Dropout and DropConnect for Reliable Neuromorphic Inference under Communication Constraints in Network Connectivity

Yasufumi Sakai, Bruno U. Pedroni, *Member, IEEE*, Siddharth Joshi, *Member, IEEE*, Satoshi Tanabe, Abraham Akinin, *Member, IEEE*, and Gert Cauwenberghs, *Member, IEEE*

***Abstract*—Dropout and DropConnect are known as effective methods to improve on the generalization performance of neural networks, by either dropping states of neural units or dropping weights of synaptic connections randomly selected at each time instance throughout the training process. In this paper, we extend on the use of these methods in the design of neuromorphic spiking neural networks (SNN) hardware to improve further on the reliability of inference as impacted by resource constrained errors in network connectivity. Such energy and bandwidth constraints arise for low-power operation in the communication between neural units, which cause dropped spike events due to timeout errors in the transmission. The Dropout and DropConnect processes during training of the network are aligned with a statistical model of the network during inference that accounts for these random errors in the transmission of neural states and synaptic connections. The use of Dropout and DropConnect during training hence allows to simultaneously meet two design objectives: improving robustness of inference to dropped spike events due to timeout communication constraints in network connectivity, while maximizing time-to-decision bandwidth and hence minimizing inference energy in the neuromorphic hardware. Simulations with 5-layer fully connected 784-500-500-500-10 SNN on the MNIST task show a 3.42-fold and 7.06-fold decrease in inference energy at 90% test accuracy, by using Dropout and DropConnect respectively during backpropagation training. Also the simulation with convolutional neural networks on the CIFAR-10 task show a 1.24-fold decrease in inference energy at 60% test accuracy by using Dropout during backpropagation training.**

***Index Terms*—Neuromorphic Hardware, Spiking Neural Networks, Timeout Error, Dropout, DropConnect**

I. Introduction

DEEP neural networks (DNN) can be trained on massive datasets to perform a wide variety of object classification and recognition tasks [1]. One drawback of DNN is that they usually require power hungry hardware such as GPUs. Using neuromorphic hardware for cognitive processing has been proposed as one of the solutions to restrict power under severe resource constraints [2]–[5]. Neuromorphic hardware transmit information through spike events, and hence focus their computational effort on currently active parts of the network, effectively saving power on the rest of the network.

Currently, neuromorphic hardware that realize the operation of spiking neural networks (SNN) composed of millions of integrate-and-fire (IAF) neurons with high efficiency have emerged [6]–[8]. Such neuromorphic hardware implement a reconfigurable neural network on hardware by interconnecting neuron units via routers. Additionally, in typical neuromorphic hardware, as many neurons are serviced by the same router, if two or more events reach the arbiter within the same time step, an arbiter assigns priority randomly to incoming events to avoid the event collisions [9]–[12]. However, reducing the processing rate of the routers for low-power operation of the neuromorphic hardware, causes sporadic dropped spike events due to timeout errors in the transmission.

In order to analyze the effect of spike event drops on the quality of inference in SNN, we created a model of spike event registration and communication in the arbiter and router in the presence of timeout errors in the transmission and incorporated this model with conventional rate-based inference for SNN in the simulation environment [13]. Also we proposed to extend the widespread use of Dropout [14] and DropConnect [15], shown to be effective to improve on the generalization performance of neural networks, for training of DNN in the design of spike-based neuromorphic hardware. The aim of this work is to extend the benefits obtained by Dropout and DropConnect in improved generalization performance to further improvements in the reliability of inference, as impacted by resource constrained errors in network connectivity of optimized hardware. Specifically, we analyzed the effect of Dropout and DropConnect during backpropagation training of a DNN on inference using the trained DNN mapped onto a rate-coded SNN through weight and threshold balancing [16], on a conventional MNIST task [17].

This work extends these results to other datasets beyond MNIST, using the CIFAR-10 dataset [18] in SNN inference to analyze the effect of Dropout during training with DNN. Furthermore, we investigated effects of scaling the weights using two methods: 1) scaling up the normalized weights, and 2) scaling down the weights before normalizing, to correct for weight mismatch between DNN training and SNN inference induced by Dropout during training.

Manuscript received July 31, 2019; revised October 1, 2019; accepted November 3, 2019. Date of publication xxx, date of current version xxx. This paper was recommended by Guest Editor An-Yeu (Andy) Wu.

Y. Sakai and S. Tanabe are with Fujitsu Laboratories Ltd., Kawasaki, Kanagawa, Japan (e-mail: sakaiyasufumi@fujitsu.com; tanabe.s@fujitsu.com)

Y. Sakai, B. U. Pedroni, A. Akinin and G. Cauwenberghs are with the Department of Bioengineering, University of Cailfornia, San Diego, La Jolla, CA 92093, USA (e-mail: bpedroni@ucsd.edu; akinin@gmail.com; gert@ucsd.edu)

S. Joshi is with University of Notre Dame, Notre Dame, IN, USA (e-mail: sjoshi2@nd.edu)

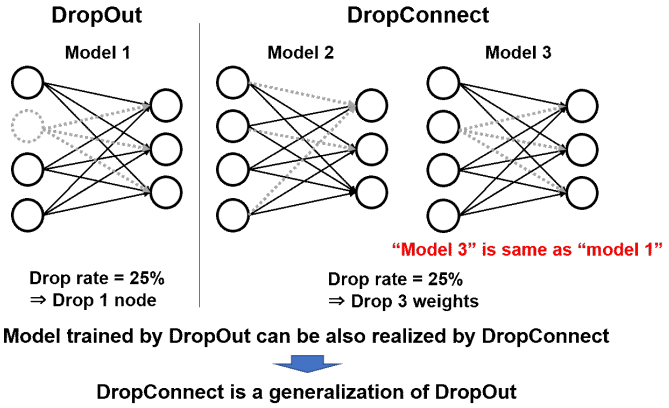


Fig. 1. Dropout and DropConnect correspondence.

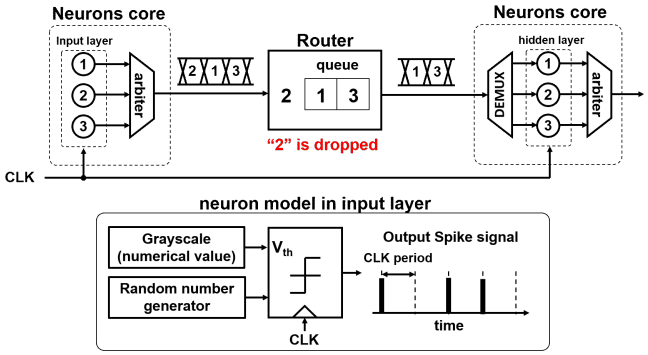


Fig. 2. Model of neural spike event communication path.

II. Background

A. Spiking Neural Networks

Spiking neurons in SNN generate an action potential "spike" represented as a binary variable that goes active only when the accumulated synaptic input contributions exceed a set threshold. The sparse and stochastic activity of the neural units, which only operate when incoming spikes arrive, leads to high energy efficiency in neuromorphic hardware implementation [2].

The accuracy of inference by SNN using normalized weights obtained directly from training a DNN has shown competitive performance approaching the accuracy of inference by DNN [16]. The weights in the SNN for inference are obtained by training the DNN with rectified linear units (ReLU) with zero bias, and normalizing the trained weights by the maximum weight or activation value in each layer. To obtain ReLU equivalence during inference, the SNN uses an IAF model with soft decremental reset rather than the usual hard reset to the rest potential. The dynamics of the membrane potential of spiking neurons is given by [19]:

$$V_j(t+1) = V_j(t) + \sum_i w_{i,j} \cdot s_i(t) \quad (1)$$

$$s_j(t) = (V_j(t) \geq V_{th}) \quad (2)$$

where $s_i(t)$ is the binary-valued spike from neuron i, $V_j(t)$ is the membrane potential of neuron j, $w_{i,j}$ is the weight of synaptic connection from the i^{th} to the j^{th} neuron, and V_{th} is the spike threshold. When the membrane potential $V_j(t)$ is higher than the threshold V_{th}, j^{th} neuron generate the spike $s_j(t)$. Additionally, each cycle all active neurons k generating a spike $s_k(t) = 1$ are subjected to a constant decrement in membrane potential prior to reset:

$$V_k(t) \leftarrow V_k(t) - V_{th} \quad (3)$$

Using the same value for the spike threshold and the membrane decrement V_{th} ensures consistency with the ReLU mean-rate model.

B. Dropout and DropConnect

Dropout [14] and DropConnect [15] are widely used in DNN training owing to their superior generalization performance. The operation of Dropout and DropConnect during training, and differences and correspondences between them, are illustrated in Fig. 1. While Dropout drops the states of neural units randomly selected at each time instance throughout the training process, DropConnect instead drops weights of synaptic connections. As DropConnect drops the weights randomly, there is some chance of dropping only the weights connected to a specific neuron, which becomes equivalent to Dropout as shown on right side of Fig. 1.

III. Effect of Network Connectivity

To analyze the effect of spike event drops on the quality of inference in SNN, we created a model of spike event registration and communication in the arbiter and router in the presence of timeout errors in the transmission shown in Fig. 2. Input data represented by numerical value (e.g. grayscale) are transformed into Poisson synchronized clock signals by comparing value output from a random number generator. In typical applications for computer vision, the firing rates of the Poisson synchronized clock signals are proportional to the numerical value of each image pixel [20]. The probability of generating spikes from multiple neurons within the same time step is non-negligible because of the large number of neurons that are mounted in neuron cores on the neuromorphic hardware. Thus the spikes generated by neurons are gated to the routers for transmission via arbiters to avoid signal collision on the neuron communication path. The arbiter randomly selects spikes output by multiple neurons within the same time step and time-multiplexes the spikes in random order to avoid systematic bias in latency for spikes from specific neurons. The signals time-multiplexed by arbiter are input to the queue in the router in the multiplexed order. The queue operates as first-in first-out (FIFO) mode queue. The signals output from the queue are transmitted to receiver neurons via a demultiplexer. If the data length of the signals input to the queue exceeds the queue depth, the leftover signals exceeding the queue capacity are dropped and never transmitted to receiver neurons, producing a *timeout error*. Importantly, the identities of sender neurons with dropped

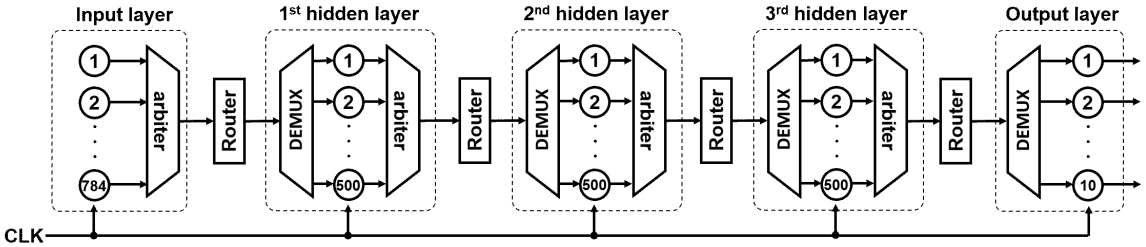


Fig. 3. 5-layer network used for evaluating the effect of timeout communication errors in SNN inference on the MNIST handwritten digits classification task.

events caused by router timeout are random, because the order of the signals input to the queue is randomly determined by the arbiter. The rate of timeout errors in dropped signal transmission increases when the processing rate of routers decreases, or when the input spike rate increases. Hence the incidence of timeout errors can be reduced by increasing the clock signal, which implies a trade-off between accuracy and power consumption in the network communication.

We analyzed the effect of timeout errors in the neuron communication path during inference in SNN, initially without Dropout or DropConnect during DNN training. Figure 3 shows the simulation model for this analysis. We used the MNIST dataset [17] which consists of 28×28 grayscale images, each containing a digit 0-9 for our experiments. The training set consists of 60,000 digits and the test set consists of 10,000 digits. The grayscale values of the MNIST were normalized between 0 and 1. We trained five-layer fully connected neural networks (FCN), 784-500-500-500-10, with ReLU activation function for the neural units, and without bias units. The FCN was trained by backpropagation using stochastic gradient descent, a fixed learning rate of 1, and batch size of 100. After training the FCN, we used the weight normalization procedure of [19] outlined in Section II. Inputs were generated for each neuron in input layer with firing rates proportional to the grayscale pixel value of each MNIST image. We evaluated SNN performance on the test set by collecting statistics over 10,000 parallel models each trained on the training set from random initial conditions.We used the spiking neuron model as shown in Equations (1)-(3). The threshold values of the spiking neurons were fixed to 1. The classification output of the rate-coded SNN was evaluated through spike counting over a variable time interval defining classification bandwidth. The router queue depth was set to 500. The input rate of Poisson spike trains were set to 1k events/sec. Time step was 1 msec. To evaluate the effect of timeout communication erros in SNN inference, the processing rate of routers was set to 1M events/sec without signal drop, and 100k events/sec with signal drop.

Figure 4 shows the simulation results of accuracy and accumulative synaptic operations S_o for SNN inference and Figure 5 shows the spike event activity per time step in the hidden layers and output layer. Synaptic operations, S_o, is derived by summing the product of output spikes and fan-out for each layer. Accumulative S_o is derived by accumulating S_o, for every time step, over the entire test set. Therefore, accumulative S_o in Figure 4 represents the energy consumed

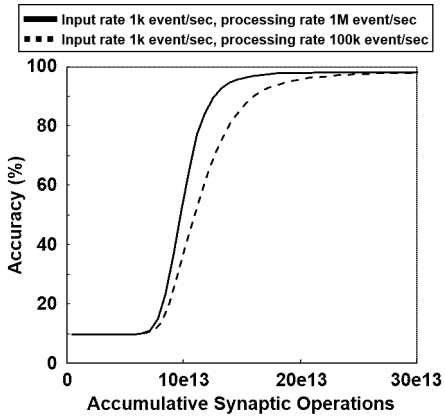


Fig. 4. Effect of timeout errors on SNN inference accuracy, as a function of the accumulative synaptic operations S_o over the entire MNIST test set.

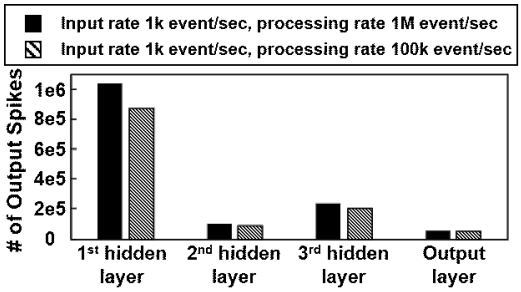


Fig. 5. SNN network activity per time step in hidden and output layers at constant input layer event rate. Time step is 1 msec.

by the neuromorphic hardware to reach a certain accuracy, over the entire test set. In the case of timeout error, shown by the dashed curve in Figure 4, the accumulative S_o to reach convergence at nominal accuracy was larger, hence degrading the energy performance. On the other hand, the final accuracy without signal drop reached 98.17% and the final accuracy with signal drop reached 98.14%. As characteristic for rate coding in SNN, the information present in the sourcing data represented in numerical format such as grayscale is conveyed to the network more accurately when the number of spikes input to the network increases [20]. Hence, with sufficiently elapsed time, until enough spikes are input to the network, the final accuracy during inference by rate coding reaches almost same value whether signals are dropped or not.

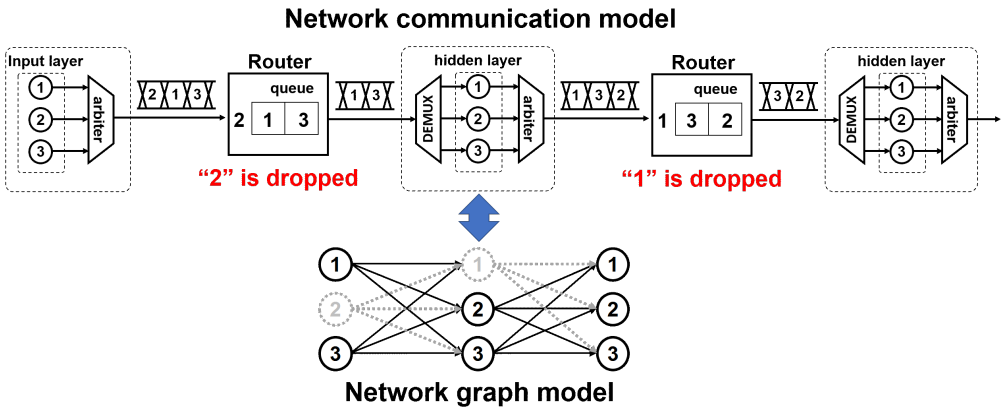


Fig. 6. Network communication model with timeout signal drop at routing nodes, and equivalent network graph model with Dropout/DropConnect.

IV. Dropout and DropConnect for Reliable Neural Computing

We propose a method to improve the degradation of the convergence S_o caused by timeout error during SNN inference by using Dropout and DropConnect during training. Recall that Dropout and DropConnect operate by dropping states of neural units or weights of synaptic connections, randomly selected at each time instance throughout the training process in DNN. Since signals transmitting from sender neurons in the SNN are dropped randomly by the function of arbiter when timeout error occur at routers, the statistical model of neuromorphic hardware during inference in SNN can be aligned with the Dropout processes during DNN training as shown in Fig. 6. Furthermore, since all network models of Dropout during training are subsumed by equivalent network models of DropConnect during training as shown in Fig. 1, the statistical model of neuromorphic hardware during inference can also be aligned with the DropConnect processes during training. Importantly, by using Dropout and DropConnect during DNN training, since the network is trained with the same configuration as at inference by SNN, an improvement in the reliability of inference by SNN with signal drop is expected. We analyzed the effect of using weights trained with Dropout and DropConnect in SNN inference through simulations on the MNIST and the CIFAR-10 tasks.

A. MNIST task with Fully Connected Neural Networks

The fundamental setup of analyzing the effect of using weights trained with Dropout and DropConnect in inference for SNN is the same as the analysis in Section III. We used the MNIST dataset and five-layer FCN, 784-500-500-500-10. The drop rates of Dropout and DropConnect which define the rate of dropping nodes and weights during training were set to (0, 10, 20, 30, 40, 50). Dropout and DropConnect were applied to only the neurons of hidden layers and not applied to the neurons of the input layer. Note that the convergence time of SNN inference, *i.e.* inference energy, increases as the final accuracy increases [19]. In order to isolate the influence of the difference between using Dropout and DropConnect during training, we trained the network so that the final accuracy of SNN inference is approximately at the same level across all conditions in this analysis. Table I shows the results of final accuracy of SNN inference for each condition.

TABLE I
Accuracy of SNN inference using weights trained with/without Dropout/DropConnect.

Drop rate (%)	Accuracy w/ Dropout (%)	Accuracy w/ DropConnect(%)
10	98.15	98.15
20	98.14	98.24
30	98.22	98.14
40	98.17	98.20
50	98.23	98.20
0	98.17*	

*Drop rate = 0% means training w/o Dropout and DropConnect.

Figure 7 (left) shows the simulation results of SNN inference accuracy using weights trained by Dropout. In order to compare the convergence S_o among all conditions, the results of accumulative S_o when the accuracy reached 90% were compared. The convergence S_o using weights trained using Dropout was about 3.42 times lower at the maximum than when not using drop methods during training. In this analysis, the degradation of convergence S_o was most improved when the Dropout rate was 50%. Figure 7 (right) shows the simulation results of SNN inference accuracy using weights trained by DropConnect. The convergence S_o for weights trained using DropConnect was about 7.06 times lower at the maximum than when not using DropConnect during training. In this analysis, the degradation of convergence time was most improved when the DropConnect rate was 50%.

Figure 8 shows the total number of spikes per time step input to hidden and output layers. By using trained weights with Dropout and DropConnect, the number of spikes increased in all conditions comparable with the condition without both drop methods. In addition, the conditions of drop rate with the most improved convergence S_o was aligned with the conditions of drop rate with the largest number of spikes. The reason of increasing spikes when using Dropout and DropConnect during training is that the value of the weights for SNN inference is large. Figure 9 shows the distribution of the

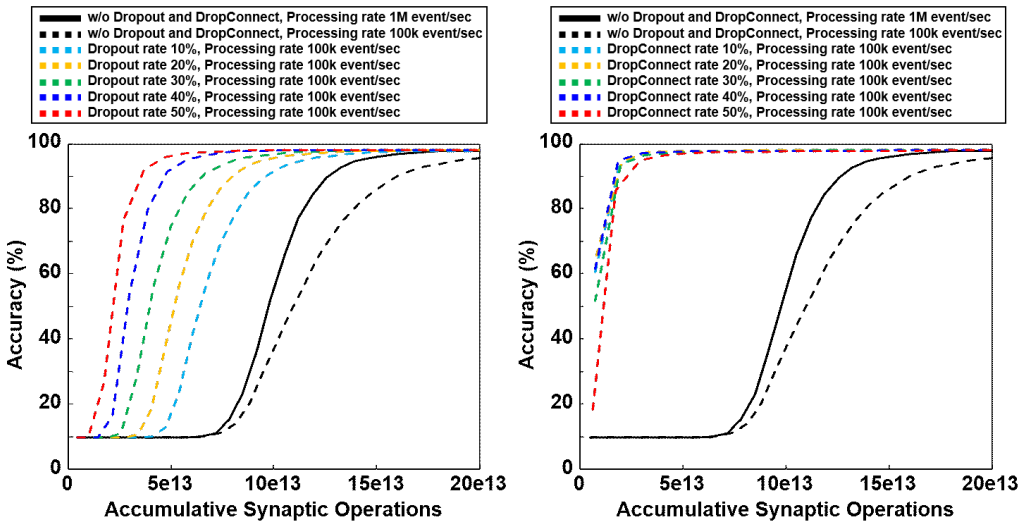


Fig. 7. SNN inference accuracy, as a function of accumulative synaptic operations S_o over the MNIST test set, using weights trained with Dropout (left) and DropConnect (right).

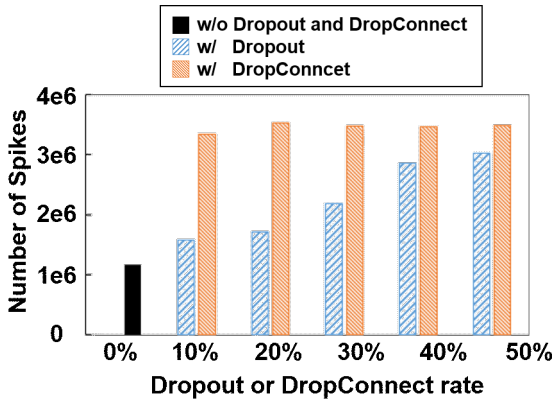


Fig. 8. The total number of spikes in hidden and output layers per time step at varying Dropout or DropConnect rate during training consistent with transmission timeout signal drop rate during inference.

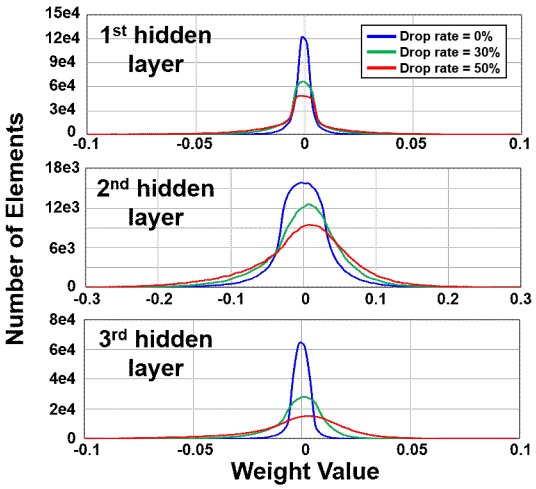


Fig. 9. The distribution of the weights for SNN inference trained by using Dropout.

weights value using in SNN inference. When using the weights trained by using Dropout and DropConnect to SNN inference, the weights are multiplied the scaling factor less than unity "1" determined by drop rate [14]. For example, the weights are multiplied $\times 0.8$ when the drop rate is 20% for inference. Note that as the weights for SNN inference are obtained by normalizing the weights trained in DNN by the maximum weight and activation value in each layer [16], in case of very small maximum weights trained in DNN before normalization, the weights for SNN inference after normalization becomes large. As a result, since the weights before normalization are scaled down when using Dropout and DropCoonect, the weights after normalization become large under all conditions using Dropout and DropConnect, hence increasing the number of spikes. Figure 10 shows the accumulative S_o to reach 90% accuracy. This result shows that the use of Dropout and DropConnect realize improving inference energy in the neuromorphic hardware.

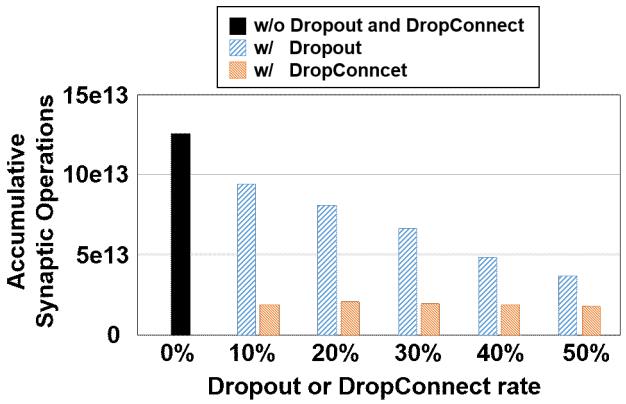


Fig. 10. Accumulative synaptic operations S_o to reach 90% accuracy across the MNIST test set.

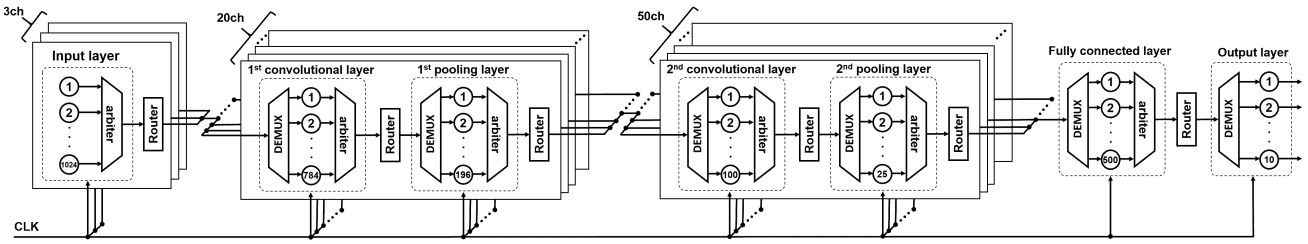


Fig. 11. Convolutional neural networks used for evaluating the effect of timeout communication errors in SNN inference on the CIFAR-10 image classification task.

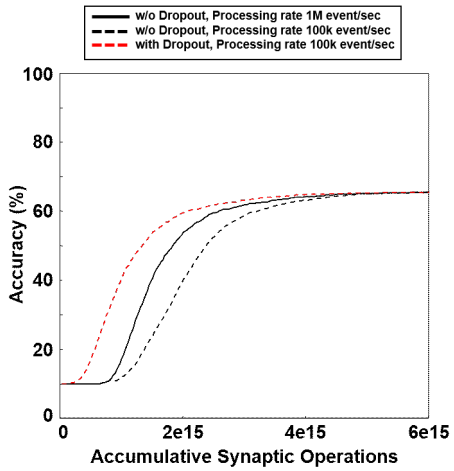


Fig. 12. Effect of timeout errors on convolutional SNN inference accuracy, as a function of accumulative synaptic operations S_o across the CIFAR-10 task.

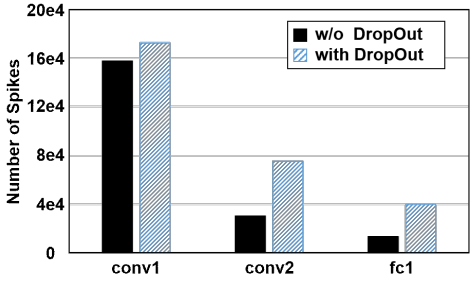


Fig. 13. Convolutional SNN network activity on the CIFAR-10 task at 1k events/sec input layer event rate.

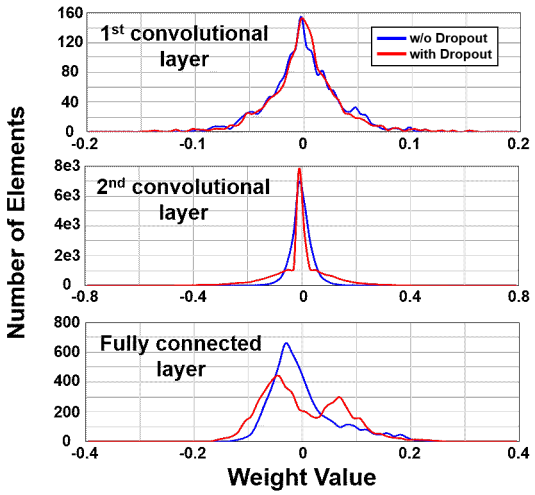


Fig. 14. Distribution of weights for convolutional SNN inference on the CIFAR-10 task trained using Dropout.

B. CIFAR-10 task with Convolutional Neural Networks

Next, we analyzed the effect of using the weights trained with Dropout in SNN inference by using convolutional neural networks (CNN) with the CIFAR-10 dataset [18] to check whether the effects of improving the SNN inference energy using Dropout extend to other datasets beyond MNIST. Figure 11 shows the simulation model for the analysis using the CIFAR-10 dataset with CNN. The CIFAR-10 dataset which consists of 32×32 RGB images, each containing 10 kinds of image. The training set consists of 50,000 images and the test set consists of 10,000 images. The RGB values of the CIFAR-10 were normalized between 0 and 1. The architecture of CNN used in this analysis was input(32×32, 3ch)-conv1(5×5, 20ch)-pool1(2×2)-conv2(5×5, 50ch)-pool2(2×2)-fc1(500)-output(10). The activation function of the trained CNN was ReLU with no bias. The stride of conv1 and conv2 was 1, and the stride of pool1 and pool2 was 2. The padding was not used. We used average pooling. The CNN was trained by backpropagation using momentum SGD and the batch size of 128. The initial learning rate was 0.5 and the learning rate was multiplied by 0.5 every 70 epochs. The momentum coefficient was 0.9. We trained the weight under two conditions with and without Dropout. The drop rates of Dropout were set to 25% for conv2 layer and 50% for fc1 layer. In SNN inference, We used average pooling model for SNN used in [16]. We assumed that the model shown in Figure 11 only process 1 batch test data in this simulation, hence all of test data were processed parallelly by 10,000 models.

Figure 12 shows the simulation results of SNN inference accuracy. Figure 13 shows the total number of spikes in each layer and figure 14 shows the distribution of the weights value in kernel of convolutional layers and fully connected layer used in SNN inference. The final accuracy in SNN inference without and with Dropout conditions reached 66.1% and 66.0% respectively. To compare inference energy among each condition, we compared S_o to reach 60% accuracy at convergence. This S_o using the weights trained with Dropout was about 1.24 times lower at convergence than when not using Dropout during training.

TABLE II
ACCURACY OF SNN INFERENCE USING THE WEIGHTS SCALED UP AFTER NORMALIZING.

Dropout rate (%)	scaling factor	Accuracy (%)
0	×0	98.17
50	×0	98.23
0	×3	96.33
0	×4	95.86
0	×5	95.57

*Drop rate = 0% means training w/o Dropout and DropConnect.

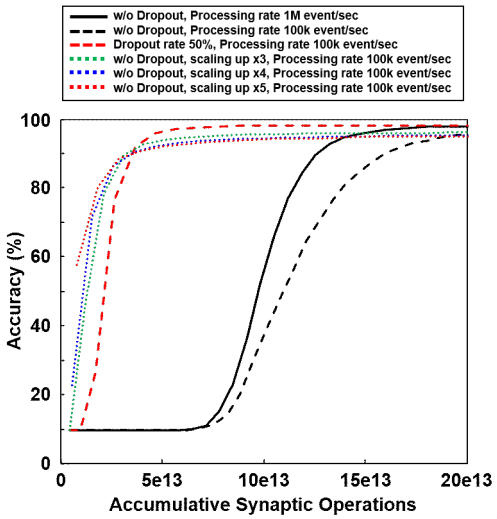


Fig. 15. SNN inference accuracy on MNIST by scaling up the weights after normalizing.

V. COMPARISON WITH SIMPLE WEIGHT SCALING

Since the energy of SNN inference can be reduced by using larger values for weights trained with Dropout and DropConnect, it seems that the energy of SNN inference can be reduced simply by scaling up the weights, without training with Dropout and DropConnect. In order to confirm this assumption, we used the weight which are simply scaled up for the simulation of SNN inference. In addition, since we normalized the weights by [16] for SNN inference as described in Section II-A, we also used the weight scaled down before normalizing for the simulation.

A. Scaling Up the Normalized Weights

First, we analyzed the effect of using the normalized weights simply scaled up. The fundamental simulation setup for this analyzing was same as in Section IV-A. We used 5-layer FCN and MNIST dataset. The scaling factors of the weighs which were multiplied by the normalized weights trained without using Dropout and DropConnect were ×3, ×4 and ×5.

Figure 15 shows the accuracy of SNN inference and Figure 16 shows the total number of spikes per time step input to hidden and output layers in each condition. Although the convergence S_o was reduced due to increasing the number of spikes, the final accuracy of SNN inference was degraded in each condition as shown in Table II. The reason of degraded the final accuracy is that the ReLU characteristics of the

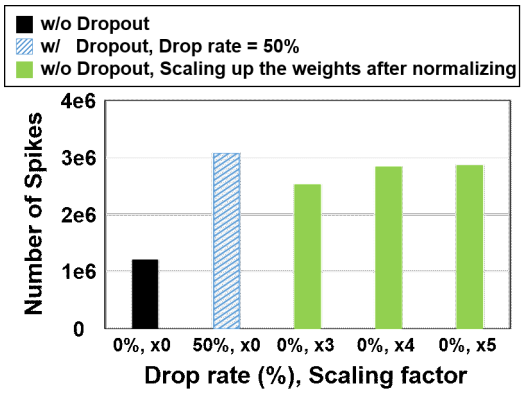


Fig. 16. The total number of spikes in hidden and output layers per time step at varying scaling up factor for normalized weights.

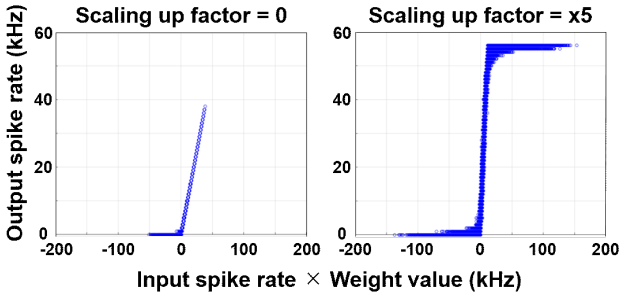


Fig. 17. Input-output spike rate transfer function for scaled-up weights after normalization.

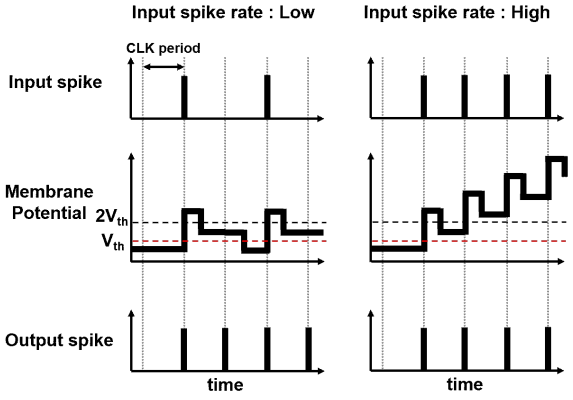


Fig. 18. Intuitive explanation for the nonlinearity in spiking neuronal response for large scaling-up factor. Increasing the input spike rate does not produce proportionally larger output spike rate due to each input spike pushing the membrane potential well over the threshold, permitting the output to spike in consecutive cycles.

TABLE III
SNN INFERENCE ACCURACY ON MNIST BY SCALING DOWN THE WEIGHTS BEFORE NORMALIZING.

Dropout rate (%)	scaling factor	Accuracy (%)
0	×0	98.17
50	×0	98.23
0	×0.7	97.45
0	×0.6	97.37
0	×0.5	97.08

*Drop rate = 0% means training w/o Dropout and DropConnect.

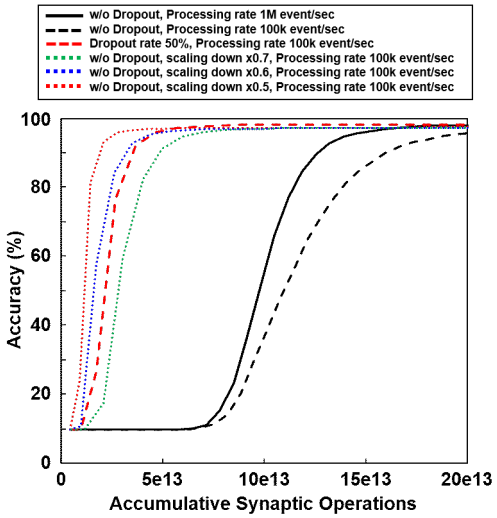


Fig. 19. Results of SNN inference using the weights scaled down before normalizing.

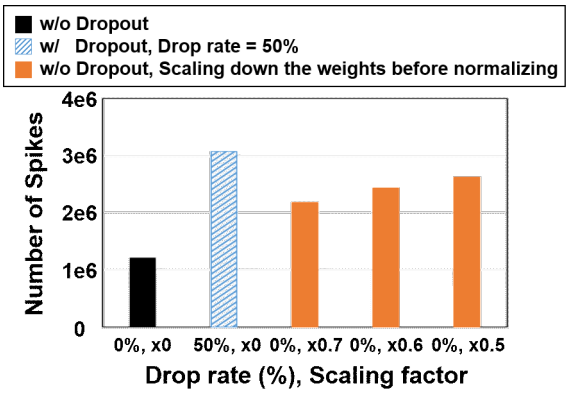


Fig. 20. The total number of spikes in hidden and output layers per time step at varying scaling down factor for the weights before normalizing.

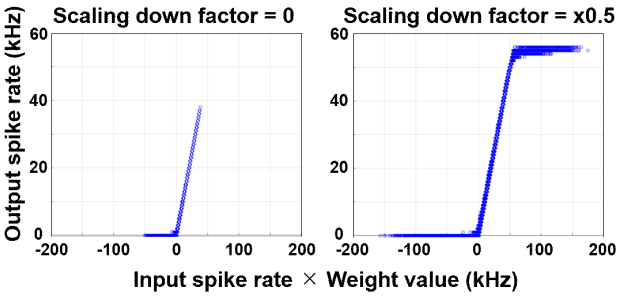


Fig. 21. Input-output spike rate transfer function for scaled-down weights before normalizing.

spiking neuron showed non-ideal characteristics. Figure 17 shows the relationship between input spike rate and output spike rate of spiking neurons in this simulation. When the scaling factor is small, the spiking neurons show ideal ReLU characteristics. But when the scaling factor is large, the ReLU characteristics of spiking neurons show non-linearity in the area of the large value of multiplied the input spike rate and the value of weight. The reason why the output spike rate does not increase substantially under increasing product of input spike rate and weight can be understood as follows. The output spike rate is limited by the clock signal which drives the spiking neuron due to the large value of weights. The value of weights becomes large when the value of scaling factor is large. When the value of weights becomes two times or greater than the threshold V_{th} of the spiking neurons shown in Equations (2)-(3), since the membrane potential after spike input to the neurons becomes greater than $2\times V_{th}$ as shown in Figure 18, the membrane potential does not become smaller than V_{th} even though the membrane potential is subtracted by V_{th} following Equation (3). In this case, even if the spike is not input to the neurons at the next time step, the neurons output the spike because the membrane potential is greater than V_{th}. Also, the larger the value of weights, since the membrane potential is easy to keep the value higher than V_{th} even if the spike is not input to the neurons, the higher the probability that the neurons output the spike every time step. As a result, even if the input spike rate is further increased when the neurons keep the state of outputting the spike every time step, since each neuron output spike only once per each time step, the output spike rate of the neurons cannot be increased in accordance with the input spike rate.

Therefore, when the spiking neurons keep the linearity, since the arithmetic operations of the neural networks used in this simulation are only inner product and ReLU operations, the value of the loss function is simply scaled when the weights in the neural networks are scaled and the classification result does not change. However, when the ReLU characteristics of spiking neurons show the non-ideal characteristics, the classification result changes because the value of the loss function is nonlinear with respect to the scaling factor of weight. Therefore, even when scaling up the weights after normalizing, the S_o of SNN inference reduces due to increasing number of spikes, while the final accuracy of the SNN inference is degraded.

B. Scaling Down the Weights before Normalizing

The reason for increasing the value of the weights trained with Dropout and DropConnect used in SNN inference is that the weights before normalizing are multiplied by the scaling factor less than 1 decided by the drop rate. Therefore, if the weights scaled down before the normalization are used for SNN inference, it is assumed that the S_o of

SNN inference is reduced. To analyze the effect of using the weights scaled down before normalization, we scaled down the weights trained without Dropout. After that, we normalized the scaled weight for SNN inference. The simulation setup for this analyzing was same as previous sub-Section V-A. The architecture and dataset used for this simulation were 5-layer FCN and MNIST. The scaling factors multiplied by the weights before normalizing were $\times 0.7$, $\times 0.6$ and $\times 0.5$.

Figure 19 shows the accuracy of SNN inference. Figure 20 shows total number of spikes per time step in hidden and output layers using the scaled down weight. The final accuracy of the SNN inference is shown in Table III. The relationship between input spike rate and output spike rate of spiking neurons in this simulation is shown in Figure 21. Although the S_o was reduced due to increasing the number of spikes, the final accuracy of SNN inference was degraded in each condition due to the non-ideal ReLU characteristics of the spiking neurons.

In both simple weight scaling methods, the convergence S_o of SNN inference was improved but the final accuracy was degraded. On the other hand, our proposed methods using Dropout and DropConnect during training can easily obtain a neural networks model that reduces convergence S_o without degradation of final accuracy. Therefore, proposed methods can be applied easily to compensate for the degradation of SNN inference energy due to the timeout errors generated by the router.

VI. CONCLUSION

Signal drop due to router timeout increases the SNN inference energy. We showed that this degradation in inference energy can be minimized by training the network using the Dropout and DropConnect frameworks. Increased generalization in the architecture of neuromorphic hardware fomented by these training frameworks increases the robustness of the network to communication reliability faults. In this paper we have demonstrated that, aside from improving accuracy, the robustness benefits of Dropout and DropConnect contribute to maximizing time-to-decision bandwidth and minimizing inference energy of a neuromorphic hardware by allowing lower speed-reliability operation during inference.

REFERENCES

[1] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[2] G. Indiveri, E. Chicca, and R. J. Douglas, "A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE transactions on neural networks*, vol. 17, no. 1, 2006.

[3] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Circuits and systems (ISCAS), proceedings of 2010 IEEE international symposium on*. IEEE, 2010, pp. 1947–1950.

[4] M. Khan, D. Lester, L. Plana, A. Rast, X. Jin, E. Painkras, and S. Furber, "Spinnaker: mapping neural networks onto a massively-parallel chip multiprocessor," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 2849–2856.

[5] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. Van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience*, vol. 5, p. 73, 2011.

[6] 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.

[7] B. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. Arthur, P. Merolla, , and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for largescale neural simulations," in *Proceedings of the IEEE*, vol. 102, no. 5. IEEE, 2014, pp. 699–716.

[8] J. Park, T. Yu, S. Joshi, C. Maier, and G. Cauwenberghs, "Hierarchical address event routing for reconfigurable large-scale neuromorphic systems," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2408–2422, 2017.

[9] M. A. Sivilotti, "Wiring considerations in analog vlsi systems, with application to field-programmable networks." 1991.

[10] M. Mahowald, *An analog VLSI system for stereoscopic vision*. Springer Science & Business Media, 1994, vol. 265.

[11] S. R. Deiss, R. J. Douglas, A. M. Whatley *et al.*, "A pulse-coded communications infrastructure for neuromorphic systems," *Pulsed neural networks*, pp. 157–178, 1999.

[12] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, 2000.

[13] Y. Sakai, B. U. Pedroni, S. Joshi, A. Akinin, and G. Cauwenberghs, "DropOut and DropConnect for reliable neuromorphic inference under energy and bandwidth constraints in network connectivity," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2019, pp. 76–80.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," in *The Journal of Machine Learning Research*. ACM, 2014, pp. 1929–1958.

[15] L.Wan, M.Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using DropConnect," in *In Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. ACM, 2013, pp. 1058–1066.

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

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86. IEEE, nov 1998, pp. 2278–2324.

[18] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[19] P. U. Diehl, B. U. Pedroni, A. Cassidy, P. Merolla, E. Neftci, and G. Zarrella, "Truehappiness: Neuromorphic emotion recognition on truenorth," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 4278–4285.

[20] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in neuroscience*, vol. 7, p. 178, 2013.

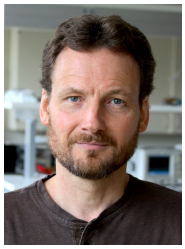
Yasufumi Sakai received the B.E. and M.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 2005 and 2007 respectively. In 2007, he joined Fujitsu Laboratories Ltd., Kawasaki, Japan, where he engaged in the research and design of CMOS analog, RF circuits, and high-speed CMOS interconnect circuits until 2017. From 2017 to 2018, he was a Visiting Scholar with the University of California, San Diego, La Jolla CA, where he researched low-power neuromorphic computer system. In 2018, he was back to Fujitsu Laboratories, and engaged in reserch of efficient training algorithms for neural networks.

Bruno U. Pedroni received the B.S. and M.S. degrees in electrical engineering from the Federal Technological University of Paraná, Curitiba, Brazil, in 2005 and 2009, respectively, and the Ph.D. degree in bioengineering from the University of California San Diego, La Jolla, USA, in 2019. From 2005–2012, he was a Software Developer and Telecommunications Engineer for Siemens Brasil and the Paraná State Power Company. He is currently a Postdoctoral Researcher in the Institute for Neural Computation (INC) at the University of California San Diego, where he continues his study and work on designing large-scale digital learning neuromorphic systems and developing spiking energy-based learning algorithms using biologically-relevant neural elements as computational primitives.

Siddharth Joshi (S' 14-M'17) is an assistant professor in the department of Computer Science and Engineering and the department of Electrical Engineering at the University of Notre Dame, IN. He received the B.Tech. degree in information and communication technology from the Dhirubhai Ambani Institute of Information and Technology (DA-IICT), Gandhinagar, India, in 2008, and the M.S. and PhD degrees in electrical and computer engineering from the University of California, San Diego, CA, USA, in 2011 and 2017 respectively. His research focuses on the design and implementation of low-power architectures and adaptive circuits enabling machine intelligence in highly resource constrained environments.

Satoshi Tanabe received the M.E. degree in Electrical and electronic engineering from Tokyo University, Tokyo, Japan,in 2008. In 2008, he joined Fujitsu Laboratories Ltd., Kawasaki, Japan. He engaged in the research and design of low-power SOC until 2013. Since 2013,he has been engaged in research and development of signal-processing and computer vision.

Abraham Akinin (S'13) received the B.S. degree in biomedical engineering and physics from the University of Miami, Coral Gables, FL, USA, in 2010. He is currently working toward the Ph.D. degree at the Bioengineering Department and the Institute for Neural Computation, University of California San Diego, La Jolla, CA, USA. Since 2018 he has been with Nanovision Biosciences as Bioelectronics Design Engineer developing a next generation retina prosthesis. His research interests include design of closed-loop neuroprosthetics and integrated biomedical instrumentation to restore sensory and cognitive function.



Gert Cauwenberghs (S'89-M'94-SM'04-F'11) is Professor of Bioengineering and Co-Director of the Institute for Neural Computation at UC San Diego, La Jolla CA. He received the Ph.D. degree in Electrical Engineering from California Institute of Technology, Pasadena in 1994, and was previously Professor of Electrical and Computer Engineering at Johns Hopkins University, Baltimore, MD, and Visiting Professor of Brain and Cognitive Science at Massachusetts Institute of Technology, Cambridge. He co-founded Cognionics Inc. and chairs its Scientific Advisory Board. His research focuses on micropower biomedical instrumentation, neuron-silicon and brain-machine interfaces, neuromorphic engineering, and adaptive intelligent systems. He received the NSF Career Award in 1997, ONR Young Investigator Award in 1999, and Presidential Early Career Award for Scientists and Engineers in 2000. He was a Francqui Fellow of the Belgian American Educational Foundation, and is a Fellow of the American Institute for Medical and Biological Engineering. He served IEEE in a variety of roles including as Distinguished Lecturer of the IEEE Circuits and Systems Society, as General Chair of the IEEE Biomedical Circuits and Systems Conference (BioCAS 2011, San Diego), as Program Chair of the IEEE Engineering in Medicine and Biology Conference (EMBC 2012, San Diego), and as Editor-in-Chief of the IEEE Transactions on Biomedical Circuits and Systems.