CSCI 2592N Final Project Report: Conversion of CNN to SNN

Jorge Chang, Yipu Gao, Heejun Lee, Peisen Zhou.
Brown University

Abstract

Spiking neural networks (SNN) are designed to mimic discrete spike trains in biological neurons. In addition to weight matrices, spiking neurons also maintain their membrane potentials and activation thresholds. For example, an activation spike would only be generated when a neuron's membrane potential exceeds its threshold. The sparse nature of spikes means that SNNs can be more efficient and faster on neuromorphic hardware. However, the discrete nature of SNNs means that they cannot be directly trained with gradient descent. One popular approach to overcome this challenge is converting artificial neural networks (ANN) to SNN. In this project, we attempted to convert all layers in a CNN to SNN and explored the conversion loss of different layers.

1. Introduction

Spiking Neural Networks (SNN) are a group of networks designed to mimic biological neurons. The most popular model for SNN is the Integrate-and-Fire (IF) model [1][3]. In this model, the neurons would keep track of their membrane potential and threshold voltages. Unlike the constant activation at each step in Artificial Neural Networks (ANN), a spike would only be produced when the accumulated membrane potential exceeds the threshold. The membrane potential would then be reset after the spike as in a real neuron. The sparse nature of the spike trains means that SNNs have the potential to be both more energy-efficient and faster at inference time. The discrete spikes produced from the IF model also suit event-based data (Figure 1.), where a signal is only produced when changes are observed instead of the traditional frame-based signals. SNNs have not been widely used despite the advantages due to a lack of training methods. The discrete nature of SNN outputs means that we cannot directly apply backpropagation without estimating the gradient [9]. To counter this issue, methods have been proposed to convert trained ANNs to SNNs. Conversion allows us to exploit the training methods of deep learning. One main issue with this approach is conversion loss. Not all layers can be directly converted to SNNs. An approximate conversion often leads to too many or too few spikes produced in the final SNN, resulting in a loss of accuracy and efficiency[8]. In this project, we converted a deep CNN, VGG16, to SNN. Through this comparison, we evaluate the effect of different layers on the SNN performance. Specifically, we explored the previously proposed implementation of converting ReLU, max-pooling, softmax, and batch normalization and attempted to analyze different combinations of SNN layers systematically.

In section 3, we introduce the concept of event-based data and the theory of the conversion from ANNs to SNNs. Finally, our experiment results are reported in section 4.

1.1. Difference from Proposal

We mostly followed the tasks outlined in our proposal. We compared the effect of converting different components in a CNN and tested our converted models on both frame-based and event-based datasets. However, we were not able to train an SNN from scratch to compare with the converted model on event-based data. We were also unable to find a meaningful way to measure energy efficiency.

2. Related Works

There have been two main approaches in SNN literature: SNN training and SNN conversion. For a recent review on SNN training, see [11] and references therein. But due to the non-differentiable nature of SNNs, it is difficult to directly train deep SNN models.

Since ANN-to-SNN conversion has been proposed [7], there have been plenty of attempts to losslessly transfer the success of deep ANN models to SNN performance.

[2] proposed to convert the ReLU activation into spiking models and converted a shallow convolutional neural network on the CIFAR-10 dataset. [4] suggests that the reset-by-subtraction can be a proper dynamics of the membrane potential for the purpose of conversion. We took this approach in this project.

[10] proposes conversion methods for various deep learning components based on the normalization of weights of

the base ANN models. We adapted the conversion of max pooling, batch normalization, and softmax from [10]

[3] keeps the weights of the base ANNs and normalizes the threshold of spiking neurons. And they theoretically prove that replacing ReLU activations in ANNs with threshold ReLU can improve the performance of the SNN conversion. But they only converted simple models.

The framework of our project is based on [3] in that we normalize the thresholds of spiking neurons and use the threshold ReLU in ANN models. Furthermore, we combine the conversion methods of [10] with the threshold normalization and conduct simulations on both static and event-based datasets.

The SNN solutions aim to develop energy-efficient models that can handle noiseless temporal data. One example of noiseless temporal data is information gathered from Dynamic Vision Sensor (DVS) cameras and SNNs research that focuses on analyzing movement since these neural networks are suitable to model temporal data. For instance, SpikeFlowNet [5] is a deep hybrid neural network architecture that integrates SNNs and ANNs for optical flow estimation from DVS cameras without sacrificing performance.

3. Methods

3.1. Conversion Methods

3.1.1 ReLU

ReLU is at the center of ANN to SNN conversion. [2] proposed direct conversion of ReLU to spiking neurons based on the close relationship between activation of ReLU and firing frequency of converted neuron. If we assume a 1-to-1 relationship between ANN and SNN units, we can directly use the weights learned by the ANN. However, the unbounded property of ReLU means that the converted SNN units might produce too many or too few spikes. This issue has been countered through the normalization of weights [10]. In this project, we employed the conversion method proposed in [3]. According to this new method, the ReLU function in ANN is replaced by a threshold ReLU function, which includes an additional upper bound for activation values. Activation between 0 and the threshold would not be modified, while values larger than the threshold would be set to the threshold. During conversion, the threshold voltage of spiking neurons is set to the maximum activation of the corresponding layers. This effectively limits the maximum activation and brings the continuous ReLU activation closer to the discrete SNN step activation. By bounding the ReLU function, this conversion method removes the need for weight normalization. In order to bring threshold ReLU even closer to spiking neurons, the authors also added a shift according to the simulation length in SNN. Simulation length is a hyperparameter in SNN and refers to the number of times an input would be passed through the SNN until reaching the final inference.

3.1.2 Batch Normalization

Since batch normalization is a linear process, we can directly apply batch normalization with learned mean and variance[10]. Although batch normalization is not converted, we still included it in our experiment to determine its effect in a deeper network. We hypothesize that reducing variation in data through batch normalization would reduce conversion loss in a deeper network.

3.1.3 Max-Pooling

Max-pooling has been one of the main limitations in CNN to SNN conversion [8]. Since different neurons in SNN layers might fire at different time steps, we can not directly compute the maximums. Previously, converted SNNs have to try to prevent this problem by using average pooling instead. While average pooling can be directly computed, using average pooling reduces the performance of trained ANN. In our project, we use the simple spiking max-pooling proposed in [10]. The spiking max-pooling layer would estimate the firing frequency of each neuron by keeping the total number of spikes each neuron has generated through time. When pooling, the neurons with maximum firing frequencies would be selected.

3.1.4 Softmax

While SNNs output can be directly interpreted from the number of spikes each output neuron generates, this approach is unstable since some neurons might never spike. This issue is more likely to occur in deeper networks as spikes become sparser deeper in the network. Similar to max-pooling, our conversion method of softmax is also based on [10]. However, unlike normal softmax in ANN, spiking softmax is not just a direct normalization of class likelihoods. During inference, the spiking softmax would calculate a normal softmax base on the membrane potential of each neuron. The probabilities from softmax are then used to parameterize a Poisson process for each neuron. These Poisson processes then determine if a spike should be generated. In actual implementation, a random uniform distribution between 0 and 1 is used to determine whether a neuron should fire by comparing the generated number to the softmax probabilities.

3.2. Event-Based Datasets

In order to analyze further the capabilities of SNNs, we need to test them on event-based data. Event-based data comes from cameras containing dynamic vision sensors that capture motion. The event-based data contains information on positive and negative intensity changes in time that could be visualized in a two-channel image.

Since the popular datasets in computer vision come from standard cameras, the community has explored methods to

test SNNs by converting popular datasets to event-based ones by directly using DVS cameras and recording the repeated closed-loop movement of frame-based images. In the project, the dataset used was N-MNIST[6], the event-based version of these popular datasets created using the second conversion method.

Figure 1. N-MNIST 2-channel image visualization [6]

4. Experiments

4.1. MNIST

The goal of the first experiment was to test the correctness of our conversion by comparing the accuracies between the original model and the converted neural network. As a result, the average difference between models is less than 3%, which indicates a correct implementation, although it is not a lossless conversion. Moreover, the threshold ReLU stabilizes conversion, a method already tested in [3].

Model	ANN	SNN
ReLU	97.68%	95.21%
thReLU	97.65%	95.82%
thReLU + Maxpool	93.9%	91.44%
thReLU + Maxpool + Softmax	95.1%	92.2%

Table 1. Accuracy for ANN and converted SNN on MNIST

We also experimented with varying the threshold to control the number of spikes in the layer activations. For MNIST, a higher threshold improves accuracy and decreases conversion loss, meaning that we need to limit the number of spikes for this data.

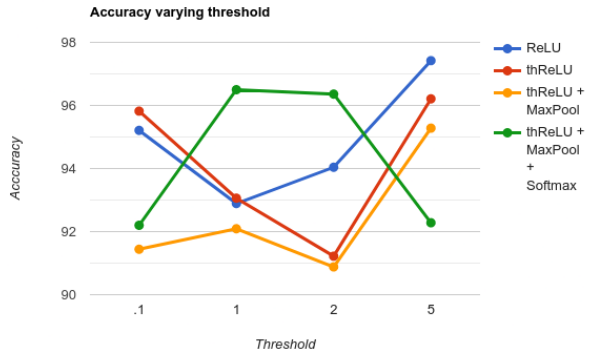

Figure 2. Accuracy change when varying the threshold on MNIST

4.2. N-MNIST

To further analyze the capabilities of SNN, we tested the converted model with the event-based dataset [6]. Compared with the experiment on the original MNIST dataset, the difference in accuracy after conversion was significantly higher, meaning that the hyperparameter choices did not favor the extraction of temporal information from the data. However, by adding more spiking layers, this conversion loss decreases, meaning that spiking layers can handle the sparsity of the inputs.

Model	ANN	SNN
ReLU	98.58%	90%
thReLU	93.52%	87.72%
thReLU + Maxpool	93.83%	89.27%
thReLU + Maxpool + Softmax	93.83%	91.97%

Table 2. Accuracy for ANN and converted SNN on N-MNIST

Experiments to compare performance by varying the threshold were done in this dataset. Compared to the MNIST, N-MNIST decreases performance when a higher threshold is used for every model except ReLU. Once again, this result tells that further experiments need to be done by varying the simulation length of the data and searching for an adequate way to feed the temporal data in training and testing.

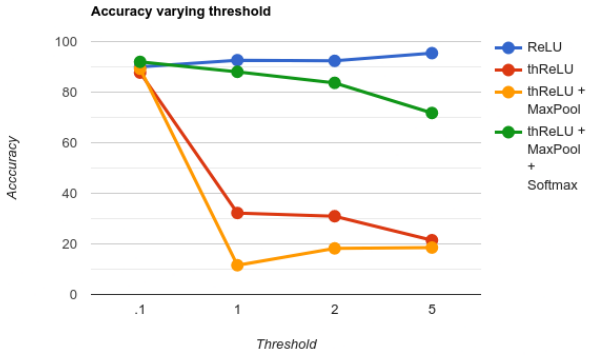

Figure 3. Accuracy change when varying the threshold on N-MNIST

4.3. VGG16

To investigate the performance of conversion methods on deeper models, we convert various VGG models on the CIFAR-10 dataset to SNN models. For example, [3] converts VGG with average pooling using the threshold normalization methods. However, the authors did not report the conversion loss of variations of VGG models, such as max-pooling and batch normalizations. They suggested those extensions would “make the framework more practical.”

In addition to VGG with average pooling (denoted as **AVG**), we consider three more possible extensions of VGG

models. **MAX** replaces average pooling layers with max-pooling layers. **AVGBN** and **MAXBN** are constructed by inserting batch normalization layers after all convolutional layers and dense layers in **AVG** and **MAX**, respectively. We choose threshold ReLU as the activation of ANNs, and the thresholds are set $y_{th} = 1$ on models without batch normalization, $y_{th} = 3$ on models with batch normalization.

In this part, we concentrate on the conversion loss of various VGG models rather than improving the accuracy of SNN models. Therefore, we train baseline ANN models using ADAM with a constant learning rate for 20 epochs instead of sophisticated optimization methods. The experiment is conducted three times, and we report the average top-1 accuracy.

The result is summarized in Table 3. A longer simulation time (denoted as T) reduces the conversion loss. It appears that the batch normalization tends to reduce the conversion loss, except for **MAX** at $T = 128$. The batch normalization substantially decreases the conversion loss of **AVG**. On the other hand, it is unclear how max-pooling in the place of average pooling affects the conversion loss.

Model	**AVG**	**MAX**	**AVGBN**	**MAXBN**
ANN	73.20 %	74.03 %	75.91 %	71.89 %
SNN, T=128	62.21 %	69.91 %	74.61 %	67.36 %
Loss, T=128	10.99 %	4.12 %	1.3 %	4.53 %
SNN, T=256	70.33 %	71.93 %	75.76 %	70.08 %
Loss, T=256	2.87 %	2.1 %	0.15 %	1.81 %

Table 3. Conversion loss of various VGG16 models

5. Conclusion

In this project, we successfully converted four different CNN layers: ReLU, Batch Normalization, Max-Pooling, and SoftMax. Threshold ReLU and softmax showed consistently low conversion loss throughout experiments. However, conversion loss becomes more unstable when paired with max/average pooling and batch normalization. It was, therefore, hard to conclude the effect of both layers. As expected, using spiking max-pooling instead of average pooling increases converted models' performance. But this relationship becomes reversed when batch normalization is also included. One major shortcoming of our experiment is that we could not train our VGG models to the best performance on CIFAR-10 due to limited computing resources. Comparison of different SNN structures might be more meaningful if we were able to train all models more.

Our experiment on an event-based dataset has demonstrated that even very simple converted SNNs can be effective. Although the loss of direct conversion from ReLU is significant, converting additional layers and applying threshold ReLU effectively lowered conversion loss and increased performance. One important aspect of event-based data that we did not cover in our experiment is the temporal information. In our experiments, we used preprocessed frame images of events instead of the original binary data. This means that we cannot fully exploit the temporal relationships and were not able to experiment with SNNs' ability to maintain some memory of past events through membrane potential. One potential future direction to better explore event-based data is to employ a model structure similar to LSTM and convert it to a spiking model.

We would like to explore methods further to reduce conversion loss for future works. One traditional solution to reduce conversion loss is to normalize the weights and biases on the regularization layer, which has shown to be effective under too high or low spike rate[8]. We can examine the effect of combining weight normalization and threshold ReLU. In addition, our current converted model only contains excitatory neurons with positive thresholds, which means that the model would lose the ability to propagate negative activation. It would be interesting to explore ways to include both excitatory and inhibitory neurons to mimic biological networks more closely. Finally, it would be interesting to explore the effect of skip connections in converted deep SNNs. Similar to their effects in deep neural networks such as ResNet, skip connections might mitigate the problem of too few spikes and improve inference time and accuracy in converted deep SNN.

References

[1] Michele Barbi, Santi Chillemi, Angelo Di Garbo, and Lara Reale. Stochastic resonance in a sinusoidally forced lif model with noisy threshold. *Bio Systems*, 71:23–28, 2003. 1

[2] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015. 1, 2

[3] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. 2021. 1, 2, 3

[4] Peter U Diehl, Bruno U Pedroni, Andrew Cassidy, Paul Merolla, Emre Neftci, and Guido Zarrella. Truehappiness: Neuromorphic emotion recognition on truenorth. In *2016 international joint conference on neural networks (ijcnn)*, pages 4278–4285. IEEE, 2016. 1

[5] Chankyu Lee, Adarsh Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. Spike-flownet: Event-based optical flow estimation with energy-efficient hybrid neural networks. *CoRR*, abs/2003.06696, 2020. 2

[6] Garrick Orchard, Ajinkya Jayawant, Gregory Cohen, and Nitish V. Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *CoRR*, abs/1507.07629, 2015. 3

[7] José Antonio Pérez-Carrasco, Bo Zhao, Carmen Serrano, Begona Acha, Teresa Serrano-Gotarredona, Shouchun Chen, and Bernabé Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2706–2719, 2013. 1

[8] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12, 2018. 1, 2, 4

[9] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer. Theory and tools for the conversion of analog to spiking convolutional neural networks, 2016. 1

[10] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017. 1, 2

[11] Xiangwen Wang, Xianghong Lin, and Xiaochao Dang. Supervised learning in spiking neural networks: A review of algorithms and evaluations. *Neural Networks*, 125:258–280, 2020. 1

Appendix

Team contributions

Jorge Chang SpikingReLU, thReLU, MNIST and N-MNIST event-based training/testing, and experiments

Yipu Gao Max pooling, softmax, CNN base model training and testing for CIFAR-10, and experiments.

Heejun Lee SpikingBNReLU, VGG16 for CIFAR-10 training/testing, and experiments

Peisen Zhou Topic research and experiment design. SpikingReLU, thReLU, max pooling, VGG16 experiments.