Conversion of Analog to Spiking Transformer Networks

Viktor Studenyak

Advisor: Etienne Müller

March 2021

1 Introduction

Transformer Networks revolutionized the field of natural language processing. They are able to model temporal sequences very well, achieving remarkable performance in machine language translation Vaswani et al. (2017). Transformers adapted for computer vision operating on image patches have shown excellent performance on many datasets, at the same time being less computationally expensive Dosovitskiy et al. (2020). On the other hand, recent developments in neuromorphic hardware give an opportunity for energy-efficient computations Merolla et al. (2014). The limitation is, that neuromorphic hardware is well suited for spiking neural networks, but not for more conventional artificial neural networks. Conversion of trained ANNs into SNNs yielded best results Pfeiffer and Pfeil (2018) and serve as a benchmark for SNN performance for fully-connected and convolutional neural networks Tavanaei et al. (2019).

In this work, we introduce the spiking architecture for self-attention-based Transformer networks obtained through weight conversion using weight normalization tools proposed by Rueckauer et al. (2016).

2 Related Works

There are two major approaches for training networks of spiking neurons: unsupervised learning using spike-timing-dependent plasticity Diehl and Cook (2015) and supervised learning using backpropagation Lee et al. (2016). Other works on supervised learning with backpropagation are Mostafa (2017), Neftci et al. (2017), Shrestha and Orchard (2018), Neftci et al. (2019), which propose techniques for overcoming continuous-value nature of backpropagation algorithm.

On the other hand, conversion methods show superior performance in comparison with the previously described methods. By utilizing the non-negativity property of ReLU activation functions, the average neuron firing rate can be approximated Cao et al. (2014). This method was neglecting bias and was not particularly good for activations with comparably greater values, than the weights. In subsequent work, Diehl et al. (2015) proposed a weight normalization algorithm, which involved activations of neurons for scaling factor estimation. Rueckauer et al. (2016) proposed a way to make weight normalization more robust to very large outliers to avoid simulation latency in the network of spiking neurons. In Rueckauer et al. (2017) methods for max-pooling and batch normalization layer conversion were proposed. More recent methods proposed architectures for the conversion of more complex architectures, such as deep residual networks Hu et al. (2020), Sengupta et al. (2019) and YOLO-architecture Kim et al. (2019). However, conversion of transformer networks Vaswani et al. (2017) was missing.

In Rathi et al. (2020) even went further and combined the ann to snn weight

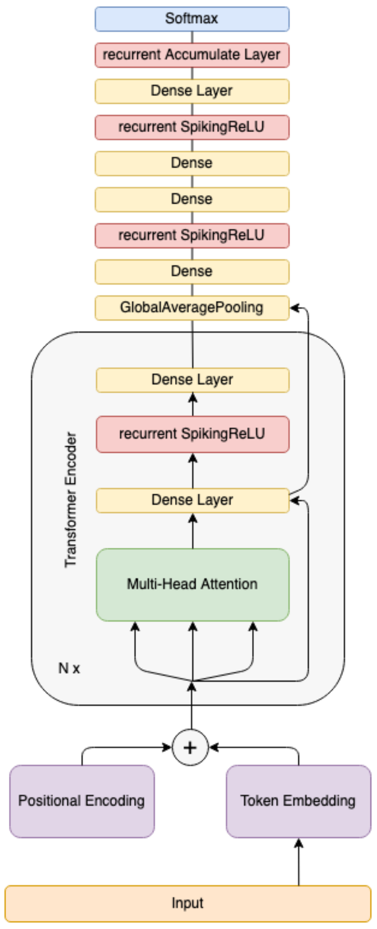

Figure 1: Spiking Transformer architecture

conversion as initialization phase with the spike-based backpropagation and achieved superior results in terms of simulation latency and accuracy compared to ann-snn conversion methods or conventional spiking training methods.

Transformer architecture was proposed by Vaswani et al. (2017) and introduced the usage of multi-head-self-attention, a mechanism for performing queries on a sequence of information. This method performed much better compared to conventional RNN- and LSTM-architectures. In Dosovitskiy et al. (2020) architecture for the computer vision classification task using multi-head-self-attention mechanism was proposed. In this work, image patches were extracted and a sequence of patches was formed. The sequence was complemented by the positional encoding and the class embedding and then fed into transformer encoder mechanism. After that multi-layer perceptron was added on the top of the architecture, which was able to perform computer vision classification tasks. This architecture also achieved state-of-the-art results.

In this work, we propose spiking transformer architecture, solving two problems: natural language processing in form of sentiment analysis and image classification for written digit recognition.

3 Methodology

Architecture. Network architectures with dense layers with ReLU activation functions are trained. In the spiking architecture, those layers are converted to dense layers without activation in combination with a recurrent layer with SpikingReLU as recurrent cells. In this combination, layers simulate the behavior of fire-integrate neurons. A dense layer with softmax activation at the tail of the network is converted to a dense layer in combination with a recurrent layer with Accumulate as recurrent cells. Accumulate cells are aggregating values over the simulation timesteps. SpikingReLUs use the reset by subtraction method to reset membrane potential, it has shown better efficiency for conversion of networks of spiking neurons Rueckauer et al. (2017). The overall architecture for the NLP Transformer network is shown in Figure 1.

Weight normalization. For weight normalization, we use the robust data-based normalization method proposed by Rueckauer et al. (2016). We take a subset of a test set and compute activations for the layers that have to be converted (layers with ReLU activation function) and get p-th percentile value of the activity distribution. This way we can ensure, that the outliers in the activation distribution do not increase the latency of the converted network. First, we compare maximal values of bias and weight for this layer and pick a greater value. Then, we compare this value with a p-the percentile value of the activation distribution and choose a greater value. This value is then rescaled by the conversion factor of the previous converted layer. Finally, we rescale weights of the layer using the previously computed scaling factor. We also rescale bias with respect to the current scaling factor, not discounting it with the scaling factor of the previous layer. Rescaled values are then assigned to the corresponding layer of the spiking neural network in the conversion pipeline.

After converting weights and assigning them to the spiking neural network, we simulate this network on the dataset and evaluate it's accuracy over the simulation timesteps.

4 Experiments

Datasets. We train and evaluate NLP Transformer Network on IMDB movie review sentiment classification dataset Maas et al. (2011). The dataset includes 25.000 reviews from IMDB, each review has one label, positive or negative sentiment. Dataset is split into two halves, one half is used for training, another one for testing.

For the first experiment, we use a vocabulary size of 20.000, which means that only the top 20.000 words will be considered. We also crop reviews to a maximal length of 200 words, reviews that have fewer words will be filled with zero paddings. We transform labels to categorical vectors, which can have two states: positive or negative. We do not represent reviews as one-hot encoded vectors but as sequences of encoded words. We use embedding layers for the word sequences and also for encoding positions of the words. Then, we add token embeddings to positional embeddings, which then are fed to the multi-head self-attention module.

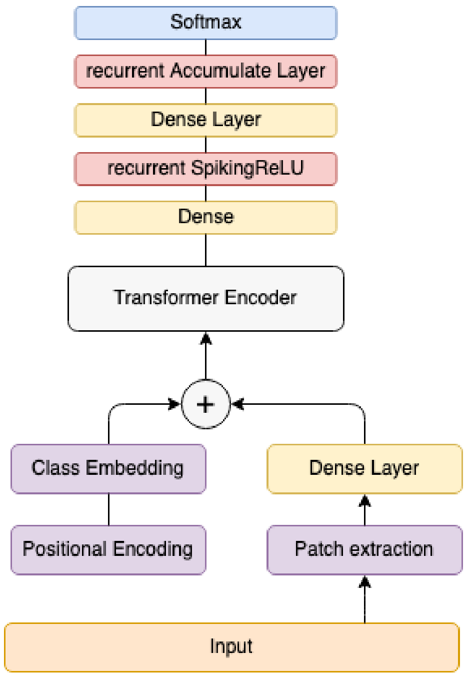

Figure 2: Spiking Vision Transformer architecture

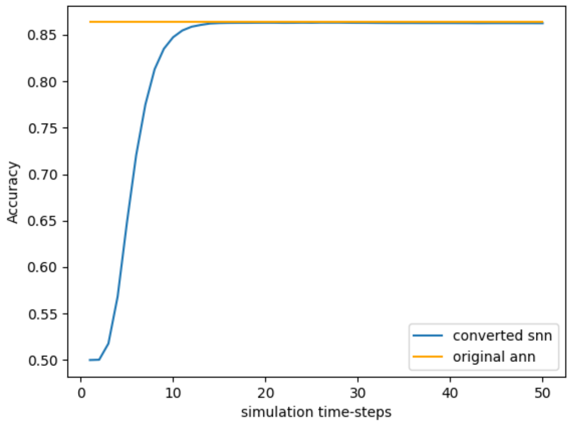

Figure 3: Simulation accuracy of NLP transformer

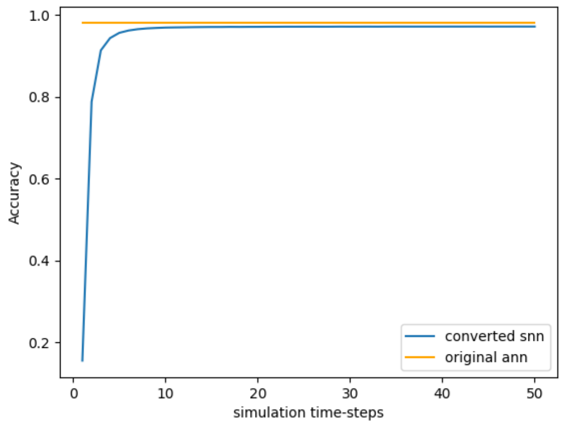

Figure 4: Simulation accuracy of vision transformer

In our second experiment, we use a vocabulary size of 500 and a maximal length of reviews of 50. Similar to the first experiment, we crop reviews to maximal length and fill reviews with zeros, if they don't have enough words. We represent words as one-hot encoded vectors.

Our vision Transformer Network we train and evaluate on MNIST dataset. It consists of 60.000 images for the training and 10.000 for the testing phase. For vision transformer, we use architecture proposed by Dosovitskiy et al. (2020).

Experiment 1. NLP Transformer. We have built a network similarly to Vaswani et al. (2017) illustrated by Figure 1. For details please refer to 1. We use embedding over a range of numbers to get positional encoding. We add positional encoding to the token embedding of the input. The transformer encoder part is equivalent to the original transformer, except the dense layer with ReLU as an activation function. We exchange it as previously mentioned with a dense layer without any activation function and a recurrent layer with SpikingReLU as a recurrent cell. After the transformer encoder part, we use the global average pooling layer. Subsequently, we exchange every dense layer with activation function by the dense layer without activation function followed by recurrent layer. We also exchange the final layer with the softmax activation function by recurrent layer with accumulation cell followed by softmax operation.

Experiment 2. Vision Transformer. For this experiment we have built a network similarly to Dosovitskiy et al. (2020). The exact architecture is shown in Figure 2. We divide every image into four patches, each patch consists of 7x7 pixels. We then flatten the patch and make a linear projection of all patches. We add positional encoding together with class embedding to linear projections of the patches. The transformer encoder part of the network is very similar to the encoder part of the NLP transformer. We also flatten the output of the transformer encoder instead of picking the zeroth dimension, which is different from the original vision transformer.

For both experiments, we train the ANN network 50 times and then simulate each ANN for 50 timesteps to average out the results for different random seeds. For weight conversion we use p-th percentile of 0.99 for the activation distribution, which is a lower boundary for the suggested range of [0.99, 0.999] proposed by Rueckauer et al. (2016). For the NLP transformer, we train the network for 2 epochs with a batch size of 64 and achieve the averaged accuracy of the ANN network of 86.36 percent. On a converted network after 50 simulation steps, we achieve an accuracy of 86.25 percent, which yields a conversion loss of 0.11 percent. On the other hand, we achieve an accuracy of 86.1 percent after only 13 timesteps. For the vision transformer, we train the network for 2 epochs with a batch size of 64 and achieve an averaged accuracy of an ANN network of 97.99 percent. Although we achieve an average result of 97.16 after 50 timesteps on a converted network, we achieve a result of 97.02 only after 13 timesteps. Conversion loss is 0.83 percent for the converted network after 50 timesteps.

5 Discussion

In this work, we introduced the novel conversion of transformer architectures from deep artificial networks to networks of spiking neurons. We obtained architectures with near loss-less conversion: 0.11 percent for NLP transformer and 0.83 percent for vision transformer.

The difference in the conversion loss of NLP and vision transformers can be observed. This difference is likely due to the different mechanisms utilized for the data transformation in the first part of the architecture. Ablation studies should be performed to further analyze two architectures.

For future work, several directions can be chosen. On the one hand, the existing architectures of ann-snn conversion can be further explored, e.g. run on more datasets, apply to more advanced datasets, perform a hyperparameter search or try out the architecture on the neuromorphic hardware. Further ablation studies can be performed e.g. conversion with different p-th percentiles of the activity distribution, training with different number of heads in multi-head-self-attention. On the other hand, it could be interesting to apply similar techniques to more recent and advanced types of transformers such as Choromanski et al. (2021) or Wang et al. (2020), which try to reduce the space and time complexity of self-attention mechanism. Another very interesting direction is to explore the hybrid learning approach proposed by Rathi et al. (2020) for the transformer networks. In hybrid learning approach authors proposed to combine weight conversion with spike timing dependent backpropagation. For future research it could be of a great interest to combine also other methods with the weight conversion i.e. surrogate gradient method Neftci et al. (2019). It would be also interesting to combine weight conversion as a pretraining step with the unsupervised STDP algorithm.

References

Cao, Y., Chen, Y., and Khosla, D. (2014). Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113:54–66.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., and Weller, A. (2021). Rethinking attention with performers.

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

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. (2015). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale.

Hu, Y., Tang, H., and Pan, G. (2020). Spiking deep residual network.

Kim, S., Park, S., Na, B., and Yoon, S. (2019). Spiking-yolo: Spiking neural network for energy-efficient object detection.

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10:508.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. pages 142–150.

Merolla, P., Arthur, J., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B., Imam, N., Guo, C., Nakamura, Y., Brezzo, B., Vo, I., Esser, S. K., Appuswamy, R., Taba, B., Amir, A., Flickner, M. D., Risk, W., Manohar, R., and Modha, D. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345:668 – 673.

Mostafa, H. (2017). Supervised learning based on temporal coding in spiking neural networks.

Neftci, E. O., Augustine, C., Paul, S., and Detorakis, G. (2017). Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in Neuroscience*, 11:324.

Neftci, E. O., Mostafa, H., and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks.

Pfeiffer, M. and Pfeil, T. (2018). Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12:774.

Rathi, N., Srinivasan, G., Panda, P., and Roy, K. (2020). Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation.

Rueckauer, B., Lungu, I.-A., Hu, Y., and Pfeiffer, M. (2016). Theory and tools for the conversion of analog to spiking convolutional neural networks.

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11:682.

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. (2019). Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, 13:95.

Shrestha, S. B. and Orchard, G. (2018). SLAYER: spike layer error reassignment in time. *CoRR*, abs/1810.08646.

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks*, 111:47–63.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity.