DARPA Report Task1 for Year 1 (Q1-Q4)

Task 1: Machine Learning with Spike-Timing-Dependent Plasticity (STDP)

1. Shortcomings of the deep learning approach to artificial intelligence

It has been established that deep learning is a promising approach to solving certain problems in artificial intelligence (AI) for which large amounts of data and computation are readily available [1]. On the other hand, tasks which require quick yet robust decisions in the face of little data, or a reasonable response despite the presence of an anomalous event are ill-suited for such an approach [2]. While it is true that deep learning has accomplished groundbreaking new baselines on many tasks in the domains of image processing [3], sequence learning [4], and others, these success stories have necessarily been accompanied by large, labeled datasets and increasingly powerful computers. Massive computation, and therefore energy expenditure, is required for the training of the increasingly complex models

To circumvent some of these shortcomings, we take inspiration from the energy efficient, massively parallel, and unsupervised brain. Mammalian brains are especially capable at complex reasoning, long-term planning, and other high-level functionalities, some that seem to be far out of the scope of deep learning systems. For this reason, taking advantage of the biological mechanisms employed by the brain (e.g., learning through plasticity, incorporating the relative timing of events, and massive parallelization) may enable the development of AI programs with similarly useful behaviors or properties.

2. Spiking neurons and spike-timing-dependent plasticity (STDP)

2.1 Spiking neuron and synapse models

The first step towards designing a more robust approach to machine learning with neural networks is to incorporate more powerful computational units. In a typical deep neural network (DNN), the basic computational operations include simple weighted sums, nonlinear activation functions, convolutions, pooling, and normalization layers, among others. A common feature of these computations is that they do not incorporate a sense of time, and unlike biological neurons, which demonstrate an all-or-nothing response via an action potential, they communicate by sending precise floating-point numbers downstream to the next layer of processing units.

A natural choice for a biologically inspired unit of computation is the leaky integrate-and-fire neuron, a simplified model of a neuron. In our models, we use a network of these units, some of which are excitatory (exciting the neurons to which it connects), and some of which are inhibitory (inhibiting the units to which it connects). The membrane voltage V is given by

$$\tau \frac{dV}{dt} = (E_{rest} - V) + g_e(E_{exc} - V) + g_i(E_{inh} - V)$$

Here, E_{rest} is the resting membrane potential, E_{exc} and E_{inh} are the equilibrium potentials of excitatory and inhibitory synapses, respectively, and g_e and g_i are the conductance of excitatory and inhibitory synapses, respectively. The time constant tau is chosen to be longer for excitatory

neurons than for inhibitory neurons, as observed in biology. When a neuron's membrane potential exceeds its membrane threshold v_{thresh}, it fires an action potential and resets back to v_{reset}. The neuron then undergoes a few-milliseconds refractory period, during which time it cannot spike. The values of the remaining variables of the neuron equations are also chosen to lie in a biologically plausible range, expect for that of the excitatory membrane potential. Increasing this time constant to from the experimentally observed 10-20ms to 100ms greatly increased the classification accuracy of the spiking neural network model introduced in [5], which we retain in the models we have developed.

Synapses are modeled by conductance changes; that is, a synapse increases its conductance at the moment a presynaptic action potential arrives by its synaptic weight w [6]. Otherwise, the conductance is decaying exponentially. The dynamics of the synaptic conductance (for an excitatory synapse) are given by

$$\tau_{g_e} \frac{dg_e}{dt} = -g_e$$

Again, the associated time constant lies in a biologically plausible range. If the presynaptic neuron is instead inhibitory, the same update equation is applied, albeit with a different associated time constant.

2.2 Spike-timing-dependent plasticity

In our network models, we used spike-timing-dependent plasticity (STDP) to modify the weights of the synapses between certain designated neurons. For the sake of computational efficiency, we use online STDP, in which each synapse keeps track of a value x_{pre} (the synaptic trace), a simplified model of its recent presynaptic spike history. Each time a presynaptic spike arrives at the synapse, x_{pre} is set to 1; otherwise, it decays exponentially towards zero. When a postsynaptic spike arrives at the synapse, the weight change Δw is calculated using a STPD update rule, the simplest of which we use is given by

$$\Delta w = \eta x_{pre}(w_{max} - w)^{\mu}$$

The term η denotes a learning rate, w_{max} the maximal allowed synaptic weight, and μ a parameter which governs the dependence of the weight update on the value of the previous weight. We also tested three other STDP rules in our models, where the first introduces exponential dependence on the weights prior to the update, the second uses both a presynaptic trace and a postsynaptic trace, where postsynaptic spikes trigger a decrease in the corresponding synaptic weight, and the fourth uses a combination of these last two mechanisms. Consult Figure 1 for a schematic of the online STDP scheme.

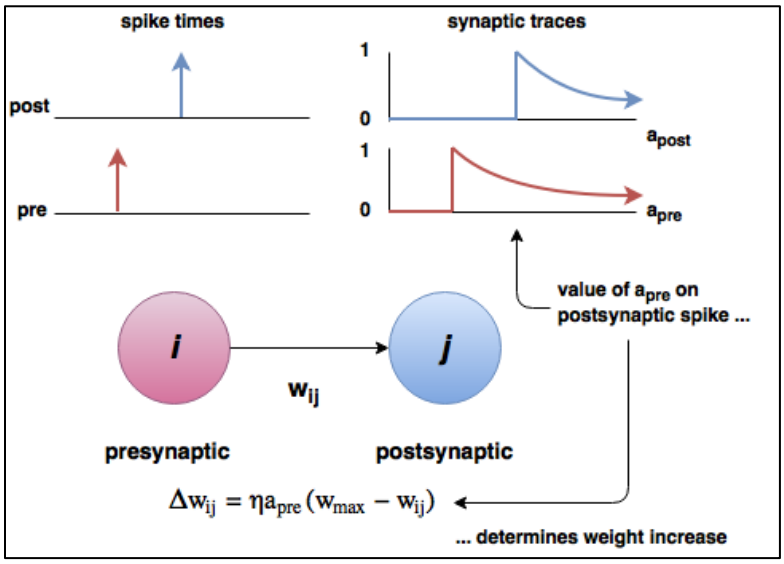


Figure 1: Illustration of STDP operational principles

There may be many more STDP rules worth testing in our networks, but as these rules are conceptually and computationally simple, they are a good starting point from which to develop our learning algorithms further. Moving forward, testing more empirically validated STDP rules will be a crucial step in developing a more complete framework for our network models.

3. Network models

3.1 The MNIST dataset

Currently, all experiments with our spiking neural network models are on the MNIST handwritten digit dataset [7], which contains 28x28 pixel grayscale images of digits scanned from tax records, and their corresponding numeric labels; e.g., an image of the digit "1" is paired with the label 1. Each pixel is given by an integer in the range [0, 255], with 0 and 1 corresponding to completely white and completely black pixels, respectively. There are 60,000 training and 10,000 test examples, which are used to train and test a machine learning model. This problem is widely considered to be solved in the domain of supervised learning, as relatively shallow convolutional neural networks able to obtain test dataset classification accuracies greater than 99%. However, we chose to continue to develop our networks on this dataset before moving onto the more challenging CIFAR-10 or ImageNet image datasets for its manageable dimensionality and complexity, as we lack an efficient parallelized software implementation of spiking neural networks.

3.2 Baseline spiking neural network (SNN)

Our spiking neural network models are based on the network described in [1], which we will abbreviate as SNN. The network's architecture is comprised of three layers: an input layer, a layer of excitatory neurons, and a layer of inhibitory neurons. The input layer has a number of

neurons equal to the dimensionality of the input (784 in the case of the MNIST digit dataset), each of which is a Poisson spiking neuron [8] whose expected average firing rate is determined by the intensity of the input pixel which it corresponds to.

The input is connected to the excitatory layer with excitatory synapses in an all-to-all fashion, whose weights are learned during network training using one of the described STDP rules. The number of neurons in the excitatory layer is arbitrary, but equal to the number of neurons in the inhibitory layer. The excitatory neurons are connected in a one-to-one fashion with the inhibitory neurons, and each inhibitory neuron connects to the all neurons in the excitatory layer, except for the one from which it receives an excitatory synapse. Consult Figure 2 for a schematic of this network architecture.

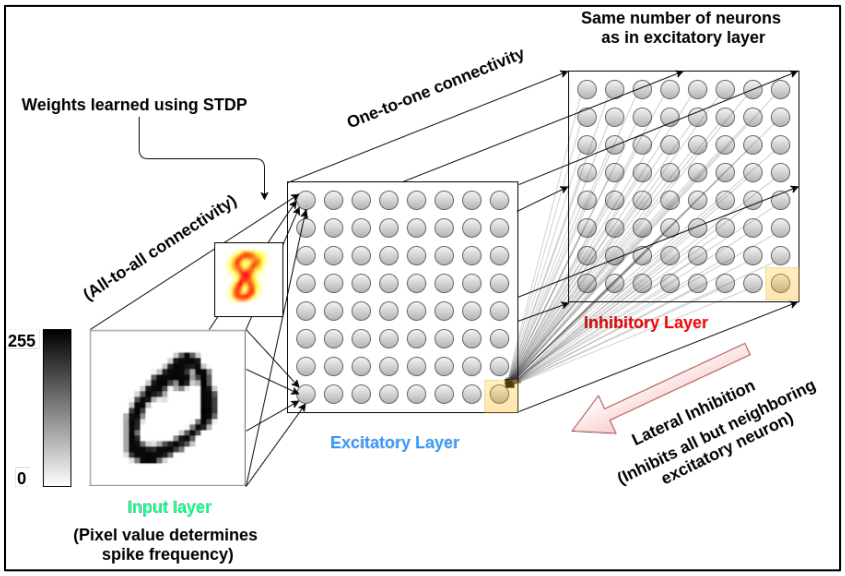


Figure 2: Baseline spiking neural network (SNN)

The weights of the synapses from excitatory to inhibitory layers are fixed such that a spike from an excitatory neuron will trigger a spike in the corresponding inhibitory neuron, which will in turn inhibit all other inhibitory neurons by a fixed amount, leading to competition between the neurons of the excitatory layer. While training the network, we set the input of the network to each data sample, one at a time, for 350ms each, solving the neuron and synapse equations with a time step of 0.5ms, after which we let the network settle back to equilibrium over a 150ms *rest period*, before running the next sample. After the training phase is complete, one thousand samples from the training data (split evenly between all digit classes) are used to set the *labels* of the neurons in the excitatory layer; e.g., a neuron is assigned the label '7' if it spiked the most overall for the data samples labeled 7. These labels are used in turn to classify the test data; i.e., a new data point is labeled '1' if the number of spikes attributed to class '1' (counted by summing all spikes from neurons labeled with '1') is greater than the number of spikes associated with any other class. We call this the "all" *voting scheme*.

3.3 Convolutional spiking neural networks (CSNNs)

Our first modification to the SNN architecture was loosely based on the widely utilized convolutional neural networks (CCNs) found in the deep learning literature [9]. Instead of connected all neurons in the input layer to each neuron in the excitatory layer, such as with the SNN model, we connect regularly-sized and spaced chunks of the input neurons to specific, spatially arranged neurons in the excitatory layer, populations which we call *patches*. Each neuron in an excitatory patch connect to a unique inhibitory neuron, which in turn inhibits all spatially similar neurons in all other excitatory patches; e.g., if the upper-left neuron in the first excitatory patch fires, its corresponding inhibitory neuron will inhibit the upper-left neuron in all other patches.

This modification was introduced in part because the SNN model tends to *memorize* prototypical images from the dataset by storing their pixel values as a particular excitatory neuron's synaptic weights. Although these stored digits tend to smooth out irregularities present in the image data, this scheme clearly does not achieve a compact compression.

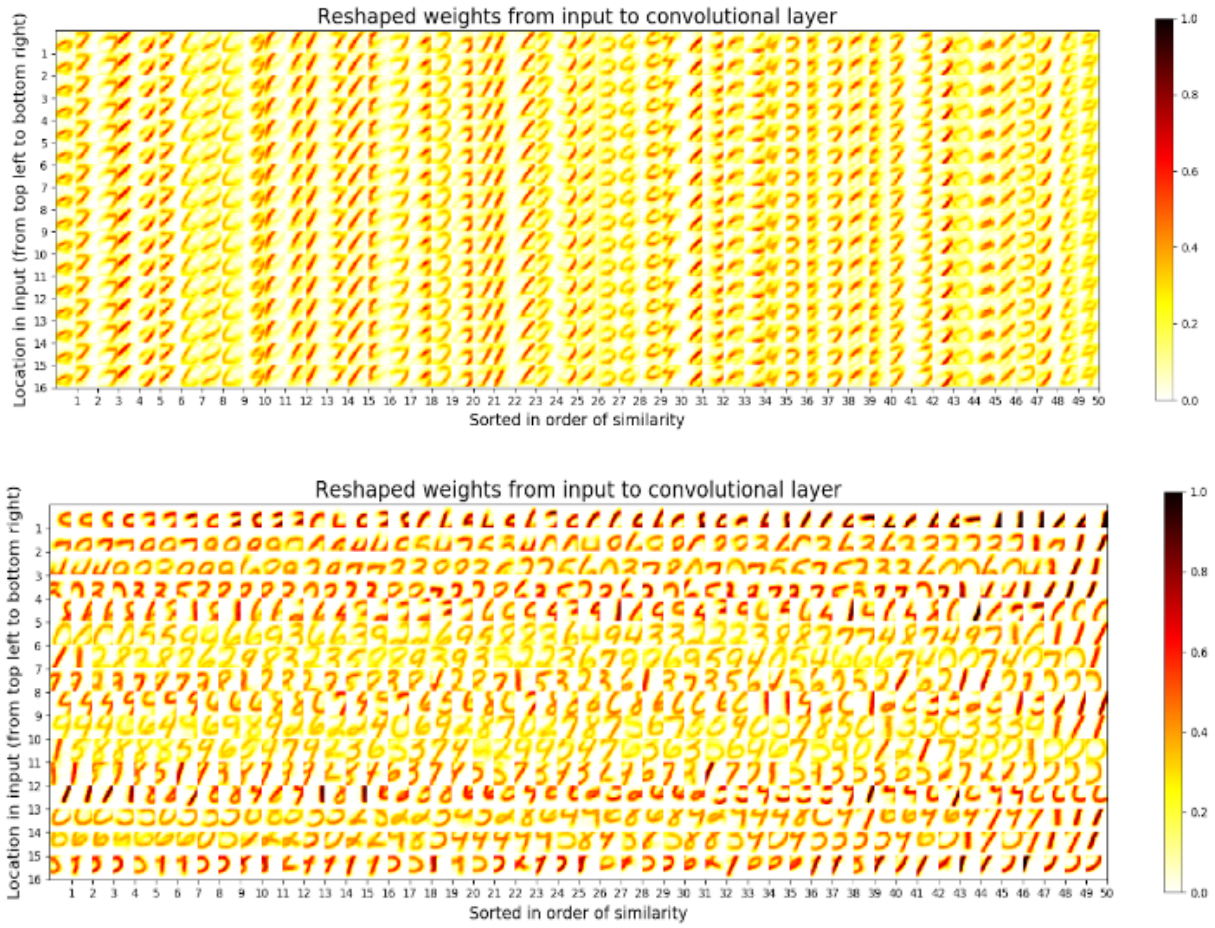


Figure 3: Example convolution filters following STDP learning process; (a) top plot: with employing weight sharing; (b) bottom plot: no weight sharing.

We define a *convolution size* k, covering a square region of size $k \times k$, *stride* length s, which determines the number of neurons we shift by, horizontally and vertically, across the input, and number of convolution patches m, which allows us to control the capacity of the model; i.e., how

many features of the input to learn. The convolution window, which covers the upper-left corner of the input connects to the upper-left neuron in each excitatory patch; likewise, the window which covers the bottom-right corner of the input connects to the bottom right neuron in each excitatory patch. We call the set of weights on the synapses, which connect a convolution window to a neuron in the excitatory layer a *filter*, two collections of which we've included in Figure 3a and 3b.

The implementation of CNNs typically includes a mechanism called weight sharing, in which all neurons within a patch learn and share a common set of weights. This drastically reduces the number of learned parameters in the convolutional layer, and gives rise to *translational invariance*, a property, which allows a convolutional layer to detect a single feature at any given location in its input. We implemented our CSNN model both with and without this feature, and discovered that in general, our networks accomplished much greater accuracy without it. This is due in part to the shallowness of the models we've tested, the relatively few parameters involved in the training process, and the difficulty in designing a weight-sharing mechanism with spiking neurons whose behavior is determined by ordinary differential equations.

We introduced a few new voting schemes to achieve a better classification accuracy with this model's architecture in mind. The first, which we refer to as the "most-spiked per patch", counts only the spikes from the neuron in each excitatory patch, which spikes the most out of all other neurons in the patch. This mechanism was designed with weight sharing in mind: If a neuron in a patch detects the filter it has learned, there is a good chance that feature does not appear elsewhere in the input, especially for large convolution window size. Another voting scheme, called "top-percent", counts only the spikes from the neurons whose spike count over the iteration lies in a certain top percentile; e.g., the top 10th percentile. Neither of these voting schemes greatly improved the training and test accuracy of the spiking network models, but under certain conditions, we may prefer one voting scheme to another.

3.4 Convolutional spiking neural network with between-patch connectivity (CSNN-PCs)

A further modification of the baseline SNN architecture added excitatory synapses between neurons in separate patches in the layer of excitatory neurons. These weights are also modified by STDP, typically with the same rule and parameters as with the synapses from the input to the excitatory layer. We tested several connectivity schemes between excitatory patches, and connectivity schemes between the constituent neurons in the patches. For example, we experimented with connectivity between all patches and between adjacent pairs of patches, and connected neurons to those in other patches whose receptive fields lay adjacent to its receptive field. See Figure 4 for an example of the latter connection scheme, and Figure 5 for a diagram of the network architecture, also useful for understanding the CSNN architecture.

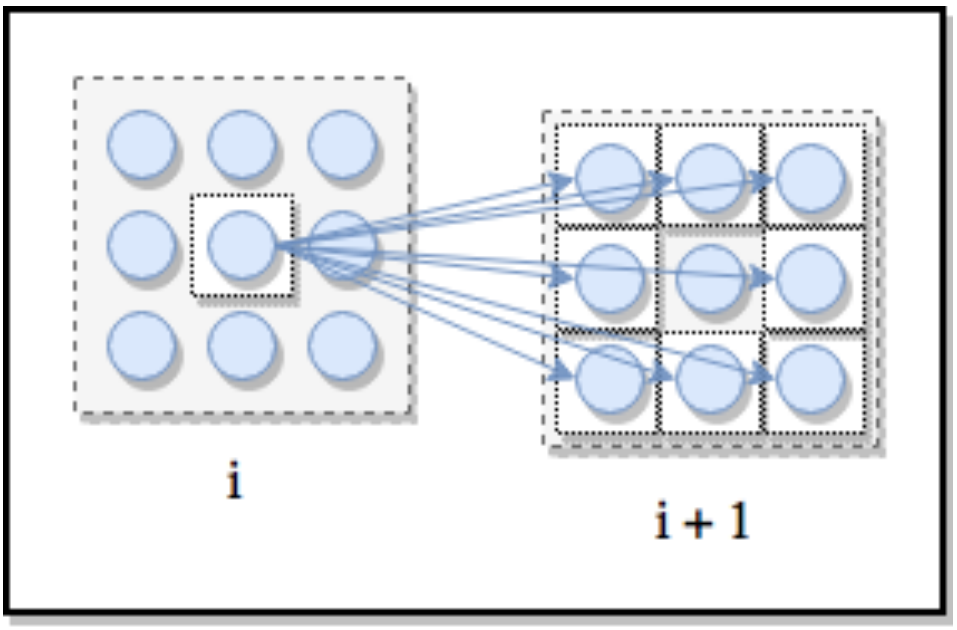


Figure 4: 8-Illustration of lattice connectivity in the proposed CNN with lateral patch connections; connections are shown from patch *i* to *i+1*; reciprocal connections from *i* + 1 to *i* are omitted for clarity.

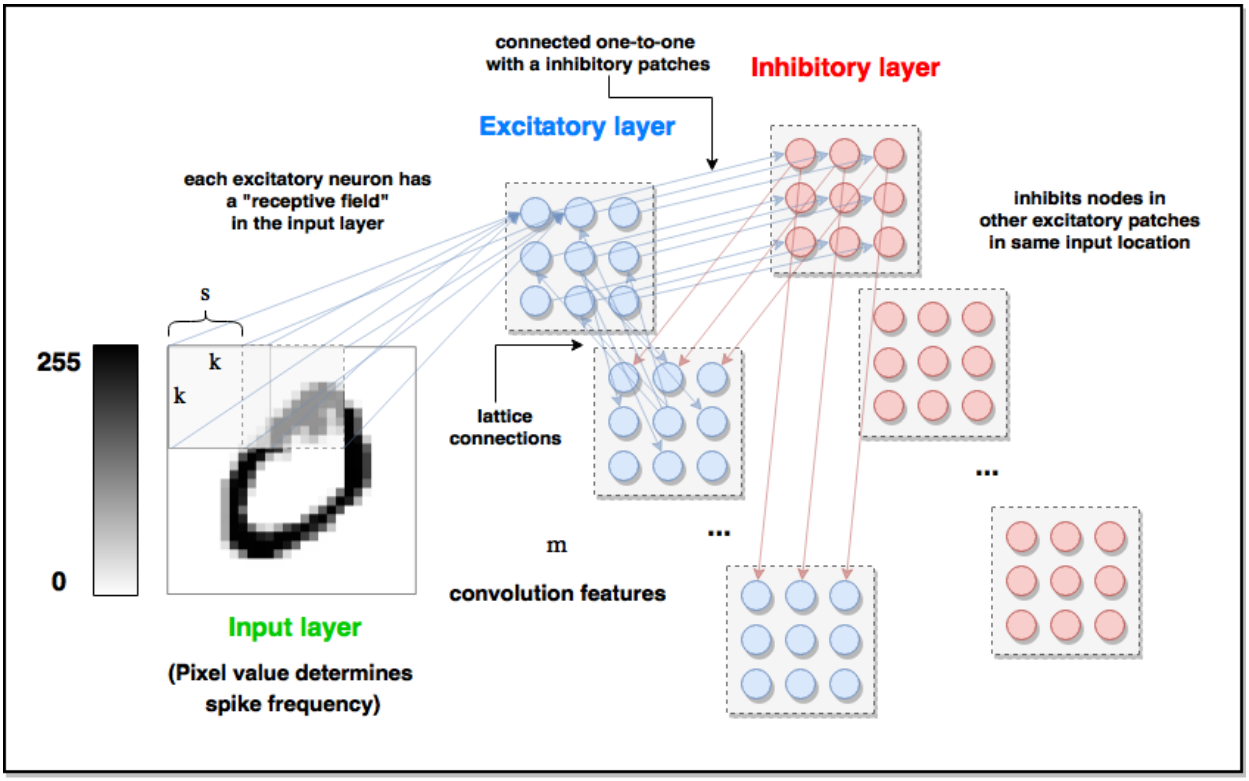


Figure 5: Convolutional spiking neural network with between-patch connectivity

4.

5. Accuracy of the results

5.1 SNN Baseline

We trained and tested the SNN architecture from [5] using the four STDP rules described above. We averaged their accuracies on the MNIST test data averaged over three trials after training on a single pass through the training data, long after their accuracies had converged. The results are reported in Table 1.

	Without Presynaptic Modification	With Exponential Weight Dependence	With Presynaptic Modification	Both
Diehl & Cook Algorithm	89.35	89.28	90.54	88.51

Table 1: SNN model accuracy by STDP rule

Since the STDP rule which included presynaptic modification was the clear victor in these experiments, we use this rule in the rest of the experiments described in this report.

We trained and tested the SNN architecture with 25, 50, 100, 200, 400, 800, 1600, 3200, and 6400 excitatory and inhibitory neurons, running three trials per model, and averaged their results. We plotted the results of these experiments in Figure 6, along with a deep convolutional neural network baseline, depicted in orange. We note that, as the number of excitatory and inhibitory neurons increases, the network's capacity increases and approaches the accuracy of the CNN. The larger networks are able to *memorize* more prototypical digit patterns than their smaller counterparts due to their larger capacity.

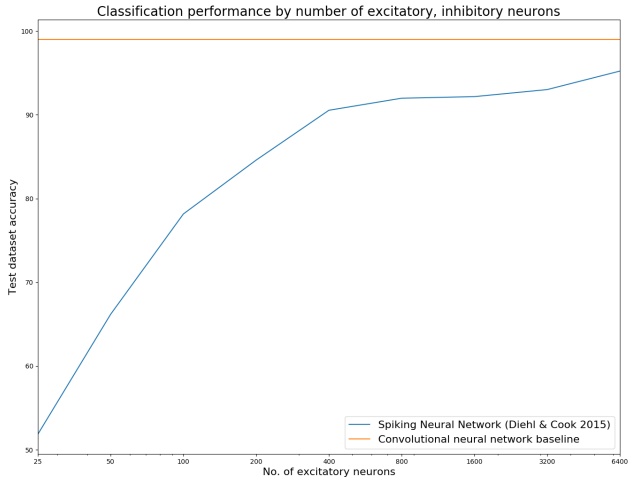


Figure 6: SNN test accuracy vs. number of excitatory neurons

5.2 CSNNs and CSNN-PCs

We trained and tested a number of CSNN and CSNN-PC models. Notice that a CSNN is a CSNN-PC in the special case that the connectivity scheme between patches includes no

connections. We include results for a number of networks trained on a single pass through the MNIST digit dataset in Tables 2 and 3. Here, the abbreviations "WS" and "NWS" correspond to *weight sharing* and *no weight sharing*, "None", "Pairwise", and "All" refer to the connectivity scheme between patches, and "All", "Most-spiked", "Top percent", and "Corr. clustering" refer to different excitatory neuron voting schemes. We found that convolution size $k = 12, 14, 16$, no weight sharing, and the "All" voting scheme produced the best results reliably, and plan to continue our experiments with these parameters in mind.

k	Voting	Connectivity					
		None		Pairwise		All	
		WS	NWS	WS	NWS	WS	NWS
8	All	63.13	80.21	58.63	**80.79**	58.71	68.64
	Most-spiked	61.07	76.69	60.21	77.21	60.81	70.83
	Top percent	67.97	73.83	65.66	75.22	64.78	69.43
	Corr. clustering	45.35	79.57	37.56	79.24	48.85	75.04
10	All	66.86	82.58	62.92	**82.95**	66.78	79.75
	Most-spiked	68.99	79.68	63.86	79.44	66.82	76.42
	Top percent	72.50	78.16	68.33	78.39	69.32	74.11
	Corr. clustering	54.53	80.03	41.67	80.37	51.51	75.70
12	All	63.45	**83.56**	63.05	83.31	67.24	80.85
	Most-spiked	66.72	81.18	67.60	81.31	69.51	77.99
	Top percent	71.27	80.50	72.07	80.56	71.45	77.46
	Corr. clustering	47.54	81.20	49.24	80.58	59.97	76.10
14	All	64.31	**84.18**	65.13	**84.18**	60.63	80.50
	Most-spiked	66.79	82.86	69.74	83.19	68.45	80.01
	Top percent	69.19	82.04	72.24	82.02	72.20	78.61
	Corr. clustering	63.31	80.83	62.20	81.17	64.05	76.09
16	All	59.27	**80.02**	60.01	79.48	57.52	76.62
	Most-spiked	66.36	79.39	67.35	78.88	66.04	76.12
	Top percent	67.76	76.82	67.32	76.37	66.08	73.64
	Corr. clustering	62.69	78.75	59.91	77.86	60.39	75.52
18	All	56.15	**78.93**	57.21	78.89	57.98	76.25
	Most-spiked	66.35	78.76	66.19	78.77	64.53	75.85
	Top percent	67.40	76.41	69.85	76.33	68.72	73.75
	Corr. clustering	63.49	77.30	63.35	77.66	64.74	73.49
20	All	61.00	75.82	60.73	75.13	57.97	73.76
	Most-spiked	68.28	**76.22**	68.87	75.95	63.51	73.29
	Top percent	70.22	74.38	68.66	73.75	66.24	70.72
	Corr. Clustering	69.03	73.88	67.04	73.15	68.81	72.34
22	All	63.84	72.81	65.93	**73.29**	64.91	68.57
	Most-spiked	66.90	73.25	66.91	73.10	69.68	68.58
	Top percent	69.22	70.64	67.32	71.90	69.11	66.60
	Corr. clustering	72.90	68.63	70.27	70.90	70.11	64.62
24	All	68.24	67.53	63.82	68.77	67.70	67.65
	Most-spiked	68.55	66.51	67.35	68.73	68.65	67.46
	Top percent	70.17	65.65	67.23	67.97	69.90	67.72
	Corr. clustering	71.70	64.67	68.47	67.43	**72.33**	63.02
26	All	65.42	67.56	66.98	67.06	65.33	64.24
	Most-spiked	63.53	67.47	66.73	66.25	63.96	62.36
	Top percent	64.31	67.16	**67.64**	65.59	66.21	60.54
	Corr. clustering	62.51	66.54	67.63	65.41	67.08	63.63
27	All	59.06	64.36	54.34	60.99	53.06	59.15
	Most-spiked	60.01	**64.71**	58.47	61.68	56.42	58.83
	Top percent	61.70	62.23	58.38	60.41	55.02	57.98
	Corr. clustering	62.94	60.79	58.06	60.47	57.22	57.45

Table 2: 8-lattice connectivity, 50 patches, stride 2 (except stride 1 for k = 26, 27)

k	s	Voting	Connectivity					
			None		Pairwise		All	
			WS	NWS	WS	NWS	WS	NWS
8	4	All	57.35	76.94	41.79	76.29	51.67	75.59
		Most-spiked	58.71	72.73	52.53	72.27	51.76	72.65
		Top percent	68.64	70.51	61.83	70.00	61.59	69.67
		Corr. clustering	66.85	78.71	30.93	**79.28**	61.08	77.21
10	6	All	46.37	74.95	47.48	74.57	51.41	74.77
		Most-spiked	47.48	71.57	52.76	71.51	52.94	70.88
		Top percent	54.18	69.91	57.80	70.36	57.76	69.81
		Corr. clustering	54.80	**75.46**	38.19	74.52	33.38	74.12
12	4	All	63.09	82.02	57.54	81.28	61.38	79.52
		Most-spiked	58.25	78.59	61.41	78.16	64.36	76.72
		Top percent	62.68	78.00	66.97	77.73	68.58	76.35
		Corr. clustering	70.86	79.08	35.37	**80.35**	67.93	78.26
14	2	All	65.52	83.94	63.96	**84.32**	65.48	82.52
		Most-spiked	67.27	82.39	67.47	83.17	69.47	82.31
		Top percent	69.49	81.76	71.07	82.32	70.49	81.64
		Corr. clustering	46.99	81.33	44.10	80.31	64.01	65.06
16	4	All	53.52	**80.43**	56.23	80.00	56.35	79.23
		Most-spiked	65.78	79.79	64.92	78.68	67.66	78.00
		Top percent	66.85	77.64	66.61	77.79	69.60	75.74
		Corr. clustering	69.86	79.35	41.23	78.73	72.40	77.28
18	2	All	62.08	79.24	62.22	79.23	56.31	78.68
		Most-spiked	63.66	79.28	63.32	79.26	68.12	78.62
		Top percent	64.78	77.51	64.80	76.99	**70.38**	76.62
		Corr. clustering	42.76	65.42	45.73	67.29	66.27	65.76
26	1	All	63.72	68.79	61.21	**69.63**	60.57	65.77
		Most-spiked	63.37	67.86	63.58	69.45	63.50	65.77
		Top percent	65.97	65.95	63.62	68.48	63.67	64.13
		Corr. clustering	60.09	66.15	46.31	67.07	48.90	63.00
27	1	All	64.97	65.08	64.27	63.62	68.54	64.06
		Most-spiked	67.23	64.87	62.42	63.57	66.29	65.08
		Top percent	67.73	64.26	63.67	62.93	66.08	63.42
		Corr. clustering	54.58	61.90	50.84	62.94	60.93	**65.59**

Table 3: 4-lattice connectivity, 50 patches, variable stride

6. Comparison – Deep neural networks vs. spiking neural networks

An important motivation for this work was to circumvent some of the drawbacks, which are typical to the deep learning approach to AI. Here, we include remarks of the relative computational efficiencies of our spiking neural network models and those used in deep learning. We note that these comparisons are necessarily incomplete and crude approximations, as our approach with SNNs is largely untested and there does not yet exist software frameworks for the fast, parallel simulation of comparably sized models.

6.1 Memory

In order to run the training phase of the spiking network models, we must record synaptic traces (for modifiable synapses) and conductance, neuron membrane potential, and elapsed time between Poisson spiking neuron spikes. This amounts to $2 * 784n_e$ values stored for pre- and post-synaptic traces, $784n_e + n_e^2$ values stored for synaptic conductance, and 784 values stored for the elapsed time between Poisson spiking neuron spikes. Empirically, we found that simulations of various size were able to run on laptop machines with 8Gb of memory without a problem.

The spiking neural network models have the potential advantage of being more memory efficient then their deep learning counterparts. Deep neural networks must cache the results of its layer-wise computation produced during its forward pass, and use these cached results to compute its backward pass. This effect is again proportional to the number of layers utilized in the deep neural network.

6.2 Convergence speed

While training the CSNN and CSNN-PC models, we observed that the number of data samples needed to train the model to its optimal accuracy was much fewer the number of samples needed for a CNN. We include a plot of the training accuracies over time of both a CSNN and and a CNN, both selected from their respective class of models to be particularly high performing, in Figure 7. Our spiking neural network models converge faster than typical deep learning networks to reasonable classification accuracy levels, partly because they have to train less parameters. For a comprehensive performance evaluation, extensive experimentation is in progress.

Figure 7: Training accuracy of CSNN vs. CNN models

7. Conclusions and Future Work

Our convolutional spiking neural network models with lateral patch connectivity introduce a novel approach to machine learning and pattern recognition. They show great potential because of their ability to scale to large problem instances with modest increase in computational demand and memory use, their biologically inspired computation units, architecture, and learning mechanisms, and their ability to learn good representations of data without the need for labels. With our convolutional spiking networks, it is straightforward to explain classification decisions by inspecting the filters corresponding to the most active excitatory neurons, once the network has reached a somewhat steady-state activation.

Our approach aims at a tradeoff between the decreased classification performance inevitably due to the unsupervised nature of the model, and the advantage provided by the distributed nature of the local training architecture. We plan to test different STDP rules, connectivity patterns from input to excitatory or between excitatory patches, inhibition schemes, neuron and synapse equations, and excitatory neuron labeling and voting schemes.

References

[1] Lecun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning." *Nature* 521.7553 (2015): 436-44. Web.

[2] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining And Harnessing Adversarial Examples". *Arxiv.org*.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. *Arxiv.org*.

[4] Sutskever, I., Vinyals, O., & Le, Q. (2014). *Sequence to Sequence Learning with Neural Networks*. *Arxiv.org*.

[5] Diehl, Peter U., and Matthew Cook. "Unsupervised Learning of Digit Recognition Using Spike-timing-dependent Plasticity." *Frontiers in Computational Neuroscience* 9 (2015): n. pag. Web.

[6] Markram, H., W. Gerstner. "Spike-Timing-Dependent Plasticity: A Comprehensive Overview." *Frontiers in Synaptic Neuroscience* 4 (2012): n. pag. Web.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.

[8] Averbeck, Bruno B. "Poisson or Not Poisson: Differences in Spike Train Statistics between Parietal Cortical Areas." *Neuron* 62.3 (2009): 310-11. Web.

[9] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." *Communications of the ACM* 60.6 (2014): 84-90. Web

See Report part on Q4 next

DARPA Task1 Report Supplement on Q4

Leaky integrate-and-fire neuron

In order to reduce the computational demands of our SNNs, we removed some complexity from their computational units; i.e., the neuron and synapse models. In [3], the authors chose a leaky integrate-and-fire (LIF) neuron along with modeling excitatory and inhibitory conductances. The next simplest choice is to use the plain LIF neuron, which removes the neurons' state's dependence on the synaptic conductances as well as the modeling of synapse state variables. The membrane potential v is therefore given by

$$\tau \frac{dv}{dt} = v_{rest} - v,$$

where τ and v_{rest} are defined as above, and the neuron spiking and resetting behavior are the same. In the event of a spike from a presynaptic neuron, the membrane potential v of the postsynaptic neuron is updated by the value of the weight w of the synapse which connects them. We are currently working to match the behavior of SNNs with the more complex neuron and synapse models, but we are confident that this is possible with some fine-tuning of network hyper-parameters.

LM-SNN Network Architecture

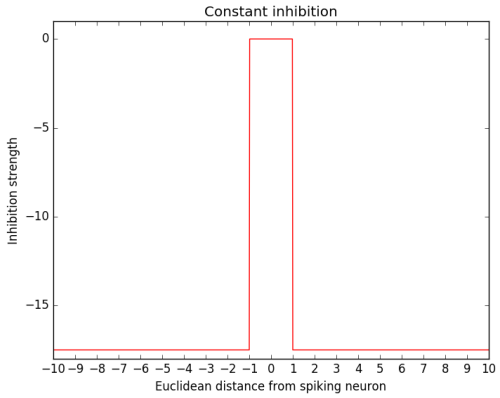
We implemented a new inhibition scheme in order to mitigate the total inhibition characteristic of the SNN from [3], and from our convolutional spiking neural network (CSNN) models (between neurons who share the same *receptive field).* Inspired by the self-organizing map (SOM), we arrange the excitatory neurons in a 2D lattice (requiring a square number of excitatory and inhibitory neurons) and introduce two new mechanisms which operate during the learning:

1. Instead of inhibiting all other neurons at a large constant rate, we vary the level of inhibition with distance between any two neurons. As a first attempt, we chose to increase the inhibition in proportion to the square root of the Euclidean distance. This requires a parameter c_{inhib} which determines the rate at which the inhibition increases, as well as a parameter c_{max_inhib} that gives the maximum allowable inhibition. With proper choice of c_{inhib}, in the event that an excitatory neuron reaches its firing threshold, instead of preventing all other neurons from firing for the rest of the iteration, a neighborhood of nearby neurons will be weakly inhibited and may have the chance to fire. This encourages neighborhoods of neurons to fire for the same inputs and, therefore, learn similar filters. See Figure 1 for plots of the effects of constant inhibition (introduced in [**?**, eth snn] and our new scheme. Note that neurons do not inhibit themselves upon firing, as illustrated in the figure.

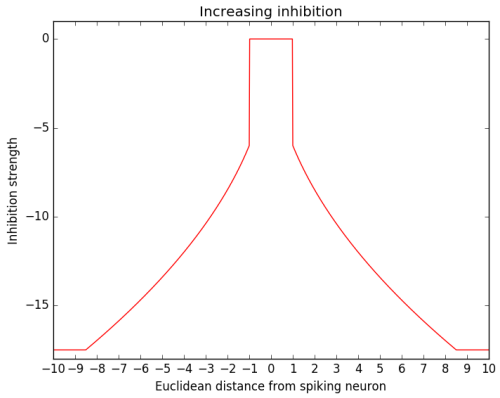
2. As with the training of an SOM, on every training iteration we update the values of a neuron's neighbors' weights by a fraction of the weights learned over the iteration. This mechanism requires a parameter $c_{strengthen} \in [0, 1]$ which determines the fraction of learned weight which are shared between neighboring neurons. If we denote the weight learned by neuron indexed on the lattice by (i, j) on some iteration k as $\Delta w^k_{(i,j)} = w^k_{(i,j)} - w^{k-1}_{(i,j)}$, and the set of neurons neighboring the (i, j)-th neuron as $\mathcal{N}(i, j) = \{(i', j') : \max(|i - i'|, |j - j'|) = 1 \ \wedge \ \neg((i, j) = (i', j'))\}$, then, at the end of the iteration, we set $w^k_{(i',j')} = w^k_{(i',j')} + c_{strengthen} \times w^k_{(i,j)} \ \forall (i', j') \in \mathcal{N}(i, j)$.

These two mechanisms together allow the excitatory neurons to learn groups of similar filters which may vary smoothly between input classes over the 2D lattice. We call this family of SNNs Lattice Map Spiking Neural Networks (LM-SNN).

There are a number of ways to define the inhibition (and excitation) as a function of the distance between two neurons. It is possible to use other distance measures; for example, Manhattan and Box distance, and we may want the inhibition to increase as linearly or with the square of the distance between neurons, regardless of distance measure used. We are also interested in investigating the usefulness of Mexican hat inhibition [6] in our networks, in which nearby neurons are excited by a spiking neuron, and farther away neurons are inhibited, yet the inhibitory effect decays to zero with increasing distance. See Figure 2 for a plot of the form of Mexican hat inhibition strength with distance. Note that this magnitude of the excitation and inhibition and the rate of decay of this function may be controlled with scaling hyper-parameters.



(a) Constant inhibition



(b) Increasing inhibition with distance

Figure 1: Inhibition as a function of Euclidean distance

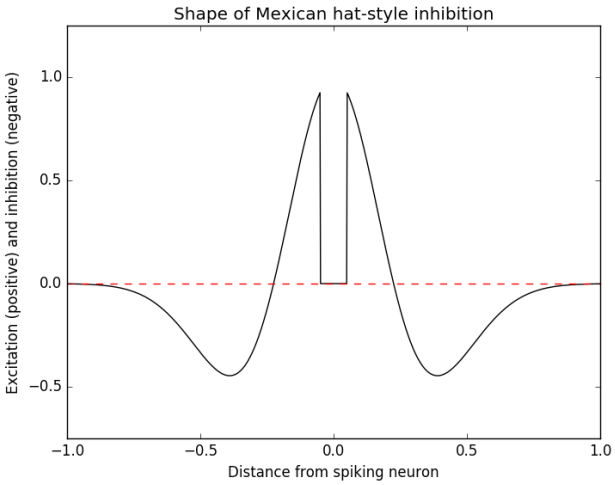
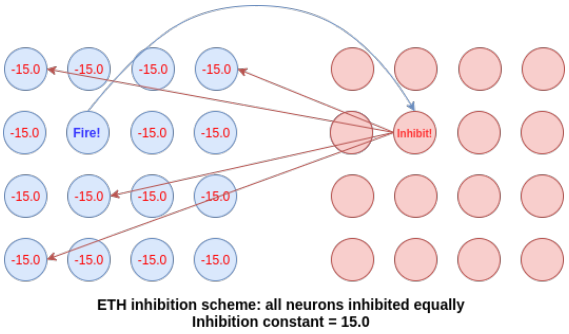
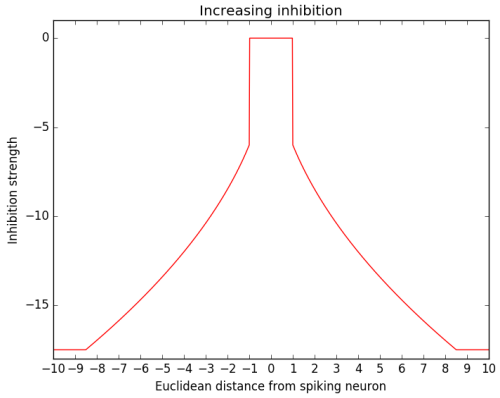


Figure 2: Form of Mexican hat-style inhibition



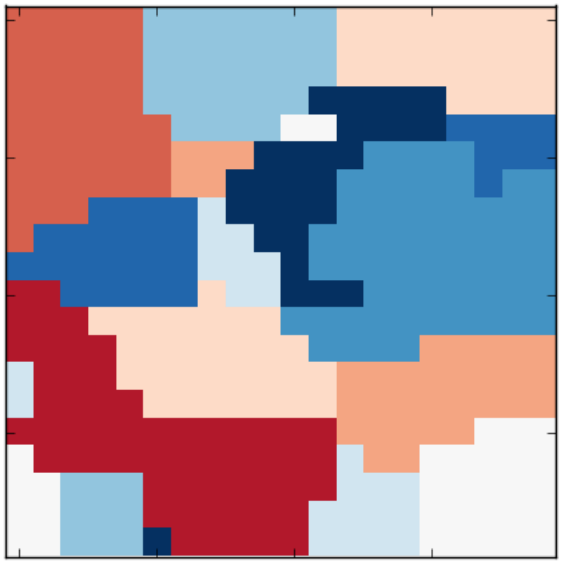
(a) Constant large inhibition



(b) Inhibition increasing with distance

Figure 3: Constant inhibition vs. increasing inhibition

(a) 2D lattice of neuron filters



(b) Corresponding neuron class assignments

Figure 4: LM-SNN filter map and class assignments

We are experimenting with these modified inhibition schemes using fully-connected networks (as in [3]) in order to better analyze the network learning dynamics and develop a corresponding classification scheme. In Section 3.1, we list preliminary accuracy results using fully-connected networks with a lattice of 20×20 excitatory and inhibitory neurons. We plan to extend our experiments with the LM-SNN and use the convolutional structure implemented in the CSNN architecture. See Figure 3 for a 2D diagram of how the modified inhibition scheme works. See Figure 4a for an example filter map from a LM-SNN consisting of 20×20 excitatory and inhibitory neurons, and 4b for the corresponding neuron label assignments calculated by a neuron labeling algorithm.

Table 1: LM-SNN Test Dataset Classification Accuracy

c_{inhib}	$c_{strengthen}$	all	confidence weighting	most spiked neuron	top 10% spiking
0.5	0.05	74.69	75.38	72.72	73.61
0.5	0.075	73.02	73.82	71.06	72.65
0.5	0.1	73.66	73.99	72.02	72.75
0.5	0.125	72.96	73.29	71.14	71.68
0.5	0.15	71.18	72.07	69.07	70.67
0.75	0.05	75.85	75.89	74.22	74.96
0.75	0.075	72.58	72.95	71.95	72.11
0.75	0.1	72.49	72.98	72.29	72.71
0.75	0.125	75.38	75.50	73.18	74.96
0.75	0.15	73.58	73.92	71.78	72.85
1.0	0.05	72.48	72.44	70.67	70.44
1.0	0.075	75.93	76.13	73.82	75.05
1.0	0.1	75.02	75.43	72.84	74.21
1.0	0.125	73.38	74.12	72.05	71.81
1.0	0.15	75.08	75.52	73.65	74.44
2.0	0.05	76.50	76.80	75.60	75.00
2.0	0.075	75.64	76.00	74.35	74.96
2.0	0.1	75.04	74.95	72.95	73.74
2.0	0.125	76.50	76.90	74.64	75.35
2.0	0.15	77.02	77.68	74.94	75.80
3.0	0.05	77.92	77.44	76.13	77.09
3.0	0.075	77.08	77.40	75.21	76.30
3.0	**0.1**	**78.34**	**78.35**	**76.94**	**76.88**
3.0	0.125	76.88	76.65	74.34	75.43
3.0	0.15	76.56	76.55	74.50	75.46

Results

LM-SNN Accuracy Results

We ran a number of experiments using the LM-SNN networks while searching over a grid of values of the hyper-parameters c_{inhib} and $c_{strengthen}$. We trained and tested networks using settings of these parameters on 30K / 10K, training / test examples from the dataset, respectively. For some *voting schemes*, neurons are assigned the label of the input class for which they have fired most, and these labels are used to classify new data based on network activation. In the *confidence weighting* voting scheme we implemented, we keep track of a vector of proportions of times for which a neuron fires for each input class, and used a weighted average of these to classify new inputs. Each network was trained and tested three times with different initial conditions and averaged their test dataset classification accuracy. The results are shown in Table 1. We observe that, with increasing c_{inhib}, the accuracy of the networks increasing as well. The LM-SNN with $c_{inhib} = 3.0$, $c_{strengthen} = 0.1$ performed the best over all voting schemes, leading us to believe the voting schemes are somewhat independent of network activation while varying these particular hyper-parameters.

Conclusions and Future Work

We have demonstrated a spiking neuronal network with self-organization and clustering properties in an unsupervised fashion. We have taken a step back from our study of the convolutional spiking neural networks in order investigate difference modes of interaction between the inhibitory and excitatory layer of neurons. Our goal is to combine the self-organizing behavior of the LM-SNN architecture with the efficiency and scaling benefits of the CSNN architecture, in hopes of improving classification accuracy and learning

convergence rate. We hope that slicing up the input space with a convolutional structure will ultimately allow us to obtain a more robust representation of input data.

Our preliminary work on LM-SNNs will be expanded to include more experiments with larger searches over hyper-parameters. We aim to find the best inhibition-excitation scheme between neurons in order to speed the learning of the neuron filters while reaching a higher classification accuracy. Implementing a simpler model of neuron and synapse will speed network operation and also ease the analysis of the learning dynamics. Discovering a better neuron labeling and new input classification scheme will be important in fully utilizing the filters learned by our networks.

References

[1] Lecun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning." *Nature* 521.7553 (2015): 436-44.

[2] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining And Harnessing Adversarial Examples"

[3] Diehl, Peter U., and Matthew Cook. "Unsupervised Learning of Digit Recognition Using Spike-timing-dependent Plasticity." Frontiers in Computational Neuroscience (2015)

[4] Markram, H., W. Gerstner. "Spike-Timing-Dependent Plasticity: A Comprehensive Overview." Frontiers in Synaptic Neuroscience (2012)

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[6] H, v. (2017). Chaos in neuronal networks with balanced excitatory and inhibitory activity. - PubMed - NCBI . Ncbi.nlm.nih.gov.