

Optimizing Implementation of SNN for Embedded System

Hyeonguk Jang, Jae-Jin Lee, Kyuseung Han

Electronics and Telecommunications Research Institute, Daejeon, 34129, Korea

lemon@etri.re.kr, ceicarus@etri.re.kr, han@etri.re.kr

Abstract— Spiking neural networks (SNNs) are a highly promising AI technology for embedded systems, owing to their energy-efficient properties. However, the manual implementation of SNNs encounters practical challenges because of the all-to-all connections in large networks. Thus, this paper presents a novel methodology to reduce wire congestion in the SNN implementations while mitigating adverse effects on inference accuracy.

Keywords— back propagation through time, deep learning, embedded system, hardware, spiking neural networks

I. INTRODUCTION

Embedded systems are special computing systems used in a wide range of applications, from home appliances such as smartphones and IoT devices to industrial control systems, automotive systems, and medical devices [1]. Unlike existing computers, it has strong constraints on power consumption, area, and real-time processing, while performing multiple tasks or functions within a specific application or domain. Therefore, it is important to optimize both hardware and software to meet the stated requirements rather than including general-purpose features.

The recent advancements in AI have spurred a growing demand for AI capabilities in embedded systems. Spiking neural networks (SNNs) offer a distinct advantage over other AI technologies, primarily due to their low-power operation. Learning is done on an external powerful server, and the typical configuration for embedded systems is to focus on the inference function of the target application [2].

One of the major challenges in using SNNs in the embedded systems is that SNNs require complex circuits that must be implemented manually. SNNs consume low power by operating only when an event occurs. This feature can be realized when the circuits are implemented using analog or asynchronous digital techniques [3], which cannot be automated. Thus, in this paper, we will propose a methodology to alleviate this difficulty.

II. IMPLEMENTATION OF SNN

SNN is a neural network that mimics spike-based operations of a biological brain, which is show in Figure 1. Its architecture mainly consists of two types of layers,

convolution layers and fully-connected layers (FCL). Among these, we will focus on the FCL in this paper. FCL is literally a layer in which the neurons in the current layer and the neurons in the next layer are all-to-all connected as shown in Figure 2.

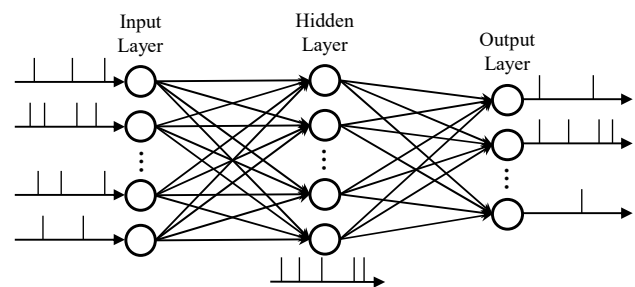


Figure 1. Spiking Neural Network

It incurs three problems when implementing FCL. First, FCL necessitates the substantial number of wires to connect between nodes. When implemented on a chip, the complexity of wire connections increases, the area expands, and the number of metal layers increases. And wire delay becomes longer and scalability decreases.

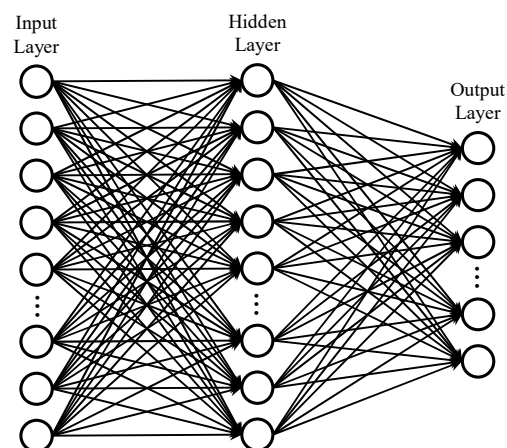


Figure 2. Fully-connected layers

Second, the first problem becomes more serious in that SNNs require manual implementation. For low power, SNN must operate only when an event occurs, and the current way

to implement SNN is asynchronous digital circuit design or analog circuit design. Both of these ways must be designed manually and cannot be implemented through HDL-level automation. Therefore, implementing FCL takes a lot of time and effort.

Lastly, if FCL is designed to have connections with different delays among them for simplifying implementation, the learning will become much more challenging, or additional hardware will be required. A representative example is mesh structure. Ideally, spikes fired from neurons in the current layer should reach neurons in the next layer at the same time. However, in mesh structure, the arrival times are not the same. There are two ways to solve this problem. The first approach is to learn SNN as a learning method that takes into account the timing characteristics of the hardware. This approach is not only difficult to learn, but also has the problem of the software framework becoming diverse. The second approach is to synchronize the spike arrival times by introducing a hardware component, such as a global synchronizer. This approach has a high level of difficulty in coordinating signals between asynchronous and synchronous logics. And because this approach is centralized, it creates additional scalability issues.

III. PROPOSED ADJACENTLY-CONNECTED LAYER

To reduce the number of connections between neurons, we introduce a structure where only adjacent neurons are connected, as illustrated in Figure 3. We call this structure an adjacently-connected layer (ACL). Because significantly reduced connections could potentially affect inference accuracy, we will investigate this effect in the experiments section. Additionally, the connection method is explained as follows.

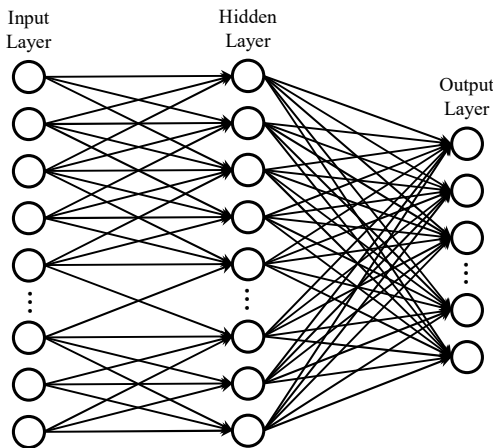


Figure 3. Proposed adjacently-connected layers

The proposed adjacent-connection restricts the number of neurons in the next layer that can be connected to a neuron in the current layer to a fixed count N . A method for determining which neurons in the next layer should be connected to neurons in the current layer is as follows. First, we calculate the relative position within the layer for a neuron in the current layer. Then we look for the most adjacent neuron

whose the relative position within the next layer is most similar to the relative position of the neuron within the current layer. Lastly, we select a total of N neurons with the most adjacent neuron and $N/2$ neurons above the adjacent neuron and $N/2-1$ neurons below the adjacent neuron. Here, if the number of neurons above is less than $N/2$ or the number of neurons below is less than $N/2-1$, the number of selected neurons is less than N . Ultimately, we can construct ACL by applying the method to all neurons in the current layer.

This approach is different from conventional pruning techniques [4]-[6]. Pruning is a method of lightening the model by training it not to use some of the connections, although they are implemented. Therefore, pruning is not a technique that affects implementation. Since the required connections change each time we train, the implementation of all-to-all connection is ultimately required.

IV. EXPERIMENTS

A. Experimental Environments

For the experiments, we performed the simulation of SNNs using the `snnTorch` library based on the PyTorch deep learning library. MNIST [7] and Fashion-MNIST [8], which are used for image classification, were used as datasets for the experiment. The network structure used for classification of the two datasets consists of input, hidden, and output layers, and the number of neurons in each layer is 784, 300, and 10, respectively. The integration and fire (I&F) neuron model without bias was used as the neuron model of SNN.

B. Learning of SNNs with Adjacently-connected Layers

For SNN learning, we used the Back Propagation Through Time (BPTT) [9] algorithm. To control synaptic connections, we utilized the custom mask in the pruning function provided by PyTorch. The custom mask enables the establishment of connections between adjacent neurons, while the weights of the unconnected synapses are set to 0, ensuring that only the weights of the connected synapses are trained.

C. Experimental Results

In this experiment, the accuracy of each MNIST dataset and Fashion-MNIST dataset was measured for the cases where the number of neurons in the hidden layer to which a neuron in the input layer is connected is all-to-all, 300, 250, 200, 150, 100, 50, and 10.

Figure 4 presents the results of the experiment on the MNIST dataset. It shows a slight improvement in accuracy when using ACL instead of FCL in the structure of the network. And the improved accuracy is maintained from 300 to 50 connections. And we confirmed that when the number of connections is less than 100, the accuracy decreases as the number of connections decreases. Considering the results, it is optimal to use ACL with 100 connections for the MNIST dataset.

The experimental result on the MNIST dataset showed that the accuracy when using adjacent-connection is slightly higher than that when using all-to-all connection. This phenomenon may appear somewhat unusual, but it may come from the

insufficient performance of the learning framework. The framework employs a heuristic that does not search the entire solution space, but rather focuses on finding local optimum. Therefore, reducing the solution space by using ACL can yield better results, while further reduction ultimately led to worse results. If there was an ideal algorithm, all-to-all connection would always provide higher accuracy.

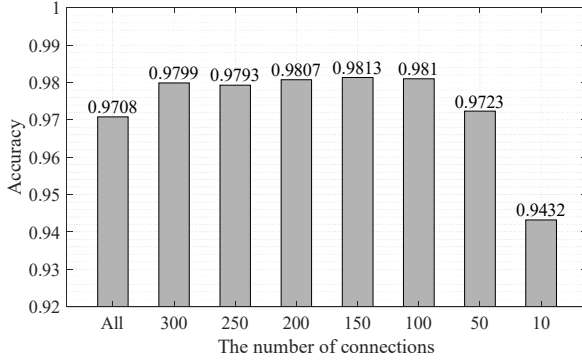


Figure 4. Inference accuracy on MNIST dataset

Figure 5 presents the results of the experiment on the Fashion-MNIST dataset. When compared to all-to-all connection, we can see that the accuracy is barely reduced until the number of connections is reduced by 200. From the results, in the case of Fashion-MNIST, the optimal number of adjacent-connection is 200 while maintaining accuracy.

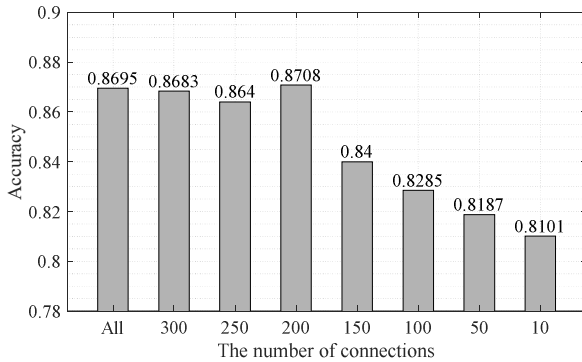


Figure 5. Inference accuracy on Fashion-MNIST dataset

As a result of comprehensive analysis of the experimental results for the two datasets, some of the SNNs using ACL have no loss of accuracy compared to all-to-all connection. And in some cases, we can find an optimal solution that reduces the number of connections while achieving better accuracy. Ultimately, we confirmed that applying ACL proposed in this paper can simplify and optimize the HW design without the loss of accuracy.

V. CONCLUSIONS

The manual layout of highly-congested wires complicates the implementation of SNNs. In order to mitigate the difficulties, we propose an adjacently-connected layer (ACL) optimized for embedded systems. By considering application

characteristics, ACL effectively reduces the number of wire connections while minimizing the loss of accuracy. In the future, we plan to utilize the 2D features of images for the connections in order to use fewer wires.

ACKNOWLEDGMENT

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-01294, Development of IoT based edge computing ultra-low power artificial intelligent processor).

REFERENCES

- [1] M. Sarrafzadeh, F. Dabiri, R. Jafari, T. Massey, and A. Nahapetan, "Low power light-weight embedded systems," in Proc. IEEE Int. Symp. Low Power Electron. Design (ISLPED), pp. 207–212, 2006.
- [2] Q. Liu and Z. Zhang, "Ultra-low power always-on intelligent and connected SNN-based system for multimedia IoT-enabled applications," IEEE Internet Things Journal, vol. 9, no. 17, pp. 15570–15577, Sep. 2022.
- [3] A. Javanshir, T. T. Nguyen, M. A. P. Mahmud and A. Z. Kouzani, "Advancements in Algorithms and Neuromorphic Hardware for Spiking Neural Networks," in Neural Computation, vol. 34, no. 6, pp. 1289–1328, May 19, 2022.
- [4] T. Hoefer, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks," Journal of Machine Learning Research, vol. 22, no. 241, pp. 1–124, Sep. 2021.
- [5] N. Rathi, P. Panda and K. Roy, "STDP-Based Pruning of Connections and Weight Quantization in Spiking Neural Networks for Energy-Efficient Recognition," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 4, pp. 668–677, Apr. 2019.
- [6] C. J. Schaefer, P. Taheri, M. Horeni and S. Joshi, "The Hardware Impact of Quantization and Pruning for Weights in Spiking Neural Networks," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 70, no. 5, pp. 1789–1793, May 2023.
- [7] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the Web]," IEEE Signal Process. Mag., vol. 29, no. 6, pp. 141–142, Nov. 2012.
- [8] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," arXiv:1708.07747, 2017.
- [9] J. K. Eshraghian et al., "Training Spiking Neural Networks Using Lessons from Deep Learning," in Proceedings of the IEEE, vol. 111, no. 9, pp. 1016–1054, Sept. 2023.

Hyeonuk Jang received his B.S. and M.S. degrees in electrical engineering from Gyeongsang National University, Jinju, South Korea, in 2013 and 2015, respectively, and his Ph.D. degree in ICT from the University of Science and Technology, Daejeon, South Korea in 2021. He is currently working at Electronics and Telecommunications Research Institute, Daejeon, South Korea, as a post doctor. His interests include spiking neural network, deep neural network and design of RISC-V based low-power embedded systems.

Jae-Jin Lee received the B.S., M.S., and Ph.D. degrees in computer engineering from Chungbuk National University, Cheongju, South Korea, in 2000, 2003, and 2007, respectively. He is a leader of the AI Edge SoC Research Section, Electronics and Telecommunications Research Institute, Daejeon, South Korea. His research interests include spiking neural network and RISC-V based embedded systems.

Kyuseung Han (Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University (SNU), Seoul, South Korea, in 2008 and 2013, respectively. Since 2014, he has been with the Electronics and Telecommunications Research Institute, Daejeon, South Korea. His current research interests include reconfigurable architecture, network-on-chip and computer-aided design of low-power embedded systems.