Performance Analysis of Spiking Neural Network using Temporal Spike-based Backpropagation on Field Programmable Gate Array (FPGA) platform

Vijay Kakani, *Member, IEEE*
Integrated System Engineering
Inha University
Incheon, South Korea
vjkakani@inha.ac.kr

Sungwoo Lee
Electrical and Computer Engineering
Inha University
Incheon, South Korea
22202326@inha.edu

Xuenan Cui
Information and Communication
Inha University
Incheon, South Korea
xncui@inha.ac.kr

Hakil Kim*, *Member, IEEE*
Electrical and Computer Engineering
Inha University
Incheon, South Korea
hikim@inha.ac.kr

2022 IEEE Region 10 Symposium (TENSYMP) | 978-1-6654-6658-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/TENSYMP54529.2022.9864534

***Abstract*—This paper explores the object classification performance of spiking neural networks (SNNs) using the temporal spike-based backpropagation technique on the Field Programmable Gate Array (FPGA) platform. The FPGA board is specially designed to host the spiking neural networks for artificial intelligence tasks such as object classification, object detection, and segmentation. The publicly available classification datasets such as MNIST, CIFAR10 were employed to examine the performance of the SNNs on the FPGA platform. Similarly, the latest temporal spike-based backpropagation technique was chosen to investigate the neuromorphic ability of the low-cost FPGA board in processing SNNs for object classification tasks. The main purpose of this research proceeding is to facilitate the neuromorphic research community with the information regarding (i). the exploitation of the low-cost FPGA design for neuromorphic image processing and artificial intelligence (AI) tasks; (ii). cross-validating temporal spike-based backpropagation trained SNNs on FPGA alongside PC; (iii). assessing the performance stability and industrial choices of low-cost FPGAs for object classification tasks and related issues. The evaluation metrics such as classification accuracy, mean average precision, and processing time were utilized to assess the performance of the SNN model on FPGA alongside PC. This study will be used as an informative report for the researchers working towards perfecting the neuromorphic hardware for processing SNNs in imminent studies.**

***Index Terms*—Object Classification Performance, FPGA, Spiking Neural Networks, Neuromorphic Image Processing**

I. Introduction

The brain-inspired neuromorphic processing units have gained popularity with the advancement in the spiking neural networks (SNNs) [1]. The processing of spike-based algorithms for computing information has facilitated neuromorphic chips such as TrueNorth and Loihi designed by IBM and Intel respectively [2], [3]. These chips are highly energy-efficient and are specifically designed to deal with the spatiotemporal spike process. However, the implementation of the outstanding SNN models that can outperform the artificial neural networks (ANNs) remains as a challenge despite the advancements in neuromorphic chips [4]. The neuromorphic hardware units stated earlier are steep in the context of the cost and handling; making them hard to achieve in terms of affordability [5]. Especially for the applications such as Advanced driver-assistance systems (ADAS) and soft biometrics, where the infrastructure alone is expensive. The cost of these neuromorphic chips will decide major portion of the project's budget. This facilitated many researchers to pave their way towards employing spiking neural networks on low cost Field Programmable Gate Array (FPGA) boards [6]–[10]. In analogy, the growth in backpropagation (BP) techniques and loss functions in traditional deep neural networks (DNNs); numerous studies were put-forward to accomplish similar level of efficiency in employing backpropagation techniques for SNNs as well [11]–[20]. Although many studies exhibited reliable results, implementing the backpropagation technique that can work in conjunction with SNN still remains a significant problem. The complexity in the backpropagation of SNNs is primarily due to the fact that the spiking behavior is discrete and it involves both spatial and temporal aspects in terms of processing. Also, the non-differentiable characteristic of the spiking events makes the gradient calculation non-trivial and uncertain. Additionally, the spiking events impose heavy computation and severe latency

This work was supported by the Industrial Strategic Technology Development Program of MOTIE/KETI [10077445, Development of SoC technology based on Spiking Neural Cell for smart mobile and IoT Devices].

while employed on the deeper architectures. Nevertheless, certain methods such as surrogate gradient descent (SGD) [21] and temporal spike sequence learning (TSSL) [22] facilitates the process of training deeper SNNs much reliable while maintaining the SNN's accuracy in the same standard with ANNs. There were few investigative studies that were carried out by the authors on exploring surrogate gradient descent backpropagated SNNs and deployed on PC alongside NVIDIA TX2 embedded platforms [23]. Similarly, authors designed few deeper SNN architectures trained with surrogate gradient descent backpropagation were presented and deployed results were recorded on the embedded board [24]. The results of these prior studies [23], [24] assisted in making reliable decisions in deploying SNNs on embedded platform with SGD backpropagation technique; thereby facilitating the usage of those SNNs on embedded ADAS platforms. However, the feasible deployment of the temporal spike sequence learning backpropagation technique has not yet been investigated on the embedded platform especially on the low-cost FPGA board. Thus, this study explores the two fundamental attributes in a practical viewpoint: Firstly, the handling of spike-based backpropagation on the SNNs. Secondly, employing such SNNs on the low-cost FPGA board. The study was considered as an investigation on the "ability to deploy the temporal spike-based backpropagated SNNs on low-cost FPGA board and to assess the pros and cons". In practice, the comparison with the previously published works includes hurdles such as open-source availability and hardware adaptability w.r.t to the open-sourced models. Moreover, the famous spike-based backpropagation techniques were investigated in our prior studies [23], [24] on various platforms.

II. Spiking Neuron Model

The spiking neural networks utilize spiking neurons for information processing unlike artificial neural networks. Therefore, the spiking neural network used in this study is constructed with the adaptation of leaky integrate-and-fire (LIF) neuron model [1]. According to the LIF neuron model, the input spike train flows from presynaptic neuron y to the postsynaptic neuron x. The input spike train can be denoted by $I_y(t) = \sum_{f_y^{(t)}} \delta\left(f - f_y^{(t)}\right)$, where $f_y^{(t)}$ represents firing time of presynaptic neuron y. The postsynaptic current $A_y(t)$ is produced from the incoming spikes through the synaptic connection between y neuron and x neuron. The membrane potential $M_x(t)$ voltage for the postsynaptic neuron x at a given time t is represented by

$$\tau_m \frac{dM_x(t)}{dt} = -M_x(t) + R\sum_y W_{xy} A_y(t) + r_x(t), \qquad (1)$$

where R is the leaky resistance of the LIF neuron and τ_m is the membrane potential time constant, W_{xy} is the weight of the synaptic connection between presynaptic and post synaptic neuron, $A_y(t)$ is the postsynaptic current inculcated by the presynaptic neuron spike and $r_x(t)$ is the reset mechanism in the spiking activity. The postsynaptic current and the reset mechanism can be denoted as

$$\begin{aligned} A_y(t) &= (\alpha * I_y)(t) \\ r_x(t) &= (\beta * I_x)(t) \end{aligned} \qquad (2)$$

where $\alpha(.)$ and $\beta(.)$ are the response mechanism kernel and reset mechanism kernel respectively. Therefore, the initial spike reaction is indicated in combination with a synaptic time constant τ_s as:

$$\tau_s \frac{A_y(t)}{dt} = -A_y(t) + I_y(t). \qquad (3)$$

By employing the Euler process to (1) to simplify the membrane potential as

$$M_x[t] = \left(1 - \frac{1}{\tau_m}\right) M_x[t-1] + \sum_y W_{xy} A_y[t] + r_x[t]. \quad (4)$$

The total spiking action is then observed by the reset system to result in the yield of the spiking neuron as

$$I_x[t] = H(M_x(t) - V_E), \qquad (5)$$

where V_E is the spiking threshold and step function $H(.)$.

III. Temporal Spike-based Backpropagation

The spike-based backpropagation in general reduces the distance between the produced spikes (actual spikes) and the desired spikes through optimizing the loss function by minimizing the backpropagated error. This study adapted the loss function optimization presented in the temporal spike sequence learning study [22] as a backpropagation scheme. The other state-of-the-art backpropagation techniques were explored on the embedded platform in our prior studies [23], [24]. The temporal-spike backpropagation was claimed to be reliable in terms of reducing latency and maintaining lower time steps. The results presented in [22] only corresponds to the PC based environment and further analysis on embedded platforms has not yet been made. It is evident that the embedded NVIDIA TX2 might almost have the same performance capabilities in handling object classification tasks for video processing applications [23], [24]. The temporal spike sequence learning mechanism was chosen to assess the uncertainty in the results pertaining to the temporal spike-based backpropagated SNNs on FPGA. The temporal spike loss function proposed in [22] is formulated as the sum of squared error w.r.t each timestep for all neurons. This enable the calculation of distance between the desired spikes $d_{spike} = [d_{spike}|_{t=t_0}, d_{spike}|_{t=t_1}, ..., d_{spike}|_{t=t_{K_t}}]$ and produced (actual) spikes $s_{spike} = [s_{spike}|_{t=t_0}, s_{spike}|_{t=t_1}, ..., s_{spike}|_{t=t_{K_t}}]$ where $d_{spike}|_t$ and $s_{spike}|_t$ are the firing events both desired and

produced (actual) respectively at time t for output neurons with total time steps being K_t. The loss function is calculated as

$$L_{\text{temp.spike}} = \sum_{n=0}^{K_t} \xi\left[t_n\right]$$
$$\xi\left[t_n\right] = \sum_{n=0}^{K_t} \frac{1}{2}\left((A)\left[t_n\right] - (B)\left[t_n\right]\right)^2 \quad (6)$$

where $\xi\left[t_n\right]$ represents the error at time t and $\Delta(.)$ represents a Van Rossum distance function between $d_{spike}|_t$ and $s_{spike}|_t$. Also, $A = \Delta * d_{spike}$ and $B = \Delta * s_{spike}$.

IV. FPGA Design and Network Architecture

A. FPGA Schematic and Data Format

The FPGA board used in this work is shown in Fig. 1 is designed as a spiking neural network processor with Quadro RTX 4000 associated memory of 8.59 GB. The data processing on the FPGA board is enabled using the Advanced Peripheral Bus (APB) registers and SNN SW package made for the purpose of running SNNs on the board are shown in Fig. 2.

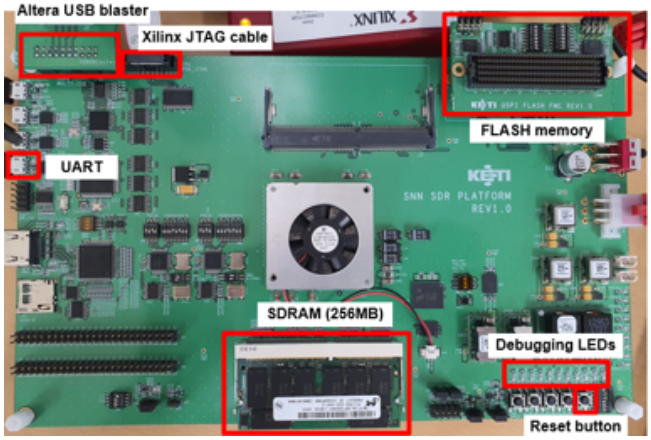

Fig. 1. FPGA board used in the study for deploying SNN.

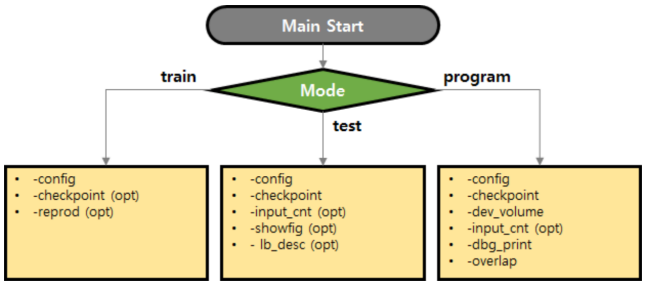

Fig. 2. Several functions associated with the FPGA SNN processor.

The board's memory assembly contain functionalities such as header, APB set, layer constraints and inputs which enable the handler to track the model over the .yaml configuration file in combination with choices that are shown in the upper half of Fig. 3 and the workings of data estimation on FPGA based w.r.t spiking aspect is shown in the lowest half of Fig. 3.

Fig. 3. SD card memory configuration with functionalities (top); Spike-based data prediction on FPGA (bottom).

B. SNN Architecture and Parameters

The SNN architecture used in this study consists of ten layers and parameters per each layer are displayed in Table I. with layer-1, layer-2, layer-4, layer-6 and layer-7 being the convolution layers with kernel (K) 3-by-3 size and stride (S) 1. The layer 3 and layer 5 belong to the pooling layers with the 2-by-2 kernel and stride 3. The 3 layers in the end are the fully-connected layers with 1-by-1 kernel. The usage of temporal spike-based backpropagation ensured that the time steps stay as little as possible. In this study the time steps achieved are 5 with comparatively less number of parameters per time step. The constraints determine the memory obligations and spike processing capacity. Consequently, the constraints needed for a single time step is 21,150,496 and total 5 time steps would make it to 105,752,480.

V. Experiments and Results

The network training was carried out primarily on the two public datasets namely MNIST and CIFAR10. The SNN network has been trained in two instances on NVIDIA GeForce RTX 3090 system with 25.77 GB of memory.

A. MNIST

The training process took $31ms$ per step, each epoch took about 31s and total 50 epochs were configured with batch size 50; the best test accuracy was record as 99.4% at the 33rd

TABLE I
SNN ARCHITECTURE AND NETWORK PARAMETERS

Layer	Input	Output	[Kernel,Stride]	Parameters
conv3-96	(32,32,3)	(32,32,96)	[3x3,1]	2592
conv3-256	(32,32,96)	(16,16,256)	[3x3,1]	221440
pooling	(32,32,256)	(32,32,256)	[2x2,3]	0
conv3-384	(16,16,256)	(8,8,384)	[3x3,1]	885120
pooling	(16,16,384)	(16,16,384)	[2x2,3]	0
conv3-384	(8,8,384)	(8,8,384)	[3x3,1]	1327488
conv3-256	(8,8,384)	(8,8,16384)	[3x3,1]	884992
fc	(1,1,16384)	(1,1,1024)	[1x1,0]	16777216
fc	(1,1,1024)	(1,1,1024)	[1x1,0]	1048576
fc	(1,1,1024)	(1,1,3)	[1x1,0]	3072

epoch. The overall training accuracy was recorded as 99.95% and the loss was about 0.060. The training and testing accuracy curves are shown in Fig. 4 alongside the loss curve. The model testing has been carried on PC alongside FPGA board with randomly selected 10 samples. The processing time for 10 samples on the FPGA board is $125ms$. The test inferences on PC and on FPGA board are shown in Fig. 5. The MNIST digits from 0 to 9 are identified with class IDs and spike counts are accumulated in correspondence to the predicted class identifier. Also, the confusion matrices shown in Fig. 6 were designed to display the accuracy over all the MNIST samples w.r.t classes.

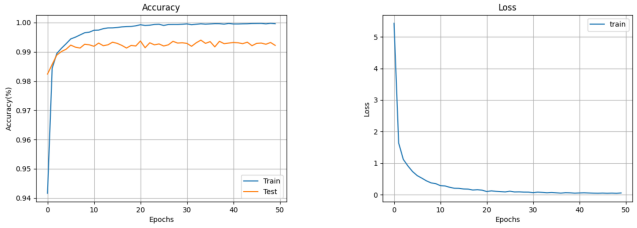

Fig. 4. Accuracy and Loss curves of SNN on MNIST dataset.

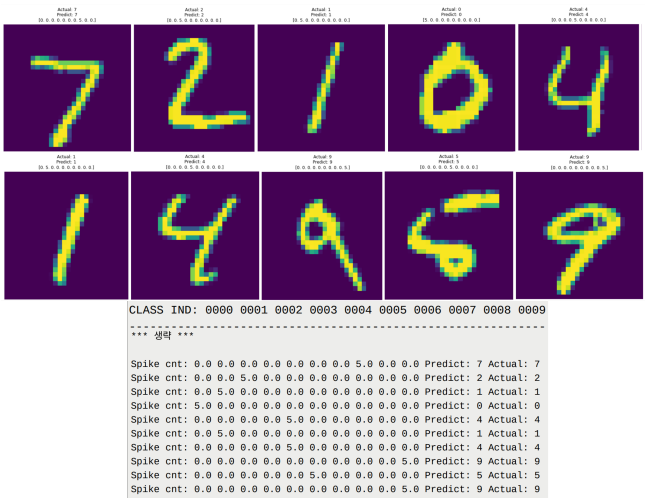

Fig. 5. Predicted outputs of PC (top) and FPGA (bottom) on MNIST dataset.

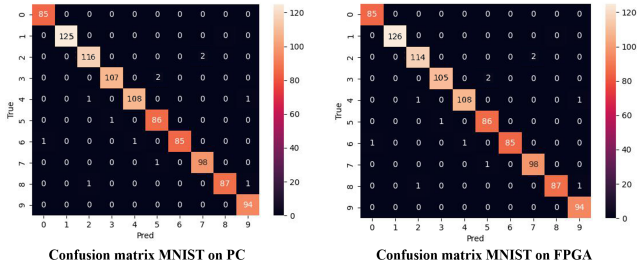

Fig. 6. Confusion matrices for inference over all the MNIST samples on PC and FPGA.

B. *CIFAR10*

The training process took $280ms$ per step, each epoch took about 4m30s and total 150 epochs were configured with batch size 50; the best test accuracy was record as 89.01% at the 94th epoch. The overall training accuracy was recorded as 90.73% and the loss was about 12.194. The training and testing accuracy curves are shown in Fig. 7 alongside the loss curve. The model testing has been carried on PC alongside FPGA board with randomly selected 10 samples. The processing time on FPGA board is $7587ms$. The test inferences on PC and on FPGA board are shown in Fig. 8. The CIFAR10 classes from 0 to 9 are identified with class IDs and spike counts are accumulated in correspondence to the predicted class identifier. Also, the confusion matrices shown in Fig. 9 were designed to display the accuracy over all the CIFAR10 samples w.r.t classes. The overall test samples from MINIST and CIFAR10

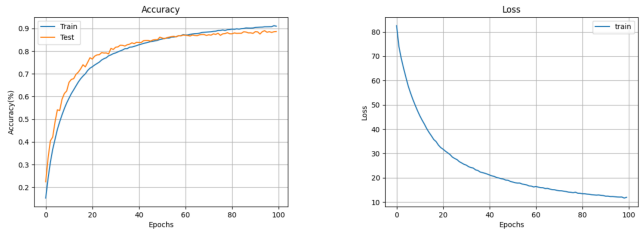

Fig. 7. Accuracy and Loss curves of SNN on CIFAR10 dataset.

were utilized to run the inference on PC alongside FPGA. The total test samples from each dataset were utilized to test the network on both platforms and the results are shown in Table II and Table III. Both tables exhibit the results w.r.t various metrics such as mean, best and mAP in terms of accuracy (in %) and processing time in terms of ms. However, the fundamental performance evaluations such as accuracy and processing time are mainly exhibited using mAP(%) and time(ms).

VI. DISCUSSIONS AND FUTURE WORK

The porting of the SNN model to the FPGA board comes with its own challenges such as the data format issues and it might lead to the reduction in accuracy. The second issue

Fig. 8. Predicted outputs of PC (top) and FPGA (bottom) on CIFAR10.

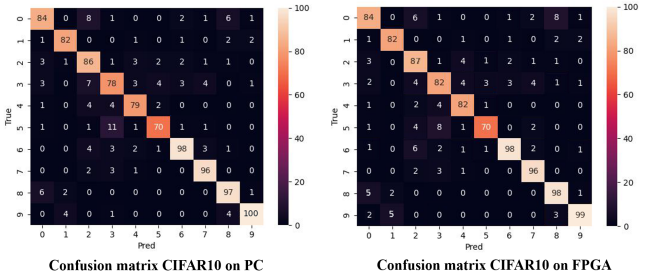

Fig. 9. Confusion matrices for inference over all the CIFAR10 samples on PC and FPGA.

TABLE II
INFERENCE ON PC ALONGSIDE FPGA WITH TOTAL TEST SAMPLES FROM MNIST DATASET.

Platform	Steps	Mean(%)	Best(%)	mAP(%)	Processing time(ms)
PC	5	99.10	99.30	99.07	36
FPGA	5	98.50	98.80	98.79	300

TABLE III
INFERENCE ON PC ALONGSIDE FPGA WITH TOTAL TEST SAMPLES FROM CIFAR10 DATASET.

Platform	Steps	Mean(%)	Best(%)	mAP(%)	Processing time(ms)
PC	5	86.90	87.00	86.94	65.5
FPGA	5	87.00	87.80	87.78	7612

is to maintain the trade-off between the power consumption and processing time. The processing time in case of the FGPA board is high compared to the PC because of resources such as core processor. This difference in processing time capabilities can be found in color-based dataset such as CIFAR10 because of expensive pixel-level computation. The other performance parameters such as memory requirements and accuracy are better in the case of FPGA board w.r.t color image datasets. Especially, this can be viewed in the CIFAR10 dataset case; where the random sample testing revealed the FPGA board processing time is far higher than that of PC. However, the memory requirement and power consumption can be a legitimate issue that might allow researchers to design FPGA board with customized resources that can fit the target application. The processing time is tenfold difference in case of the MNIST dataset between the PC and FPGA board. In the future, the authors tend to test shallower and deeper Spiking Convolution Neural Networks (SCNNs) with surrogate gradient descent on the same FPGA board where the models are customized to meet the requirements of object classification tasks in autonomous vehicle applications. As a part, the customized shallower models can be of great choice with temporal spike and surrogate gradient descent backpropagation algorithms. That way, researchers can leverage both the low-cost FPGA board and high-performance BP algorithms.

VII. CONCLUSION

The investigation of performance of the spiking neural network using temporal spike-based backpropagation on FPGA has been carried out. The FPGA board is specially designed to host the spiking neural networks for the artificial intelligence tasks such as object classification, object detection and segmentation. The publicly available classification datasets such as MNIST, CIFAR10 were employed to examine the performance of the SNNs on FPGA platform. Similarly, the latest temporal spike-based backpropagation technique was chosen to investigate the neuromorphic ability of the low-cost FPGA board in processing SNNs for the object classification tasks. The cross validation of FPGA accuracy and processing time w.r.t PC has been examined. The FPGA embedded board has resulted in fair performance in terms of accuracy. However, the processing time is highly differed from the PC because of the shortage in resources. Nevertheless, the power consumption and memory requirements of the low-cost electronic boards might encourage researchers to design specific FPGA boards for target applications.

ACKNOWLEDGMENT

This work was supported by the Industrial Strategic Technology Development Program of MOTIE/KETI [10077445, Development of SoC technology based on Spiking Neural Cell for smart mobile and IoT Devices].

REFERENCES

[1] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.
[2] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 34, no. 10, pp. 1537–1557, 2015.
[3] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.
[4] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[5] H. Fang, Z. Mei, A. Shrestha, Z. Zhao, Y. Li, and Q. Qiu, "Encoding, model, and architecture: Systematic optimization for spiking neural network in fpgas," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.
[6] R. Lent, "Evaluating the cognitive network controller with an snn on fpga," in *2020 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*. IEEE, 2020, pp. 106–111.
[7] Q. T. Pham, T. Q. Nguyen, P. C. Hoang, Q. H. Dang, D. M. Nguyen, and H. H. Nguyen, "A review of snn implementation on fpga," in *2021 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*. IEEE, 2021, pp. 1–6.
[8] A. M. Abdelsalam, F. Boulet, G. Demers, J. P. Langlois, and F. Cheriet, "An efficient fpga-based overlay inference architecture for fully connected dnns," in *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE, 2018, pp. 1–6.
[9] A. Khodamoradi, K. Denolf, and R. Kastner, "S2n2: A fpga accelerator for streaming spiking neural networks," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2021, pp. 194–205.
[10] S. Li, Z. Zhang, R. Mao, J. Xiao, L. Chang, and J. Zhou, "A fast and energy-efficient snn processor with adaptive clock/event-driven computation scheme and online learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 4, pp. 1543–1552, 2021.
[11] W. Zhang and P. Li, "Spike-train level backpropagation for training deep recurrent spiking neural networks," *Advances in neural information processing systems*, vol. 32, 2019.
[12] Y. Jin, W. Zhang, and P. Li, "Hybrid macro/micro level backpropagation for training deep spiking neural networks," *Advances in neural information processing systems*, vol. 31, 2018.
[13] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, 2002.
[14] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," *Advances in neural information processing systems*, vol. 31, 2018.
[15] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," *Advances in neural information processing systems*, vol. 31, 2018.
[16] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks," *Advances in neural information processing systems*, vol. 31, 2018.
[17] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.
[18] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.
[19] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature communications*, vol. 11, no. 1, pp. 1–15, 2020.
[20] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience*, vol. 12, p. 331, 2018.
[21] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.
[22] W. Zhang and P. Li, "Temporal spike sequence learning via backpropagation for deep spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 022–12 033, 2020.
[23] T. Syed, V. Kakani, X. Cui, and H. Kim, "Exploring optimized spiking neural network architectures for classification tasks on embedded platforms," *Sensors*, vol. 21, no. 9, p. 3240, 2021.
[24] S. Tehreem, V. Kakani, X. Cui, and H. Kim, "Spiking neural networks using backpropagation," in *2021 IEEE Region 10 Symposium (TENSYMP)*. IEEE, 2021, pp. 1–5.