Fine-Tuning *Surrogate Gradient Learning* for Optimal Hardware Performance in Spiking Neural Networks

Ilkin Aliyev and Tosiron Adegbija
Department of Electrical and Computer Engineering
The University of Arizona, Tucson, AZ, USA
Email: {ilkina, tosiron}@arizona.edu

***Abstract*—The highly sparse activations in Spiking Neural Networks (SNNs) can provide tremendous energy efficiency benefits when carefully exploited in hardware. The behavior of sparsity in SNNs is uniquely shaped by the dataset and training hyperparameters. This work reveals novel insights into the impacts of training on hardware performance. Specifically, we explore the trade-offs between model accuracy and hardware efficiency. We focus on three key hyperparameters: surrogate gradient functions, `beta`, and `membrane threshold`. Results on an FPGA-based hardware platform show that the `fast sigmoid` surrogate function yields a lower firing rate with similar accuracy compared to the `arctangent` surrogate on the SVHN dataset. Furthermore, by cross-sweeping the `beta` and `membrane threshold` hyperparameters, we can achieve a 48% reduction in hardware-based inference latency with only 2.88% trade-off in inference accuracy compared to the default setting. Overall, this study highlights the importance of fine-tuning model hyperparameters as crucial for designing efficient SNN hardware accelerators, evidenced by the fine-tuned model achieving a $1.72\times$ improvement in accelerator efficiency (FPS/W) compared to the most recent work.**

***Index Terms*—Surrogate Gradient Learning, Sparsity-aware SNN, Neuromorphic Computing.**

I. Introduction

Recent studies have demonstrated significant benefits to considering sparsity in improving hardware efficiency in Spiking Neural Networks (SNNs) [1], [2]. For example, Yin et al. [1] showed that by explicitly exploiting the sparse gradients (as high as 93% for some datasets) in hardware, training SNNs can consume up to $5.58\times$ less energy compared to training on the same hardware without considering sparsity. Similarly, Wang et al. [2] achieved a $2.1\times$ improvement in inference efficiency by exploiting sparsity in hardware compared to sparsity-oblivious hardware. The primary driving factor in the formation of the sparsity characteristic is the input coding scheme of the dataset. Recognizing that this is an active area of research and various encoding approaches are being proposed to reduce the firing rate of the network, in this study, we explore, for the first time, how training hyperparameters can influence the sparsity of SNN models and, in effect, the hardware performance of SNN accelerators.

Surrogate gradients [3] are typically employed to train SNN models due to their ability to overcome the non-linear nature of spiking neurons (i.e., binary activations/spikes instead of linear continuous activations). The surrogate gradient approximates the true derivative, thereby effectively providing state-of-the-art accuracy in classification-oriented machine learning (ML) tasks. Nonetheless, the choice of a surrogate function (or the scaling factor of its derivative) is not standardized. This raises the question of how its implementation affects the network's classification accuracy and firing intensity.

To investigate our hypothesis, we systematically study two well-known surrogate functions: `arctangent` and `fast sigmoid`, empirically evaluating each function's optimal trade-off points for learning performance versus sparsity under a range of *derivative factors*. We also rigorously evaluate the impact of two critical hyperparameters: `beta` (β) and `threshold` (θ). We leverage our in-house hardware platform to conduct these hardware experiments. This platform allows efficient "model-to-hardware" mapping by accounting for the model's layer-wise workload characteristics. To the best of our knowledge, this work is the first to present a holistic evaluation of SNN hardware from a training perspective.

II. Background

A. Spiking Neuron Model

The spiking neuron model used is a leaky integrate-and-fire (LIF) neuron, whose characteristics are shown in Equations 1 and 2.

$$u_j[t+1] = \beta u_j[t] + \sum_i w_{ij} s_i[t] - s_j[t]\theta \tag{1}$$

$$s_j[t] = \begin{cases} 1, & \text{if } u_j[t] > \theta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Here, β represents the decay or leak factor of the neuron's membrane potential, typically ranging between 0 and 1. This parameter influences how the previous potential $u_j[t]$ affects the current potential $u_j[t+1]$. A higher β value implies less decay, enabling the neuron to retain more of its previous state, which can increase the likelihood of firing. θ, on the other hand, represents the threshold value that the neuron's membrane potential must surpass to trigger a firing event, as indicated by a spike $s_j[t]$. A lower θ value reduces the potential required for firing, thereby increasing the neuron's firing frequency.

B. Surrogate Approximation Functions

Surrogate gradients have emerged as a competitive solution to effectively approximate the step function [3]. Equations 4 and 3 show the approximation formulas used for `arctangent` and `fast sigmoid`, respectively.

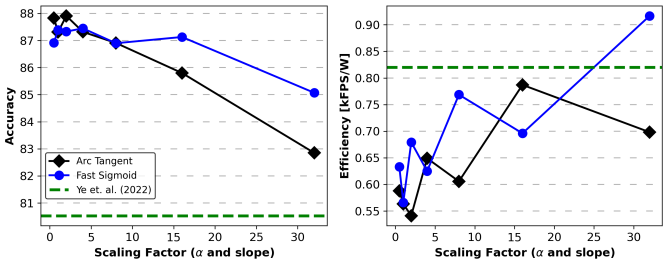

Fig. 1: Cross-comparison results for `arctangent` and `fast sigmoid` surrogate functions over varying derivative scaling factors

$$S \approx \frac{1}{\pi}\arctan\left(\pi U \frac{\alpha}{2}\right) \quad (3) \qquad S \approx \frac{U}{1+k|U|} \quad (4)$$

where U is the membrane potential and α and k are derivative scaling factors for each surrogate gradients.

III. Surrogate Gradient Fine-Tuning and Hyperparameter Search Results

A. Experimental Setup

We used snnTorch [4] to construct the spiking neuron models and PyTorch for training the network using the Street View House Numbers (SVHN) dataset. The network is a convolutional SNN with the following structure: 32C3-P2-32C3-MP2-256-10 (where XCY stands for X filters with size $Y \times Y$ and MPZ for maxpooling with size of $Z \times Z$). We employ cosine annealing [5] for the learning rate scheduler (with epochs set to 25) during training due to its ability to rapidly converge to optimal accuracy. The trained model was mapped to an in-house hardware platform[1], developed in SystemVerilog, and implemented on a Xilinx Kintex® UltraScale+™ FPGA. This hardware efficiently allocates platform resources for the model by leveraging the model's layer sizes and layer-wise sparsity characteristics to achieve an ultra-low power resource allocation scheme. In addition, the hardware operates in a layer-wise lock-step manner to save memory resources and achieve high throughput.

B. Results

Surrogate functions: To systematically evaluate each surrogate function, we performed a parameter sweep over the derivative scaling factors k and α while leaving both β and θ set to their default values (0.25 and 1.0 respectively). Figure 1 shows the variation in accuracy and accelerator efficiency (FPS/W) for each surrogate gradient. We set both k and $\alpha$to the value range of 0.5 to 32, beyond which the accuracy for the `arctangent` surrogate drops below 20%. We observe that while both surrogate gradients follow a similar trend in accuracy and efficiency, the `fast sigmoid` yields lower firing activity (i.e., higher sparsity) compared to the `arctangent`, resulting in higher accelerator efficiency. Moreover, in both surrogate gradients, through hyperparameter tuning, our network yields higher accuracy than previous work [6] (as highlighted by the horizontal green line in Figure 1),

[1]Publicly available at https://github.com/githubofaliyev/SNN-DSE

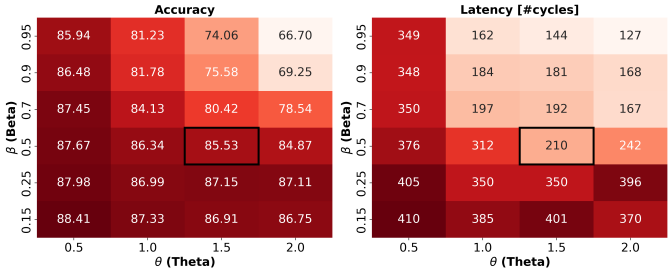

Fig. 2: Cross-sweep results for β and θ parameters.

using the same network architecture and dataset, while the `fast sigmoid` achieves 11% better accelerator efficiency.

Beta-threshold cross sweep: Given its high sparsity, we chose the `fast sigmoid` surrogate with a slope scaling factor of 0.25 for the following experiments. In Figure 2, we cross-sweep β (leakage factor) and θ (threshold potential). Our analysis identifies the optimal balance at a β value of 0.5 and a θ value of 1.5. This configuration significantly reduced the inference latency by 48%, while only incurring a minor accuracy loss of 2.88%, compared to the best accuracy configuration. Compared to prior work [6], this fine-tuning—with β set to 0.7 and θ to 1.5— achieved 1.72× greater hardware efficiency without degrading the accuracy.

IV. Conclusion

This study sheds new light on a previously unexplored aspect of SNN hardware accelerator design. While previous research has primarily focused on non-hardware-related factors like dataset encoding, our work pioneers the evaluation of the training hyperparameter space in relation to hardware efficiency. We show that fine-tuning surrogate gradient hyperparameters can provide significant benefits to hardware efficiency and should be considered carefully in the design of efficient SNN accelerators. In future work, we aim to broaden our analysis by exploring additional datasets and the hardware efficiency impacts of other hyperparameters like loss functions.

References

[1] R. Yin, A. Moitra, A. Bhattacharjee, Y. Kim, and P. Panda, "Sata: Sparsity-aware training accelerator for spiking neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[2] Z. Wang, Y. Zhong, X. Cui, Y. Kuang, and Y. Wang, "A spiking neural network accelerator based on ping-pong architecture with sparse spike and weight," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2023.

[3] 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.

[4] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, 2023.

[5] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[6] W. Ye, Y. Chen, and Y. Liu, "The implementation and optimization of neuromorphic hardware for supporting spiking neural networks with mlp and cnn topologies," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 2, pp. 448–461, 2022.