Towards Biologically-Plausible Neuron Models and Firing Rates in High-Performance Deep Spiking Neural Networks

Chen Li
Department of Computer Science
The University of Manchester
Manchester, UK
chen.li@manchester.ac.uk

Steve Furber
Department of Computer Science
The University of Manchester
Manchester, UK
steve.furber@manchester.ac.uk

ABSTRACT

Spiking neural networks (SNNs) have the potential to reach the same accuracy level as their counterpart artificial neural networks (ANNs) through ANN-to-SNN conversion. However, this relies significantly on artificially modifying the voltage reset mechanism in the Integrate-and-Fire (IF) neuron model to support bias and batch normalization. This paper shows that our proposed MCR-Norm (minimum chain rule normalization) can enable the modeling of these two key elements by using the standard IF model. The SNNs built by the IF model can thus reach the same accuracy level as the SNNs built by the artificially modified IF model. While previous approaches tended to push deep SNNs towards very high firing rates, we found that the IF neuron is suitable to run in a low firing rate range. This is in line with biological observations and is also crucial to go beyond SNN simulation and apply it to real neuromorphic hardware. In addition, tuning the neural network inputs is dismissed in earlier work, but we claim that it is closely related to the success of parameter normalization inside the SNNs built by the standard IF model, and can be integrated naturally into MCR-Norm.

KEYWORDS

spiking neural networks, ANN-to-SNN conversion, Integrate-and-Fire neuron

ACM Reference Format:
Chen Li and Steve Furber. 2021. Towards Biologically-Plausible Neuron Models and Firing Rates in High-Performance Deep Spiking Neural Networks. In *International Conference on Neuromorphic Systems 2021 (ICONS 2021), July 27–29, 2021, Knoxville, TN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3477145.3477146

1 INTRODUCTION

SNNs are neural network models using spiking neurons as computational units and spikes as information carriers between these units, with the aim of achieving high computational performance by imitating the human brain [2, 14]. SNNs have been successfully adopted by computational neuroscientists to describe the dynamics of biological neural systems and neural circuits [4, 15], benefiting from their biological plausibility and computational capability. With these successes in biological neural modeling and promoted by the parallel achievements of deep learning, there have been many approaches proposed to build deep SNNs to achieve power-efficient, event-based and low-latency computing [1, 3, 5, 7, 9, 11, 16, 17, 22].

The most successful approach to date trains an artificial neural network (ANN) using supervised learning and converts the trained ANN into a rate-based SNN [1, 9, 13]. Lossless conversion can be achieved by this method, but it usually requires using spiking neuron models that are “artificially" modified to transmit precise information to the subsequent layer [18, 20, 21]. These methods can achieve good accuracy on many benchmarks but work against the original objective of using spiking neurons in deep neural networks.

We believe that using more biologically plausible spiking neuron models to build functional SNNs is a desirable way to go. Though there will be many obstacles to this research direction, this is an unavoidable process if we wish to unlock the mysteries of the human brain and duplicate them on the machine. Previous work has explored building deep SNNs using the standard IF model [19], but the modeling of bias and batch normalization remains unaddressed until now. These two elements directly affect the inference accuracy in ANNs which is also the target accuracy of SNNs. Lacking of these elements is obstructing the performance of SNNs built by the standard IF model as well as their further applications

In this paper, we will demonstrate that it is possible to retain both high performance and biological plausibility in SNNs. Also, some insights into managing the firing rate range in deep SNNs are offered. The main contributions of this paper are:

- We achieve state-of-the-art (SOTA) accuracy on the MNIST and CIFAR-10 data sets on deep SNNs built using the standard IF model.
- Bias and batch normalization are modeled in deep SNNs whilst retaining biological fidelity.
- Our method features a 2.5 to 7.5 times lower spike rate than previous SOTA ANN-to-SNN conversion methods, so paves the way to run these networks on neuromorphic hardware.
- We emphasize the significance of input normalization for SNNs and integrate it into MCR-Norm to form the first systematic parameter normalization strategy for SNNs.

2 RELATED WORK

Several successful parameter normalization approaches have been proposed in deep SNNs to facilitate ANN-to-SNN conversion [3, 18, 19].

Data-based normalization scales weights according to the input patterns of spiking neurons. The input patterns of the test data set are estimated from the input patterns of the training data set. This

ICONS 2021, July 27–29, 2021, Knoxville, TN, USA

ACM ISBN 978-1-4503-8691-3/21/07.
https://doi.org/10.1145/3477145.3477146

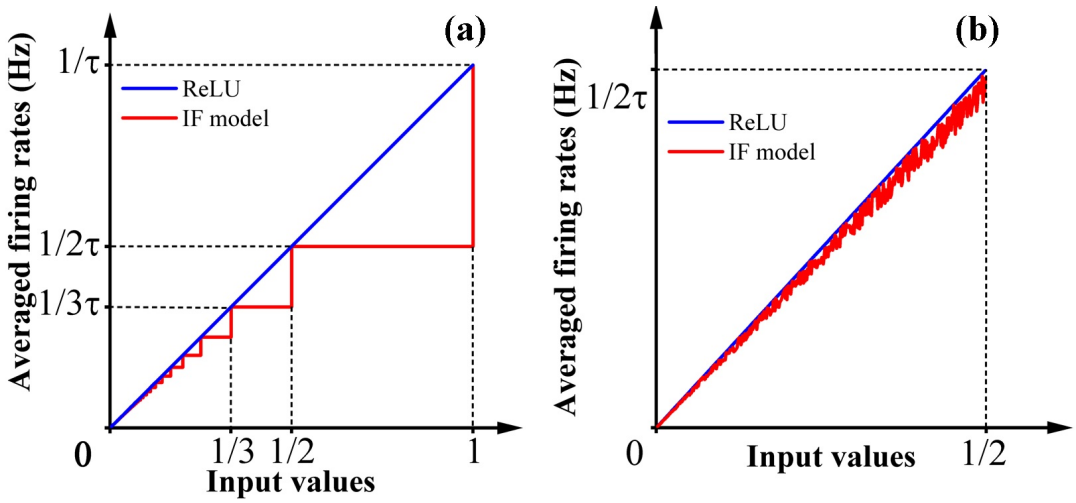

Figure 1: The response curves of the IF model and ReLU. The input is a constant current injection, and the synapse model is the delta model. (a) The response curve of the IF model and ReLU. (b) Changing the input to a statistical current injection.

estimation relies on an independent and identically distributed (IID) hypothesis between the training data set and the test data set.

Percentile data-based normalization provides an additional bias normalization equation on the basis of data-based normalization [18]. The balance of weights and biases in SNNs are maintained by adding an extra membrane potential reset mechanism to the IF model. The outliers of the neuron inputs are discarded to improve firing rates. However, this discard may bring accuracy loss if the outliers contain important information.

Spike-Norm calculates the maximum inputs in spiking neurons layer-by-layer in a sufficiently long time window to determine the values of the thresholds [19]. Spike-Norm successfully applies the original IF model to challenging data sets, but does not include the normalization of biases and batch normalization layers. Due to the inherent noise in SNNs, the normalized thresholds calculated by this method vary in different trials and with different lengths of time-window.

3 BACKGROUND ON ANN-TO-SNN CONVERSION

ANN-to-SNN conversion generally involves training an ANN and then converting it into a rate-coded SNN. Usually, normalization of weights and biases is applied to reduce the accuracy loss originating from the different neuronal dynamics in artificial and spiking neurons [3, 18].

When the activation function in the ANN is ReLU, the weights w_n and biases b_n in layer n are normalized as $\lambda_{n-1}/\lambda_n * w_n$ and b_n/λ_n respectively [18]. The issue of thresholds is ignored and is set arbitrarily to 1 for all spiking neurons. λ_n is the maximum activation value in the artificial neurons of layer n, and λ_0 is set to 1. With this parameter normalization, the weights and biases in layer n will be scaled by $1/\lambda_n$ to map the activation values of the artificial neurons to the output firing rates of the spiking neurons. In addition, the weights need to be scaled by λ_{n-1} to compensate for the effect of the scaling in previous layers. This additional scaling stems from the fact that in neural networks, weights receive information from the previous layer whereas biases are fed directly into the current layer, therefore only the weights are affected by the scaling in the previous layer. If using a chain rule to accumulate the impact of the scaling in all previous layers, the weight normalization equation will have the same scale factor as in the bias normalization equation:

$$\widetilde{w}_n = \prod_{k=1}^{n} \frac{\lambda_{k-1}}{\lambda_k} * w_n = \frac{1}{\lambda_n} * w_n \tag{1}$$

Batch normalization (BN) scales the inputs of artificial neurons according to the input distributions and learnable parameters. These computations are linear, so a BN layer can be modeled by weight and bias scaling in SNNs [18].

$$\widetilde{w}_n = \frac{\gamma_n}{\sigma_n} * w_n \tag{2}$$

$$\widetilde{b}_n = \frac{\gamma_n}{\sigma_n} * (b_n - \mu_n) + \beta_n \tag{3}$$

4 TWO CHALLENGES IN APPLYING THE STANDARD IF MODEL TO DEEP SNNS

The difficulties in using the standard IF model after ANN-to-SNN conversion come from the “firing rate degeneration" phenomenon in deep SNNs. Figure 1(a) illustrates the response curve of the standard IF model to constant current injection and a comparison with the target response curve, e.g. ReLU, in ANNs. We can see that in this figure the shape of the IF response curve is unevenly stepped and a gap exists between these two curves.

The stepped shape can almost be eliminated when the input is noisy. The noisy input can be generated by changing constant current injection to Gaussian current injection. In deep SNNs, the input of a spiking neuron is noisy as well since it receives randomly generated spikes from spiking neurons in the previous layer. A typical

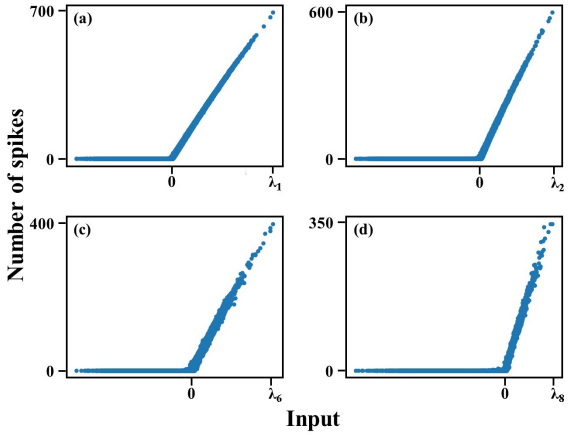

Figure 2: The firing rate degeneration phenomenon on deep SNNs. The figures with labels (a) to (d) show the response curves in layers 1, 2, 6 and 8 respectively. The neural model is the standard IF model and the weights are normalized by the traditional parameter normalization method [18].

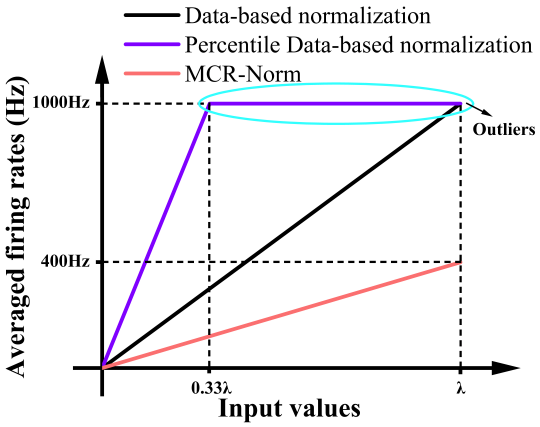

Figure 3: The response curves of spiking neurons under different parameter normalization methods.

response curve of the IF model smoothed by noisy input is shown in figure 1(b). We can see that the response curve of the IF model is now similar to ReLU, especially when the input value is small. However, the output firing rate is still lower than the expected ReLU-like shape; we call it the "firing rate degeneration" phenomenon in the IF model. This phenomenon imposes two challenges to deep SNNs that are converted from ANNs:

(1) The degenerated firing rate in one layer generates a ripple effect in subsequent layers. More specifically, with the network going deeper, the spike firing rate range gradually moves towards a lower region. (Figure 2). When the network is deep, the spiking neurons in deep layers are rarely activated so require more time steps to transmit the same amount of information, which eventually leads to a long inference latency.

(2) When conducting data-based normalization, weights are scaled additionally by a factor λ_{n-1} more than the bias to compensate for the weight scaling in the previous layer. Nevertheless, since the response curve of the spiking neuron suffers from firing rate degeneration (as well as its ripple effect in subsequent layers), this scaling is actually bigger than it should be. For this reason, the weights and biases are mismatched and, consequently, the inference accuracy drops. In other words, the key problem is that the degree of firing rate degeneration is inestimable, which makes all the weight and bias normalization relying on deterministic calculation fail.

5 MCR-NORM

To overcome these challenges caused by firing rate degeneration, and to achieve high-performance deep SNNs using the standard IF model, we propose a normalization method called MCR-Norm, an abbreviation of minimum chain rule normalization. The “minimum” in this term emphasizes that the firing rate range in every layer after applying MCR-Norm is maintained identically at a low level. “chain rule” emphasizes that the normalization is performed layer-by-layer under a chain rule.

The essence of MCR-Norm is that the accuracy of deep SNNs depends on achieving a balance between weight and bias - by scaling weights correctly; the inference latency of deep SNNs needs to be reduced by preventing too low firing rates in deep layers - by normalizing the firing rate to an identical range in all layers.

Figure 3 compares the response curves of spiking neurons in MCR-Norm and other parameter normalization strategies. All these approaches normalize thresholds proportional to the maximum activation λ. Note that this diagram is simplified to show the essence of these parameter normalization methods. The real response curve of spiking neurons has variation and noise. We can see that for the same inputs, the spike firing rate after using MCR-Norm is 2.5 times smaller than data-based normalization and 7.5 times smaller than percentile data-based normalization (if ignoring the 0.1% to 1% outlier neurons).

6 INSPIRATION FROM MACHINE LEARNING AND CONSIDERATIONS OF NEUROMORPHIC HARDWARE

The firing behaviour of spiking neuron can be controlled by one hyper-parameter threshold: a high threshold causes a low output firing rate and a low threshold results in a high output firing rate. In MCR-Norm, we adjust weights (which is inversely equivalent to changing thresholds) slightly to compensate for the effect of firing rate degeneration in the IF neuron. Then the firing rate range is identical in every layer, and biases can be normalized according to this determined firing rate range.

The phenomenon of firing rate degeneration occurs in every layer so the weights (thresholds) need to be tuned in each layer. Inspired by batch normalization in ANNs which normalizes the inputs in each layer to the same distribution, our normalization is conducted layer-wise and the spike firing rate is normalized to the same range.

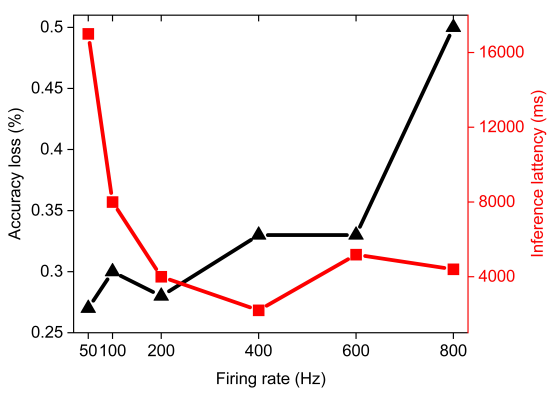

Figure 4: Inference accuracy loss after ANN-to-SNN conversion and inference latency for different firing rates. The data set is CIFAR-10 [18].

One main goal of neuromorphic hardware is conducting the simulation of biological neural networks. The biological neural network is naturally sparse and the maximum firing rate is below 200Hz [12], rather than 1000Hz as used in most SNN simulations. From the hardware viewpoint, the construction of neuromorphic hardware usually considers the worst situation during running. For example, the highest spike rate that might be received by one hardware node, and whether this traffic upper bound is the same in different hardware nodes. This consideration of the highest firing rate is reflected in our proposed MCR-Norm, where the firing rate upper bound is limited to an identical low value in every layer. Consequently, when the simulated results are applied to the real neuromorphic hardware, the spike traffic upper bound is low and identical in hardware nodes.

7 NORMALIZING WEIGHTS VS NORMALIZING BIASES

To maintain the balance between weights and biases under the firing rate degeneration phenomenon, there are two feasible solutions: scaling weights to match biases, or scaling biases to match weights. The latter solution can achieve a balance between parameters but cannot solve the firing rate degeneration problem. With that in mind, we choose to scale weights in our method.

8 LOW FIRING RATES VS HIGH FIRING RATES

MCR-Norm scales weights to make the firing rate rage in every layer identical. In this section, we quantitatively explore the optimal firing rate range to achieve high performance. Under the same experimental conditions (listed in the supplementary materials), we adjust the upper bound of the firing rate range and record its impact on the network performance, specifically, inference accuracy and inference latency as shown in Figure 4. We can see that the accuracy drops when the firing rate is high, while high latency appears at low firing rates. The results suggest that unlike modified IF models, there is a trade-off inside the choice of the working zone of the standard IF model. The balance point is approximately between 200Hz - 400Hz. This is in line with some biological observations that neurons usually fire below 200Hz as the cost of spike generation is high [12].

9 MCR-NORM EQUATIONS

The equations used in MCR-Norm are shown below. λ_n and λ_{n-1} are the maximum activations in layer n and layer $n-1$ respectively. Compared with the traditional parameter normalization method, there are three additional scaling factors g_{n-1}, g_n and h_{n-1} added to the normalization equations. The subscript of these scale factors represents the layer in which the scaling calculation happens. Since the normalization is performed layer-by-layer, only two scale factors g_n and h_n needed to be calculated in layer n. The value of λ_0, g_0 and h_0 are set to 1.

$$\widetilde{w}_n = w_n * \frac{\lambda_{n-1} \cdot g_n}{\lambda_n \cdot g_{n-1}} \tag{4}$$

$$\widetilde{b}_n = b_n * \frac{h_{n-1} \cdot g_n}{\lambda_n \cdot g_{n-1}} \tag{5}$$

To normalize weights in layer n, only the value of g_n needs to be determined, as g_{n-1} was derived in the previous layer. g_n is found by scanning different values of g_n in layer n and choosing the one that makes the maximum firing rate in this layer equal to the target, e.g. 200Hz.

For the bias normalization in layer n, h_{n-1} needs to be determined besides g_{n-1} and g_n calculated in weight normalization. The scaling factor h_{n-1} is predefined to represent the target firing rate in the previous layer $n-1$, e.g. h_{n-1} is set to 0.2 if the target firing rate range is (0, 200Hz).

According to Equation 4 and 5, the weights are normalized additionally by a factor λ_{n-1}/h_{n-1} than biases. This additional scaling compensates for the mapping in the previous layer $n-1$ from the ANN activation in the range of $[0, \lambda_{n-1}]$ to the SNN firing rates in the range of $[0, h_{n-1}/\tau]$. More detailed information is in the supplementary material.

.

10 MCR-NORM INPUT NORMALIZATION

In addition to normalizing the weights and biases inside the network, the normalization of the inputs fed into the network also has a major impact on the performance of the SNN. Previous SNN normalization techniques skip the input normalization and start the normalization from the first hidden layer [3, 18, 19]. The correctness of these normalization methods relies on the assumption that the inputs to the ANN are in the range (0, 1) or (-1, 1), and the normalized SNN inputs are in the range of $(0, 1/\tau)$, which is (0, 1000Hz) if the time constant τ is 1ms. These two assumptions are not always fulfilled, e.g. the inputs of CIFAR-10 after input normalization in ANNs are in the range about (-2.2, 2.2), and the input firing rate range of SNNs can be (0, 400Hz) in some configurations [3].

To correct this fault in parameter normalization after ANN-to-SNN conversion, and to enable the flexible configuration of input firing rates, MCR-Norm is extended to include normalization in the

Table 1: Network structures for MNIST and CIFAR-10

MNIST
28*28-64c3BN-128 c3BN-128 c3BN-p2-D0.1 -128 c3BN-256c3BN-256c3BN-p2-D0.1-256 c3BN -512 c3BN-D0.1 -2048FC-D0.4-10FC
CIFAR-10
32*32*3-64c3BN-128c3BN-128c3BN-p2-D0.1 -128 c3BN-256c3BN-p2-D0.1-256 c3BN- 256c3BN -p2-D0.1-512 c3BN-p2-D0.1-2048FC-D0.4-10FC

Table 2: Training parameters and hyperparameters of neural networks for MNIST and CIFAR-10.

Dataset	MNIST	CIFAR-10
Criterion	Cross entropy	Cross entropy
Optimizer	Adadelta	Adadelta
Epochs	150	150
Batch size	100	100
Other techniques	Early stopping	Early stopping, data augmentation, L2 regulator

input layer. This is achieved by considering that the input range $(0, \lambda_0)$ in the ANN is linearly mapped to the input spike firing rate range $(0, g_0/\tau)$ in the SNN. Thus, the MCR-Norm equation starts in the input layer rather than the first hidden layer. g_0 controls the target firing rate range, e.g. g_0 is set to 0.4 if the target input firing rate range is (0, 400Hz). h_0 is simply set as 1. The normalization of negative inputs is achieved in the same way by using signed spikes [18].

11 EXPERIMENTAL SETUP

The datasets used here are MNIST and CIFAR-10 [8, 10]. The network architectures are inspired by earlier work [8] and are given in Table 1. There are several convolutional layers with BN layers to extract features, and some fully-connected (FC) layers at the end. With the deepening of the network, the number of convolution channels increases but the size of channel decreases through average pooling layers. The size of the convolutions kernels is 3*3 and the size of the pooling kernels is 2*2. Spatial dropout is applied after some pooling layers and the standard dropout is applied before the FC layers. The activation function used in these networks is ReLU. The network parameters and hypermeters are shown in Table 2. The MNIST inputs are in (0,1) and the CIFAR inputs are pre-processed in three channels.

The spiking neuron model used here is the standard IF model and the synapse model is the delta model [3]. We use a statistical rate coding as the input of these networks to linearly map input values to averaged firing rates. The spike generation of the rate coding follows a Bernoulli distribution every time step. The time resolution is 1 ms.

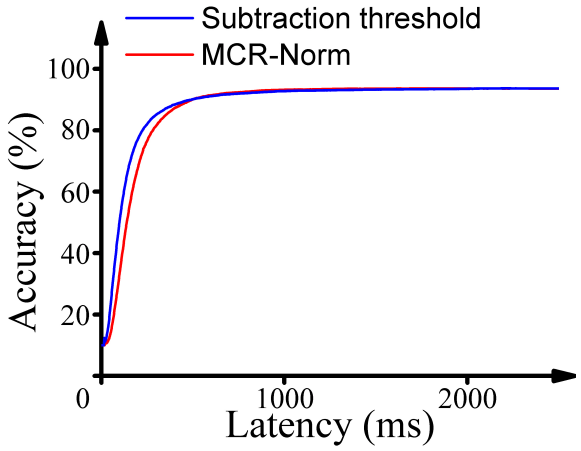

Figure 5: The convergence time of the SNN on CIFAR-10.

All experiments are run on a computer with a Core i7 2.8 GHz multi-core CPU, 8 GB RAM and a GTX1050Ti GPU. The operating system on this computer is Windows10. The ANNs and SNNs are built using Pytorch.

12 BENCHMARK RESULTS

MCR-Norm was tested on the MNIST and CIFAR-10 pattern recognition data sets. The inference accuracy is compared with various ANN-to-SNN conversion techniques in Table 3. We achieved a state-of-the-art accuracy of 99.71% on MNIST, benefiting from the powerful architecture and the use of the batch normalization technique. More importantly, we achieved zero accuracy loss after ANN-to-SNN conversion, without any modifications of the standard IF model.

For CIFAR-10, we achieved an accuracy of 93.60%, which is the best reported accuracy with the standard IF model up to now. It shows approximately 2% accuracy improvement over SNNs built using the standard IF model without biases and BN layers [19]. The ANN-to-SNN conversion accuracy loss is 0.29%, which is better than many methods using less biologically plausible neural models [18, 20, 21]; however, our conversion accuracy loss is higher than [7, 18] which uses the IF model with an extra membrane potential reset mechanism.

13 INFERENCE TIME

The SNN takes 100 ms to converge to the final accuracy on MNIST during inference. The inference latency of the SNN on CIFAR-10 is much longer than on MNIST. It takes 1,700 ms to reach the final accuracy as shown in Figure 5. We compare our results on CIFAR-10 with a deep SNN that is converted from the same ANN model but normalized using data-based normalization and uses the IF model with subtraction mechanism [18]. The results are shown in Figure 5 and we can see that our method does not incur additional latency during inference. What is more, the firing rate range is (0, 400Hz) rather than (0, 1000Hz) appeared in data-based normalization and percentile data-based normalization.

Table 3: Accuracy loss on MNIST and CIFAR-10 with ANN-to-SNN conversion techniques.

Neuron type	Bias	BN	Dataset	ANN Loss (%)	SNN Loss(%)
Original IF model [3]	No	No	MNIST	0.86	0.90
IF model with subtraction mechanism [18]	Yes	Yes	MNIST	0.56	0.56
Standard IF model (this work)	Yes	Yes	MNIST	0.29	0.29
IF model with subtraction mechanism [18]	Yes	Yes	CIFAR-10	8.09	9.15
Original IF model [19]	No	No	CIFAR-10	8.3	8.45
IF model with soft reset [7]	No	No	CIFAR-10	6.37	6.37
AMOS unit [20]	Yes	Yes	CIFAR-10	7.07	7.58
FS neuron [21]	Yes	Yes	CIFAR-10	7.01	7.58
Standard IF model (this work)	Yes	Yes	CIFAR-10	6.11	6.40

14 CONCLUSIONS AND FUTURE WORK

We propose the "MCR-Norm" normalization method to tune parameters systematically after ANN-to-SNN conversion and, using that, we achieved competitive accuracy on the MNIST and CIFAR-10 pattern recognition data sets using the standard IF model. As well as the absolute accuracy of the deep SNN, the accuracy loss compared with the original ANN after MCR-Norm is crucial to represent the efficiency of ANN-to-SNN conversion. Thanks to the layer-wise optimization, our accuracy loss is only 0.29% on CIFAR-10 and 0% on MNIST.

The core of MCR-Norm is to control the firing rate range at a low level, rather than allowing varying and/or high firing rates. We show that a biologically plausible neuron model - specifically, the standard IF model - naturally prefers a firing rate range close to the one used by biological neurons. Our normalization approach also paves the way to efficiently adopt SNNs on neuromorphic hardware [6].

Future work will further extend the proposed MCR-Norm method to more network architectures and tasks. On a similar network architecture, CIFAR-10 has an obviously longer latency than MNIST. We will explore the reasons behind this phenomenon to speed up the SNN inference.

15 FUNDING

The research leading to these results has received funding from the EU Flagship Human Brain Project (H2020 945539).

REFERENCES

[1] Yongqiang Cao, Yang Chen, and Deepak Khosla. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision* 113, 1 (2015), 54–66.
[2] Sergio Davies, Javier Navaridas, Francesco Galluppi, and Steve Furber. 2012. Population-based routing in the SpiNNaker neuromorphic architecture. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.
[3] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*. ieee, 1–8.
[4] Chris Eliasmith, Terrence C Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, and Daniel Rasmussen. 2012. A large-scale model of the functioning brain. *science* 338, 6111 (2012), 1202–1205.
[5] Steve K Esser, Rathinakumar Appuswamy, Paul Merolla, John V Arthur, and Dharmendra S Modha. 2015. Backpropagation for energy-efficient neuromorphic computing. In *Advances in neural information processing systems*. 1117–1125.
[6] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. 2014. The spinnaker project. *Proc. IEEE* 102, 5 (2014), 652–665.
[7] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. 2020. RMP-SNN: Residual Membrane Potential Neuron for Enabling Deeper High-Accuracy and Low-Latency Spiking Neural Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13558–13567.
[8] Seyyed Hossein Hasanpour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. 2016. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv preprint arXiv:1608.06037* (2016).
[9] Eric Hunsberger and Chris Eliasmith. 2015. Spiking deep networks with LIF neurons. *arXiv preprint arXiv:1510.08829* (2015).
[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[11] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. 2016. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience* 10 (2016), 508.
[12] Peter Lennie. 2003. The cost of cortical computation. *Current biology* 13, 6 (2003), 493–497.
[13] Qian Liu, Yunhua Chen, and Steve Furber. 2017. Noisy softplus: an activation function that enables snns to be trained as anns. *arXiv preprint arXiv:1706.03609* (2017).
[14] Wolfgang Maass. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural networks* 10, 9 (1997), 1659–1671.
[15] Wolfgang Maass, Prashant Joshi, and Eduardo D Sontag. 2007. Computational aspects of feedback in neural circuits. *PLoS Comput Biol* 3, 1 (2007), e165.
[16] Emre O Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. 2017. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in neuroscience* 11 (2017), 324.
[17] Peter O'Connor and Max Welling. 2016. Deep spiking networks. *arXiv preprint arXiv:1602.08323* (2016).
[18] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. 2017. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience* 11 (2017), 682.
[19] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. 2019. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience* 13 (2019), 95.
[20] Christoph Stöckl and Wolfgang Maass. 2019. Recognizing Images with at most one Spike per Neuron. *arXiv preprint arXiv:2001.01682* (2019).
[21] Christoph Stöckl and Wolfgang Maass. 2020. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *arXiv preprint arXiv:2002.00860* (2020).
[22] Evangelos Stromatias, Daniel Neil, Michael Pfeiffer, Francesco Galluppi, Steve B Furber, and Shih-Chii Liu. 2015. Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms. *Frontiers in neuroscience* 9 (2015), 222.

A EXPERIMENTAL SETUP FOR IDENTICAL FIRING RATES

A toy model is built to explore the optimal identical firing rate range. The data set is CIFAR-10. The network architecture is shown in Table 3. This ANN model only has weights and does not use biases and the batch normalization technique, to avoid the problem of weight-bias mismatch after conversion to SNN. After ANN-to-SNN conversion, the SNNs are built using the standard IF model with

a time constant of 1ms and are normalized to different firing rate ranges as shown in Figure 4. The baseline accuracy is 89.43%.

B DETAILED INFORMATION ABOUT MCR-NORM

In addition to calculating the maximum activation, in each layer, MCR-Norm needs to scan g_n until obtaining the desired firing rate range. The scanning operation is computationally expensive and it is not feasible to scan all possible values. To deal with this problem, we apply two techniques:

(1). Since the scale factor g_n is positively related to the upper bound of the firing rate range in layer n, some techniques such as dichotomy can be used to reduce the computational complexity to get the target firing rate range. After using dichotomy, the deep SNN needs an average of a few thousand of time steps to get the correct scale factor g_n in each layer. It is around the same computational complexity as Spike-Norm.

(2) We can also choose to conduct the scanning operation on the calibration set rather than on the whole training set. This technique can make MCR-Norm a few orders of magnitude faster, and our results show that using the scale factors gotten on the calibration set works well on the test set.

During our experiments, we found that fine-tuning h_n can achieve better accuracy. This may be because the response curve of real spiking neurons is not exactly linear even when the firing rate range is low. Due to this reason, the predefined h_n may not be the optimal value but need a small amount of fine-tuning around.

The idea behind the MCR-Norm to manage the firing rate range layer-wise can be applied to other neuron models as well. When the neuron model is the IF model with an extra reset mechanism and the input is analog input, g_n will be predefined as h_n without the need to be scanned and fine-tuning.