# A PEEK INTO SPIKING NEURAL NETWORKS

Andrew Wei
nowei@cs.uw.edu

Kushal Jhunjhunwalla
kushaljh@cs.uw.edu

## Introduction

There is evidence that current neural network architectures that we use in deep learning are not a good representation of how the brain functions. Bengio et. al. [2] describe the flaws in the biological possibilities of back propagation. The backpropagation computations of Artificial Neural Networks (ANNs) are completely linear, whereas brain activity is a mix of linear and nonlinear operations. Also, neurons in the human brain communicate with binary values (or spikes) and not continuous values as in ANNs.

We attempt to better understand brain activity by implementing a neural network that would take into account a more plausible implementation of the brain. We do this by implementing what is commonly known as a Spiking Neural Network (SNN) and comparing the results of that network with similarly sized ANNs.

## Background

A neuron is the fundamental unit of the human nervous system. It is the cell responsible for receiving sensory information and relaying electrical signals at every step. This is well-modeled in ANNs through the use of neural layers, which are usually fully connected with the previous layer's neurons and relay information to the next layer of neurons. However, in the human brain, neurons communicate by generating and propagating electrical signals called action potentials or spikes [2].

Spike-Timing-Dependent plasticity (STDP) is a biological process that adjusts the strength of connections between the neurons in the brain. It is used to explain Long term potentiation (LTP) and long term depression (LTD). STDP changes synaptic strength as a function of the timing between pre- and post- synaptic action potentials.

Neural networks used in Machine Learning have been implemented so as to implement learning mechanisms similar to the human nervous system. Even though

Artificial neural networks today are great at many classification tasks, they are over-engineered and do not have the properties that the human brain has to optimize its functioning. Based on our research ([2,3,4,5]), we found out that there is a fundamental difference in how the human brain communicates. As described above, the brain uses spikes to communicate information instead of continuous values.

We believe that we should explore neural networks that are more aligned with the real nervous system. To do this we need to be able to integrate STDP in our neural networks, which is the principle behind Spiking Neural Networks (SNNs). SNNs integrate STDP and transfer information across the network based on binary spikes that are created when neurons reach a given threshold potential. The neurons that reach the threshold are the only ones that are updated for a given input. This also brings forward a possibility of using a lower power hardware device, as these are event-driven neural networks and may not require as much computational power as an Artificial Neural Network.

We wanted to further explore how we can use these networks to achieve better learning. For this, we turned to our knowledge of deep learning. Convolutions leverage color channels in images to learn features for filters and have become very common within the field of deep learning. We wanted to try out an amalgamation of deep neural networks and spiking neural networks to see if augmenting SNNs with learned filters would be helpful. Therefore, we make a network that uses convolutions and SNNs and observe its performance.

# Methods

We implemented a spiking neural network in PyTorch (based largely on [1]). Our spiking layer takes in an input and passes it through a fully connected layer. We then decay the membrane potential by some decay factor. The output is then used to update our membrane potentials to the layer. Then for each membrane, we then check whether the membrane potential exceeded a threshold. The membrane potentials that exceed the threshold are considered to have fired, propagating the excess signal to the next layer. We will discuss why we do this instead of the binarized output in the discussion section below. Then we update the potentials of the fired neurons to be slightly negative

to imitate a refractory period before they can be fired again. We output the potentials for the layer and the fired output. Our current implementation has the threshold, refractory potential, and decay multipliers as hyperparameters for our model.

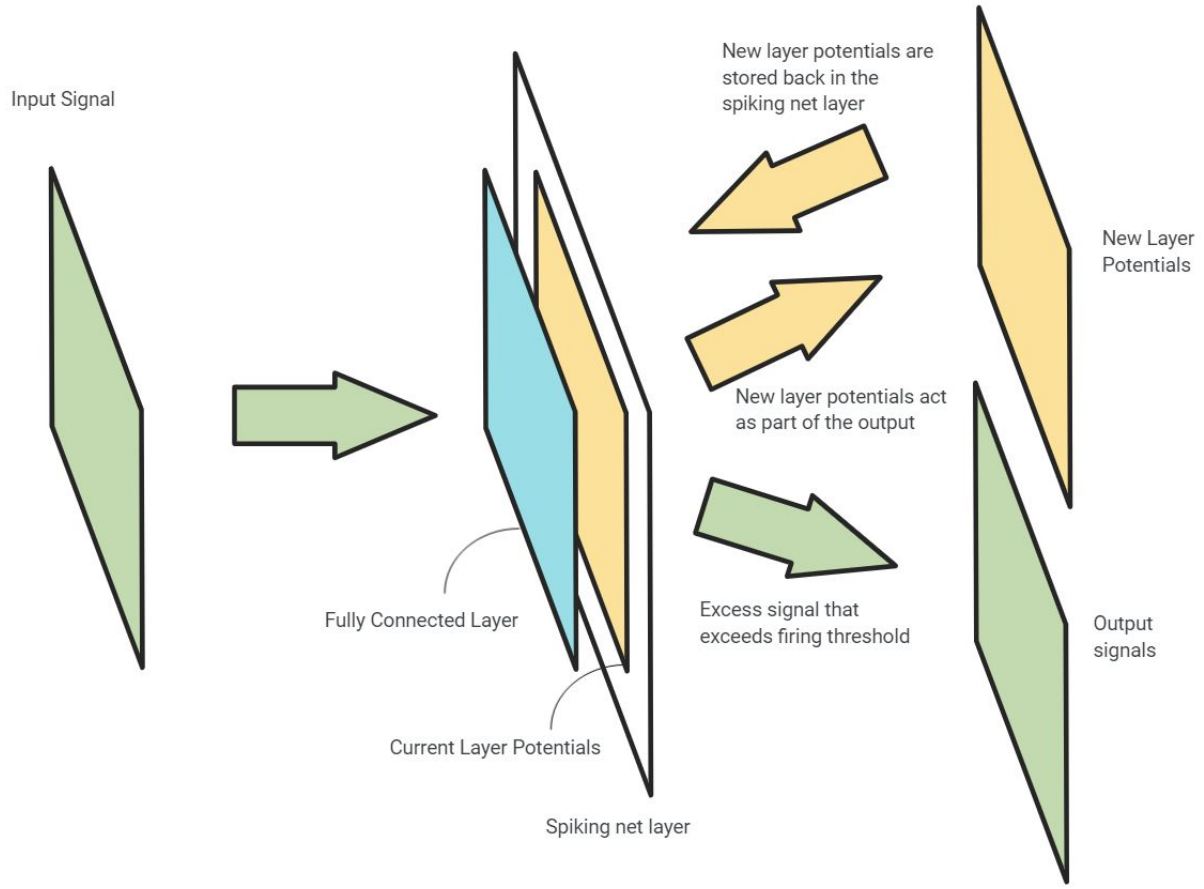


Figure 1. Spiking Net Layer structure. The layer takes in the input signals and passes them through a fully connected layer to update the current layer potentials and fires neurons that exceed the firing threshold, outputting the excess potential as the signal to the next layer.

The basic structure for our spiking neural network started with the input layer, two spiking net layers, with the last one acting as an output. We pass the input into the first spiking layer, then pass the output potentials of the layer into the second spiking layer.

We originally deviated from the implementation by Guillaume [1], in that our cell potentials were focused on the inputs rather than the outputs. We wanted to represent the signals coming in as input pulses, but our model had trouble learning due to weight initializations and hyperparameter choices such that the second layer was not firing

frequently enough for learning to be consistent. This led to long training times where little to no learning occurred.

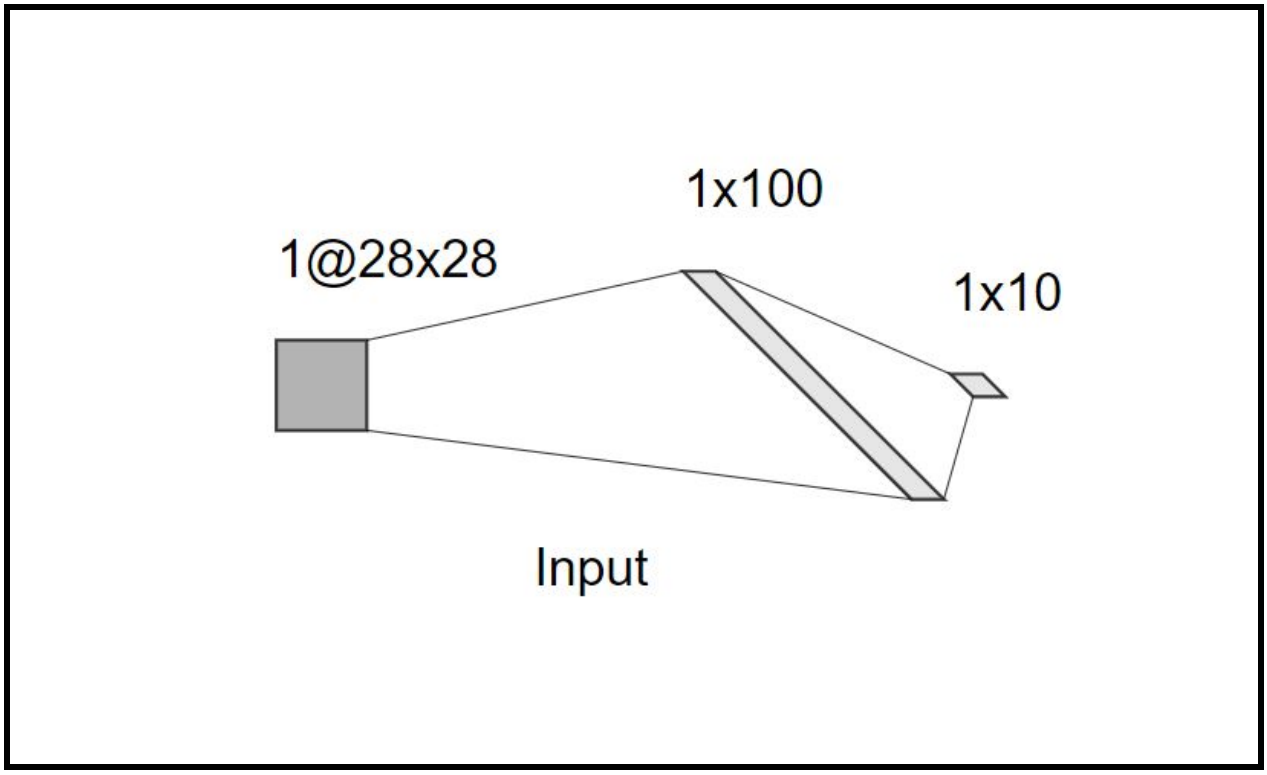


Figure 2. Neural Network structure implemented for experiments

There were also some doubts surrounding the implementation by Guillaume [1]. Specifically, while the output of the first spiking layer was used as the input to the second spiking layer, the potentials of the second layer were used as the output of the network instead of the outputs of the second layer. Eventually, we realized that they did this due to the firing of neurons being inconsistent and low in the first batches, so many of the outputs of the spiking layer would be 0. Thus, training the network to recognize the firing as the indicator of the output with this current setup may lead to slower training. We believe that the end result would be the same, given more time to tune the hyperparameters and initial weights. We also reasoned that the model is able to learn from the potentials by learning what hasn't fired in the potentials as what the output is and what has fired as what the output is not. This allows the weights for the softmax to be non-zero with higher weights on things that haven't fired (positive weight) and lower

weights on things that have fired (negative weight). This speeds up training significantly. We confirmed this by looking at the output and the predictions made by the model.

Figure 2 represents the network used for experiments in 2 hidden layer networks. This was the same architecture used by our spiking and non-spiking neural networks. The difference being that the hidden layers were implemented with different designs in mind. As the name suggests, the SNN implementation used spiking neurons whereas the non-Spiking neural network used traditional ANN neurons in its hidden layers (linearly connected weights with an activation function).

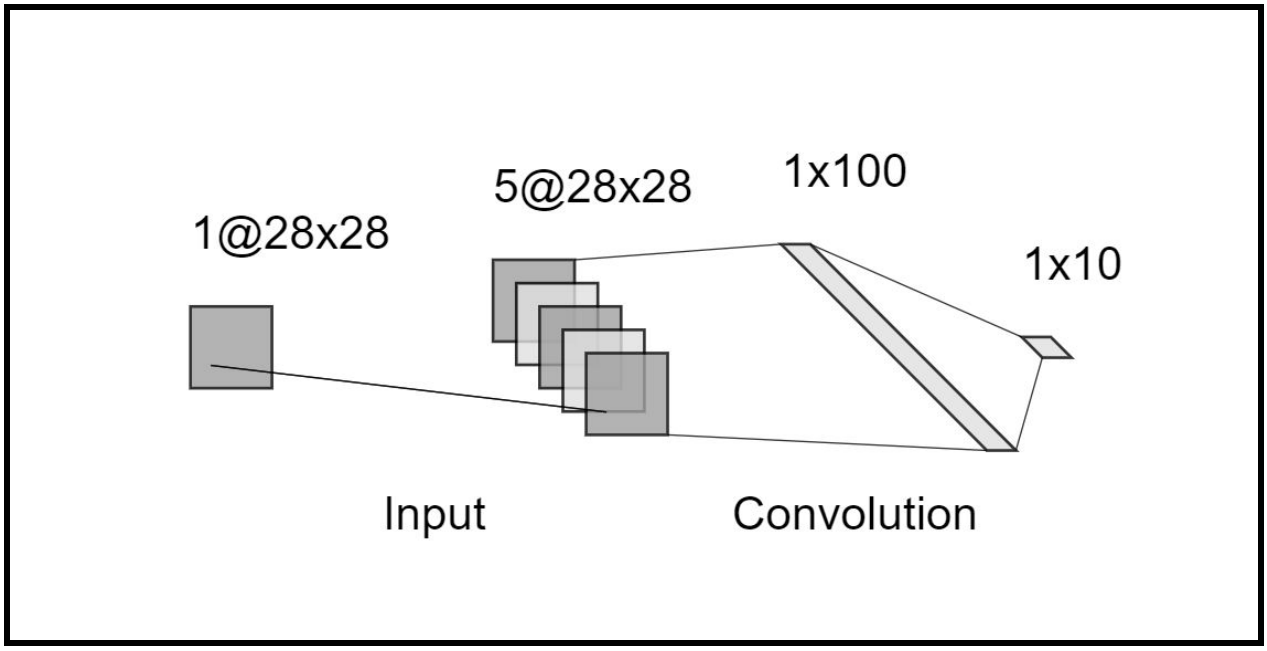


Figure 3. Spiking Neural Network with Convolutions implemented for experiments

We wanted to learn how we could further improve the performance of SNNs. Therefore, we tried combining the SNN structure with traditional deep learning components by adding a convolutional layer. This allows our model to learn filters, enabling us to see the impact of including the ability to learn important local features. We anticipate that combining filters with SNNs can help us augment established deep neural network structures with spiking neurons and give us better accuracy in our tasks. Figure 3 gives us an insight into what the augmented network looks like. We also tried this with two different kernel sizes - 3x3 and 5x5.

We tested our implementation in the setting of image classification on MNIST and CIFAR-10. For each image, we run it through the neural network multiple times (the exact number being a hyperparameter), with a random weight multiplied each time so different parts of the image will be highlighted or focused on during training as a consideration for noisy readings. The forward pass through the data multiple times represents different timesteps, allowing us to apply the decay to the membrane potentials between subsequent jittered inputs. We then take the average of the results and pass that through a softmax to get the predictions.

We compare the results of our spiking neural networks, and convolutional spiking neural networks to similarly-sized, traditional fully-connected neural networks in the next section.

# Results

The results we present are for similarly sized networks, with a similar number of layers and connections between the layers, save for the additional convolutional layers for the convolutional spiking neural networks. Even though our results are not revolutionary, we show that a network structured in the manner described can learn the concepts found within these datasets to a degree in which they were comparable with artificial neural networks. We note that the training of these spiking neural networks took significantly longer due to the simulation of timesteps.

We spent some amount of time tuning hyperparameters to achieve reasonable performance. We confess that we could’ve spent more time tuning the hyperparameters to achieve more significant results, but we aimed to show that these differently structured networks could achieve at least similar performances if not better.

When adding the convolutional layers, there was not a significant increase in the performance of the model that would justify augmenting a convolutional layer to the SNN architecture, although it’s likely that with a different architectural setup or hyperparameter choice, we could achieve greater performance.

| Model | Average Loss | Test accuracy |
|---|---|---|
| Non-Spiking Neural Net (with ReLU) | 0.0004 | 89.19% |
| Non-Spiking Neural Net (with LeakyReLU) | 0.0004 | 89.03% |
| Non-Spiking Neural Net (with Sigmoid) | 0.0013 | 75.57% |
| Spiking Neural Net | 0.0003 | 90.68% |
| Spiking Neural Net (with binary outputs for spiking layers) | 0.0011 | 71.0% |
| Spiking Neural Net + 5 3x3 conv filters | 0.0003 | 90.2% |
| Spiking Neural Net + 5 5x5 conv filters | 0.0003 | 90.5% |

Table 1. Results of different models on MNIST dataset with a cursory tuning of hyperparameters.

The learned filters from the convolutional layers do not seem to be very interpretable, which we attribute to the models picking up on local patterns that are not readily visible to human observers. We also notice that the performance of the spiking neural network with binary outputs for the spiking layers does not perform as well as its non-binary counterpart. Thus, we exclude it from the next experiment.

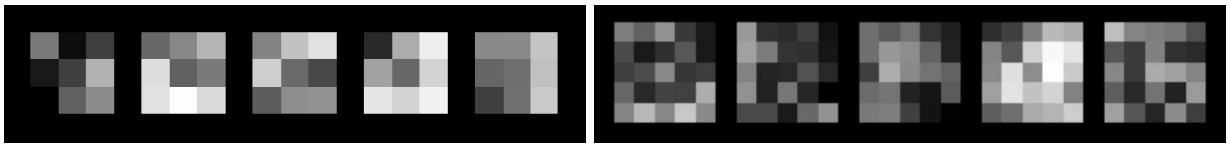

Figure 4. The learned filters from the 3x3 and 5x5 convolutional layers respectively.

We then moved on to training our models on CIFAR-10, which is generally considered to be more complicated than MNIST, even though they both share the task of classifying 10 different labels. While the accuracies are not competitive with existing deep neural networks, we show that the SNN architectures continue to perform with

similar accuracies to the simple non-spiking neural networks. This reinforces our belief that SNNs are at least as good at tasks as similarly sized ANNs. This makes us hopeful about the future of SNNs, but we believe that there needs to be additional research that needs to be done.

| Model | Average Loss | Test accuracy |
|---|---|---|
| Non-Spiking Neural Net (with ReLU) | 0.0016 | 43.64% |
| Non-Spiking Neural Net (with LeakyReLU) | 0.0016 | 43.75% |
| Non-Spiking Neural Net (with Sigmoid) | 0.0019 | 34.13% |
| Spiking Neural Net | 0.0015 | 48.31% |
| Spiking Neural Net + 5 3x3 conv filters | 0.0146 | 49.47% |
| Spiking Neural Net + 5 5x5 conv filters | 0.0145 | 50.38% |

Table 2. Results of different models on CIFAR-10 dataset with a cursory tuning of hyperparameters.

# Discussion

We note the differences between our implementation and STDP in that our implementation's sense of time is still dependent on the layered propagation and updates, while STDP is based on the timing of signal pulses entering groups of neurons. This is a significant difference because these layered updates essentially force groups of neurons to fire together, which means that we are missing out on neurons grouping naturally. Essentially, we are unnaturally enforcing that "neurons that fire together, wire together." Another difference is that even though we attempt to include a time component in our implementation, the time aspect does not allow for consistently decaying neurons due to natural, perceived time or the staggering of neuron activations. This means that we cannot accurately model the biological process of potential decay

over time and miss out on the possibility of encoding information through the relative timing of neuronal outputs that STDP is based on.

One last issue is that our implementation does not receive and output binary signals, but residual values that exceed the threshold after applying the thresholding. We have tested an implementation that outputs binary values, but the results are not comparable with if we had used the continuous values. We believe that this is in part due to the more expressive encoding of values in a continuous space, which makes up for our lack of distinguishability through timing.

# Future work

There were many parts of the models that could have been improved with increased hyperparameter tuning or architecture refinement, but we wanted to get a baseline for how a neural network that implements a version of STDP would perform.

STDP seems to be more event-driven than artificial neural networks in that they fire upon exceeding the threshold and connection strengths are based on relative timings of neurons. A possible direction for exploration is to look into implementations of SNN in hardware, as it seems like the medium can better capture the event-driven nature of STDP and decaying potentials.

When trying convolutional layers, the learned filters are not as interpretable as predefined filters like Gabor filters, which help capture the ideas of directional edges. An important benefit convolutional neural networks have over traditional artificial neural networks is the introduction of spatial relationships between local features, as it was inspired by the idea of receptive fields [8]. The features learned from convolutional neural networks may capture more information about peculiarities of the dataset than what the human brain actually does when interpreting images. Thus, we should explore the use of predefined Gabor filters in our convolutional spiking neural networks as there are interpretability benefits of using Gabor filters and arguments for biological plausibility [9].

# Conclusion

We have spent time reading about Spiking Neural Networks (SNNs) and exploring SNNs with an implementation as described in the earlier sections. As a part of learning more about SNNs, we compare the results of the models on the MNIST and CIFAR-10 datasets. Even though our discovery was not as fruitful as we believed they would be, we find that this implementation of SNNs can perform at least as good if not better than the simple Artificial Neural Networks that are used as building blocks for larger, deeper networks. There is still a lot of room to improve our capability of modeling STDP, but we have taken our first step towards understanding the mechanisms of spikes in the human brain. This acts as an important step towards realizing more biologically plausible classes of models so that we may better capture and understand the learning process that goes on in the brain.

# Acknowledgements

We would like to thank our instructor, Rajesh Rao, and teaching assistant, Beibin Li, for giving us the opportunity to explore the intersection of neurobiology and Artificial Intelligence which led us to learn about Spiking Neural Networks.

# References

1. Chevalier, G. (2020, March 16). Spiking Neural Network (SNN) with PyTorch: Towards bridging the gap between deep learning and the human brain. Retrieved December 13, 2020, from https://guillaume-chevalier.com/spiking-neural-network-snn-with-pytorch-where-backpropagation-engenders-stdp-hebbian-learning/
2. Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, & Zhouhan Lin. (2016). Towards Biologically Plausible Deep Learning.
3. Illing, B., Gerstner, W., & Brea, J. (2019, June 20). Biologically plausible deep learning - But how far can we go with shallow networks? Retrieved December 13,

2020, from https://www.sciencedirect.com/science/article/pii/S0893608019301741

4. Pfeiffer, M., & Pfeil, T. (2018). Deep Learning With Spiking Neurons: Opportunities and Challenges *Frontiers in Neuroscience, 12*, 774.
5. Ponulak F, Kasinski A. Introduction to spiking neural networks: Information processing, learning and applications. *Acta Neurobiol Exp (Wars)*. 2011;71(4):409-33. PMID: 22237491.
6. Diehl, P., & Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity *Frontiers in Computational Neuroscience, 9*, 99.
7. Hazan, Hananel, et al. “BindsNET: A Machine Learning-Oriented Spiking Neural Networks Library in Python.” Frontiers in Neuroinformatics, vol. 12, Dec. 2018, p. 89. DOI.org (Crossref), doi:10.3389/fninf.2018.00089.
8. Alake, R. (2020, June 12). Understand Local Receptive Fields In Convolutional Neural Networks. Retrieved December 16, 2020, from https://towardsdatascience.com/understand-local-receptive-fields-in-convolutional-neural-networks-f26d700be16c
9. S. Marĉelja, "Mathematical description of the responses of simple cortical cells*," J. Opt. Soc. Am. 70, 1297-1300 (1980)
10. Mozafari, Milad, et al. “SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks With at Most One Spike per Neuron.” Frontiers in Neuroscience, vol. 13, July 2019, p. 625. DOI.org (Crossref), doi:10.3389/fnins.2019.00625.