and during testing they are static.

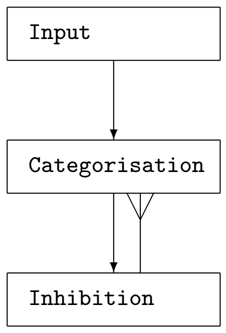

Fig. 1: The three layer topology has an input, a categorisation, and an inhibition layer.

Input:	Spike 1	Spike 2	Spike 3	Spike 4	Spike 5	Spike 6	Spike 7
1:	75.5						
2:	55.9						
3:	47.8	75.7					
4:	43.0	66.1					
5:	39.7	59.5	79.3				
6:	37.3	54.7	72.1				
7:	35.4	50.9	66.4				
8:	33.9	47.9	61.9	75.9			
9:	32.7	45.5	58.3	71.1			
10:	31.7	43.5	55.3	67.1	78.9		
11:	30.8	41.7	52.6	63.5	74.4		
12:	30.1	40.3	50.5	60.7	70.9		
13:	29.5	39.1	48.7	58.3	67.9	77.5	
14:	28.9	37.9	46.9	55.9	64.9	73.9	
15:	28.4	36.9	45.4	53.9	62.4	70.9	79.4
16:	27.9	35.9	43.9	51.9	59.9	67.9	75.9

Table 1: Table of input spike times in ms with DC clamp from 20 ms to 80 ms.

Category neurons are well connected to the inhibitory neurons, and inhibitory neurons are also well connected back to the category neurons. These connections are static, throughout training and testing. The inhibitory layer is activated when neurons in the category layer fire, and if it receives sufficient activation, its neurons fire in turn reducing activation, and, perhaps, firing in the categorisation neurons.

Test Method After training, the synaptic matrix from input to category neurons is stored; this weight matrix contains all of the parameters that have been learned from training. For testing, the trained synaptic weights are read back in to the now static synapses. Once trained, each category neuron is labelled with a category. The system, with the trained but now static synapses, is presented with some training data, and the firing behaviour of the category neurons recorded. For each instance of the data, a winning category neuron is selected; this is the neuron that fired first after the beginning of the instance input. (The neurons can be referenced by number, and in the event of a tie, the lowest numbered neuron won.)

At the end of this run, each neuron is labelled as the category it won most frequently. In the event of a tie between categories, the neuron was given the label of the first of which it won most. On the contrary, if the neuron never won, it was given (really quite arbitrarily) the category 8.

During testing, the static version of the input to category synapses is read in. An instance of the test data is presented, and the neuron that fired first (with lower numbered neuron used to resolve ties) is the winner. The category label associated with the winning neuron is used to predict the answer.

Balanced Firing One of the problems with early experiments on this system was that particular category neurons would, in essence, win the overall competition. That is, each presentation would lead to a few neurons or even one neuron firing first on each instance. In the extreme event that only one category neuron always won, the resulting system would always predict the same category. When only a few neurons won, the results were also extremely poor, near chance, which is 10%.

What is needed is a mechanism to force all of the neurons to fire about the same amount of time. This enables them to compete for particular portions of the input space.

Diehl and Cook (2015) used a dynamic threshold based on firing to force neurons to fire at about the same rate. Using their model would require a non-standard neural model for NEST. While it is possible to write user defined neural models for NEST, two other mechanisms are used. The first is a simple weight adjustment mechanism, and the second is to use standard neural models with adaptation.

The first balancing mechanism explicitly changes weights to balance firing. After an STDP run, the synaptic matrix is stored and then the system rerun with static synapses. The total number of spikes for each category neuron is recorded. A target upper and lower bound for the number of spikes is selected. If the category neuron is below the lower bound, all of its synapses from input are increased proportionally to how far the number of spikes are below the target. If the number of spikes is above the upper bound, the incoming synaptic weights are reduced proportionally to how far the number of spikes are above that bound. This change is all done in python code written in the PyNN script.

The second mechanism was to change the model of the categorisation neurons to one with adaptation (Brette and Gerstner, 2005) (*IF_cond_exp_isfa_ista* in PyNN). The neural model is a leaky integrate and fire model, but when the neuron fires, an extra variable is increased. This slowly reverts to 0, but while it is above 0, increases the firing threshold.

Digit Training and Test Method The training mechanism is to initiate the plastic synaptic matrix from input to category neurons using weights with some random variance. The system is then presented with data learning by STDP. The initial weights need to be sufficiently large to enable the input neurons to cause the category neurons to fire.

In the case of the first, compensatory mechanism, balancing is applied until the category neurons fire within the desired range on the particular training data.

The neurons are then labelled. The system is then presented with unseen test data. As the data set is divided into two, there is a two-fold cross validation.

4 The Student Performance Categorisation System

In this section, a system that categorizes student performance based on data from the UCI machine Learning Repository (Cortez and Silva, 2014) is described. A spiking neural net learning via STDP is used. The main question to be answered in this implementation is how well a student grade categorizer could be learned by this type of network.

A secondary question is how the system performs with different input data to input neuron transformation techniques. Two preprocessing techniques (one hot encoding and integer encoding) are used for categorical inputs; up-sampling and down-sampling are used for target class balancing; and min-max scaling and no scaling are used for integer value features. This led to eight data encodings.

For one hot encoding, two neurons are allocated for each category, an on neuron and an off neuron. During presentation the on neuron for the feature value is turned on, and the off neurons for the other values. For integer encoding, there is just one neuron per feature value.

There are four categories, class 4 has 40 entries with the others having more including class 3, which has 169 entries. Down-sampling balances the number in each class by making them all have 40. Up-sampling generated new artificial data so that all classes have 169 items.

The integer encoding replaces them with the nearest integer. Min-Max scaling is described by equation 5. The actual input value, X_{SC}, was determined by using the feature value X, the smallest value of that feature in the dataset, X_{min}, and the largest value, X_{max}; the result was an input feature scaled between 0 and 1.

$$X_{SC} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{5}$$

The neurons allocated to each feature (neurons per feature) vary depending on the encoding mechanism used. There is also a feature breadth of 3 for integer values; each integer input value has its neuron and its adjoining neurons stimulated.

The data is split 70-30% training and test respectively in every case.

The system is a 2-layer feed-forward neural network consisting of conductance based leaky integrate and fire neurons with fixed thresholds. The code is written in the PyNN python package and is simulated in NEST (Davison et al, 2007; Fardet et al, 2020)

4.1 Student Performance Spiking Net Model

The simulation time step is 1 ms. as are the minimum and maximum synaptic delay. The time between training data items is 30 ms. All training data items are presented four times (four epochs). Other intervals and epochs were explored but led to long training run times. So, these values are used for the all simulations described in the remainder of this section.

Training items are presented in time sequence 30 ms apart. At the beginning of training item presentation, a spike is sent to the input neurons, that is followed 3 ms later by a spike to the correct category neuron. The same approach is taken during testing, but there is no external input to the category neurons; the synaptic weights from the firing input neurons cause the category neurons to fire. The first to fire is the one that is used to predict the category.

The size of the input layer varied depending on the encoding mechanism; one hot encoding generated 9 extra columns in addition to the initial 32 and integer encoding did not. So, the total population of the input layer is neurons per feature multiplied by number of input data columns. For the output layer there were only 4 neurons, 1 neuron per output category.

The input and output layers are fully connected using plastic synapses that are governed by a biphasic STDP rule for long-term potentiation (LTP) and depression (LTD). This is a variant of the widely used Hebbian learning mechanism. The parameters that influenced the learning rule and their initial values are shown in table 2.

Parameter Name:	Value	Description
τ_+	12.0ms	Time constant required for LTP
τ_-	12.0ms	Time constant required for LTD
A_+	0.003	Synaptic Weight increase applied when LTP occurs
A_-	0.014	Synaptic Weight reduction applied when LTD occurs
w_{min}	0.0	Minimum synaptic weight possible
w_{max}	0.03	Maximum synaptic weight possible
weight	0.0	Initial synaptic weight at start of the simulation

Table 2: Parameter values for student classification system.

When the output neuron has produced spikes, these spikes are counted and separated into different arrays by category. Then a maximum argument rule is used to pick the winning category from each row. So the category of any output layer neuron with the most responses for a row is the predicted category for that row. It is important to know that for this rule if there is a tie for most responses the first serially occurring neuron of the tied neurons is chosen as the winner.

4.2 Results

The encoding techniques used resulted in 8 datasets in total, which were up-sampled scaled and unscaled, down-sampled scaled and unscaled for integer encoding and one-hot encoding. During this phase of testing the initial values for the STDP parameters from table 3 are used. $\tau_+ = \tau_- = 16.0$ are also shown. The best results are 72.4% for one hot encoded, up sampled, unscaled data.

Data Variation	$\tau_+ = \tau_- = 12.0$	$\tau_+ = \tau_- = 16.0$
Integer Encoded: Up-Sampled Unscaled	61.6%	70.9%
Integer Encoded: Up-Sampled Scaled	32.0%	23.6%
Integer Encoded: Down-Sampled Unscaled	50.0%	25.0%
Integer Encoded: Down-Sampled Scaled	29.2%	25.0%
One-Hot Encoded: Up-Sampled Unscaled	63.1%	**72.4**%
One-Hot Encoded: Up-Sampled Scaled	38.9%	23.6%
One-Hot Encoded: Down-Sampled Unscaled	62.5%	25.0%
One-Hot Encoded: Down-Sampled Scaled	22.9%	25.0%

Table 3: Results of data encoding experiments on the Student Performance Data.

The results of scaled data are low across the result set. The suspected reason for this is that in the implementation of input to neuron mapping, scaling resulted in floating point values for the values being scaled, which are rounded during the mapping, which resulted in a loss of information and caused the system to perform poorly.

For up-sampled data, the main concern is the use of 'fake' data, as new examples are generated to balance the classes. Down-sampling presented the issue of having far fewer examples to train and test the system.

Parameter exploration was performed on the learning window, τ_+ and τ_-, within ranges 10-17ms in steps of 1 and both equal. The best overall results are reported in the final column of table 3 with τ_+ and τ_- at 16.0ms.

For performance comparison with other networks, a Multi-Layer Perceptron (MLP) with a 3-layer feed forward architecture, logistic activation function and a stochastic gradient solver for error correction learning was implemented using the Scikit learn MLPClassifier that was trained and tested on the same data. The network took 1000 epochs to reach an accuracy of 76.0% compared to the spiking net that took 4 epochs to reach 72.4% accuracy.

Cortez and Silva (2014) evaluated systems on this data. A variety of models were implemented, but the one most relevant for comparison to the spiking net implemented in this paper is the 3-layer MLP. The results of that system are 49.8%. It is important to note that while the spiking net in this paper performs a four category classification task, the system in Cortez and Silva (2014) performs a five category classification.

5 The Digit Categorisation System

The commonly used MNIST task involves 50000 training items, each with a 28x28 grid of inputs from 0 to 256. Each item is a digitised scan of hand written digit, with the associated correct category. This is the work that Diehl and Cook (2015) used getting results of about 95%, on the 10000 item test set. The experiments described here work on a smaller, less widely used version of the task with 5620 items with an 8x8 grid of inputs from 0 to 16; this data set is from the University of California at Irvine (Bache and Lichman, 2013); and each 64D vector is derived from the orginal 28x28 picture by translating 4x4 squares depending on how many of them have any inputs on. So, if all the inputs are on (say, 12 at 256, and 4 at 18) then the number is 16. If only one is on (say the top left is 128) then it is 1. The authors have a great deal of experience with this data set having used it as the basis of a course work for several years for students in the final year of an undergraduate degree.

The data set is broken into two folds of 2810 items. A Euclidean distance categorisation metric gets 98.25%. Students have used a range of multi-layer perceptron learning with back propagation, and none have surpassed the Euclidean baseline, though one using a convolutional system recently was close. The standard mechanism that does better is a support vector machine, with some getting above 99%. The authors are unaware of any other spiking nets used to classify this data set.

5.1 Spiking Neuron Network Model

The two parts of the data were broken into 10 281 instance sets. The first 10 were used to train the first network for one pass. Each of the 10 STDP runs is followed by compensatory runs for the first balancing mechanism. The lower bound for neural firing is 100, and the upper bound is 281. The first of the ten training sets is used for labelling. Then the full test set is used. There are 100 category neurons and two inhibitory neurons. The second type of balancing uses adapative neuron, and these runs have just one pass on the 10 parts of the training data.

5.2 Results

It is unclear how a variant of the Diehl and Cook (2015) system would perform on this task. It is possible that the small training set size would limit its performance, but a result of 95%, like the result on the larger data set, seems reasonable. The result of the 100 category neuron test using the compensatory rule is 33.8%. Table 4 shows these results. The result using adaptive neurons with the adaptation rate $b = 0.1$ is 25.5%.

Algorithm:	Result
Euclidean Distance:	98.25%
SVM:	99.1%
Diehl and Cook:	˜95.0%
Compensatory Rule:	32.1%
Adaptive Neurons:	25.5%

Table 4: Small MNIST Categorisation Results: Results from this paper are shown in the Copmpensatory Rule and Adaptive Neurons rows. The result for Diehl and Cook is a guess.

Clearly, the performance on the digit categorisation spiking nets described in this paper are poor, but they are also clearly above chance, 10%. Parameter exploration has been minimal, and a theory for competition has not been developed. While it is unlikely that improved versions of these systems will surpass even Euclidean distance, further parameter exploration should enable their categorisation performance to improve significantly.

6 Discussion

The brain is a poorly understood organ, but it is clear that its 65 billion neurons (Churchland and Sejnowski, 1999) are used to, for instance, classify digits. All of the neurons are not critical to the task, but as it involves the primary visual cortex, billions of neurons are. With less than 200 neurons, the digit categorisation system described above is clearly not a complete model of the human neural network for solving the task.

Both systems have a neural model that is a reasonable, if simple, approximation to biological neurons. They use only one or two types of neurons and they are relatively simple models, but their parameters are based on biological evidence. Similarly, the STDP learning rule is a reasonable approximation of some of the learning done in the brain.

However, the topologies are not reasonable. The layering and well connectedness between layers does not occur in biology. Moreover, the learning mechanism in the student performance system involves forcing the output neurons to fire at a particular time. This is clearly not biologically plausible and to some extent means that that system is not using unsupervised learning. On the other hand, the digit system does not force the output neurons to fire, but uses their firing behaviour as part of the search, which is a truly unsupervised learning mechanism.

It is important that the category neurons fire at roughly the same rate over the training period because the synaptic weights only change when the pre and post-synaptic neurons both fire. If the post-synaptic neurons do not fire, the weights will remain unchanged. Diehl and Cook (2015) call this homeostasis, and it is important for a system (Hsu et al, 2007). The method first used in this paper is to balance modify synaptic weights to force balanced firing. Diehl and Cook (2015) use a dynamic threshold increasing the threshold when the category neuron fires and then allow it to decay back to base. It appears that this is a form of neural adaptation (Benda, 2021), which is similar to the second mechanism used in this paper.

Two basic learning mechanisms have been described in this paper; the two layer system learns co-occurrences. The three layer system is a competitive net providing another example of this mechanism for learning to categorise. The earlier paper (Huyck and Samey, 2021) modifies the two layer system by using the multiplicative form of STDP, so that the synaptic weights are exact co-occurrence values, and by forcing the category neurons that are not the answer to fire after the input, causing a synaptic weight reduction. A reasonable extension to this work is to try all three mechanisms on the same data set.

The most important future work is to develop a theory of competition with spiking neurons and STDP. It seems plausible that this can be directly tied to SOMs (Kohonen, 1997). Exploring the volume of input spiking, the inhibitory system, and the STDP parameters and mechanisms should support a well founded theory. Beyond this, more layers and reinforcement learning, may support improved categorisation performance. Full fledged recurrence and ongoing firing will make further advancements toward the actual biology.

7 Conclusion

These spiking systems can be used for categorisation. The digit system begins to explore how competition between the category neurons can be used to make each neuron “move” to recognise a particular part of the training states.

As machine learning systems, it is important that the mapping between input data and input neuron is understood. Several mechanisms for translating the input data are compared and contrasted in the student performance system. The digit categorisation system uses current to one neuron instead of a spike for the feature value. This means that the number of neurons needed as input is reduced, though the ramifications for learning are still unclear.

Following Diehl and Cook (2015), the digit system does not force the output neurons to fire, but uses their firing behaviour as part of the search. The inhibitory layer, the input driven behaviour and the learning behaviour force the category neurons to compete, and move to recognise particular areas of the input space. This is more biologically realistic than forcing the output neurons to fire at particular times.

This two layer system and three layer competitive system extend understanding of neurobioloigcally realistic learning systems. The spiking neuron STDP based systems are flawed as biological models, but may help build understanding of the actual biological mechanisms. Moreover, they help build understanding of machine learning with these systems.

Bibliography

Bache K, Lichman M (2013) UCI machine learning repository. URL http://archive.ics.uci.edu/ml

Benda J (2021) Neural adaptation. Current Biology 31(3):R110–R116.

Bi G, Poo M (1998) Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. Journal of neuroscience 18(24):10,464–10,472

Brette R, Gerstner W (2005) Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. J Neurophysiol 94:3637–3642

Churchland P, Sejnowski T (1999) The Computational Brain. MIT Press

Cortez P, Silva A (2014) UCI machine learning repository. URL https://archive.ics.uci.edu/ml/datasets/student+performance

Davison A, Yger P, Kremkow J, Perrinet L, Muller E (2007) PyNN: towards a universal neural simulator API in python. BMC neuroscience 8(S2):P2

Devlin J, Chang M, KLee, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint p arXiv:1810.04805.

Diehl P, Cook M (2015) Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Frontiers in computational neuroscience 9:99

Fardet T, Rajalekshmi D, Mitchell J, Eppler J, Spreizer S, Hahne J, Kitayama I, , Plesser H (2020) Nest 2.20.1. Computational and Systems Neuroscience

Gewaltig M, Diesmann M (2007) NEST (NEural Simulation Tool). Scholarpedia 2(4):1430

Hebb D (1949) The organization of behavior: A neuropsychological theory. New York: Wiley

Hsu D, Tan A, Hsu M, Beggs J (2007) A simple spontaneously active hebbian learning model: homeostasis of activity and connectivity, and consequences for learning and epileptogensis. Physical Review E 76:041,909

Huyck C (2017) The neural cognitive architecture. In: AAAI Fall Symposium on A A Standard Model of the Mind

Huyck C (2020) Learning categories with spiking nets and spike timing dependent plasticity. In: International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp 139–144

Huyck C, Samey C (2021) Extended category learning with spiking nets and spike timing dependent plasticity. In: International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp 33–43

Kohonen T (1997) Self-Organizing Maps. Springer

Silver D, Schrittwieser J, Simonyan K, et al, Hassabis D (2017) Mastering the game of go without human knowledge. Nature 550:354–59

Song S, Miller K, Abbott L (2000) Competitive hebbian learning through spike-timing-dependent synaptic plasticity. Nature neuroscience 3:9:919–926

Taherkhani A, Belatreche A, Li Y, Cosma G, Maguire L, McGinnity T (2020) A review of learning in biologically plausible spiking neural networks. Neural Networks 122:243–272