An Efficient Approach to Boosting Performance of Deep Spiking Network Training

Seongsik Park[1], **Sang-gil Lee**[1], **Hyunha Nam**[1], **and Sungroh Yoon**[1,2*]
[1]Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea
[2]Neurology and Neuroscience, Stanford University, Stanford CA94305, USA
sryoon@snu.ac.kr

Abstract

Nowadays deep learning is dominating the field of machine learning with state-of-the-art performance in various application areas. Recently, spiking neural networks (SNNs) have been attracting a great deal of attention, notably owning to their power efficiency, which can potentially allow us to implement a low-power deep learning engine suitable for real-time/mobile applications. However, implementing SNN-based deep learning remains challenging, especially gradient-based training of SNNs by error backpropagation. We cannot simply propagate errors through SNNs in conventional way because of the property of SNNs that process discrete data in the form of a series. Consequently, most of the previous studies employ a workaround technique, which first trains a conventional weighted-sum deep neural network and then maps the learning weights to the SNN under training, instead of training SNN parameters directly. In order to eliminate this workaround, recently proposed is a new class of SNN named deep spiking networks (DSNs), which can be trained directly (without a mapping from conventional deep networks) by error backpropagation with stochastic gradient descent. In this paper, we show that the initialization of the membrane potential on the backward path is an important step in DSN training, through diverse experiments performed under various conditions. Furthermore, we propose a simple and efficient method that can improve DSN training by controlling the initial membrane potential on the backward path. In our experiments, adopting the proposed approach allowed us to boost the performance of DSN training in terms of converging time and accuracy.

1 Introduction

Recently, a breakthrough in deep learning [16, 14] has shown that, with carefully selected learning rules and regularization methods, multiple non-linear representations from deep artificial neural networks can achieve state-of-the-art performance in various tasks including image recognition [13], natural language processing [3, 25, 1], and bioinformatics [2, 17, 24, 21].

While the conventional deep neural networks (DNNs) excel at the problems on many different areas, one of their major drawbacks is that they are computationally expensive: A single neuron in the layer requires all values of the neurons from the previous layer to calculate a weighted sum, which needs a huge amount of multiplicative operations. Thus, the current deep learning frameworks rely heavily on GPU-based parallel computation, making it infeasible, without some compromises, to apply them to mobile devices or (even power-efficient) chips.

In addition, the method of backpropagation, the most popular method for training DNNs, requires a differentiable activation function in each neuron, flowing information between neurons using real

*To whom correspondence should be addressed.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

numbers. However, from a neurological viewpoint, this backpropagation technique is not an efficient mechanism. Information is encoded with "all-or-none" action potentials, such as the "spike train," and only a small subset of neurons propagate information by firing spikes.

In this regard, spiking neural networks (SNNs) can provide a much more power-efficient way of implementing artificial neurons than conventional DNNs [20, 19] leveraged by their compact and sparse activity profiles. Triggered by the recent success of DNNs, the research on SNNs is gaining momentum with renewed interest: Various neuronal models [10, 7], spike encoding methods [8, 11], and learning rules [12, 6] have been proposed.

Many of existing approaches train and convert a conventional DNN to its SNN counterpart by replacing the activation functions (mostly ReLU) in the trained DNN with spiking neurons, producing the SNN. For instance, O'Connor et al. [23] mapped a trained deep belief network to an SNN by using the Siegert mean-firing-rate approximation model for integrate-and-fire spiking neurons. Diehl et al. [5] trained a DNN with the stochastic gradient descent (SGD) and then converted it to an SNN based on parameter optimization. Zambrano and Bohte [26] used the asynchronous pulsed sigma-delta coding for spike trains with adaptive dynamic range in the membrane potential.

While these pioneering methods closed the gap of performance between conventional DNNs and SNNs, they could not train SNNs with the spikes by backpropagation with SGD. Although Lee et al. [18] reported that they could train SNNs with a differentiable activation function, their work incurred additional computation overhead for handling activation and gradients, thus discouraging the deployment of their method for neuromorphic hardware with limited resources.

To address the limitation of these techniques, O'Connor and Welling [22] recently proposed deep spiking networks (DSNs), one of the first multi-layer SNNs that can be trained with spikes by backpropagation with SGD. They reported that their DSN-based deep neural network delivered performance comparable to conventional weighted-sum DNNs with similar capacity. In addition, they proposed the method of fractional stochastic gradient descent (F-SGD), which is a training algorithm to improve SNN performance. Furthermore, most parts of their method require only additive operations rather than multiplicative ones, making their network power-efficient and hardware-friendly for implementation. The authors also made an important discovery that adjusting the initial value of the membrane potential on the backward path of an SNN could significantly affect the quality of training.

Inspired by this intriguing discovery, in this paper, we thoroughly analyze the impact of initializing the membrane potential on the overall performance of training an SNN. Based on this analysis, we propose a new, efficient approach that can boost the performance of training SNNs. Our approach is grounded on the method of precharging membrane potentials in the previous work, effectively advancing it by reducing the side effects of precharged potentials as training progresses. Furthermore, we share our neurological insight gained by the observations of network behavior: We can interpret the performance boosts as originating from the long-term potentiation of synapses, while we can represent the negative impact of the initial membrane potentials as a trace of background activities interfering with memory maintenance in synapses.

2 Methods and Experiments

As baseline training algorithm, we present the original version of the training methodology for DSNs [22] in Algorithm 1.

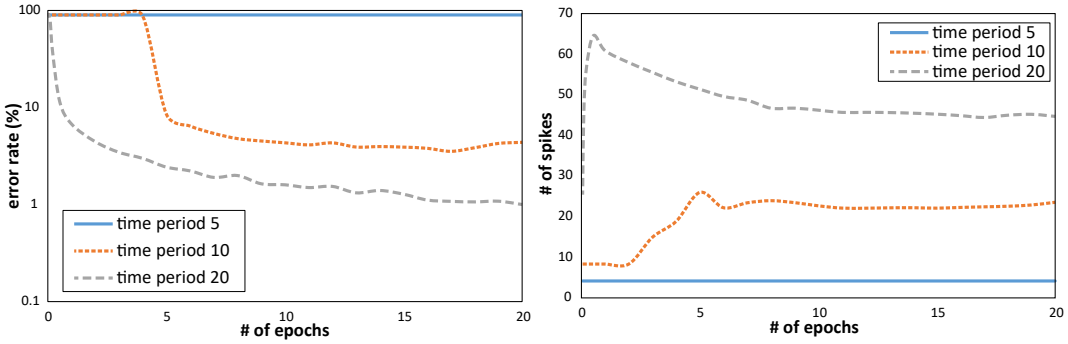
As outlined in this algorithm, the training procedure of DSNs is different from that of conventional DNNs. For each training example, we extract a set of spike trains (series of spikes) t times and use each spike train to train a DSN. In this paper, we call the quantity t the ***time period*** for training (in other papers, the term "time steps" is also frequently used). The spikes in a spike train are propagated forward and backward through the network, while the gradient information for weight updates is accumulated. After each time period is over, the parameters in the DSN are updated using the accumulated gradient information.

The properties of information encoding and propagation in SNNs make the time period for training data have a critical role. The neurons in an SNN would receive more stimuli as the time period increases since the input spike train is proportional to it. The neurons can thus have more chances to fire a spike, which would eventually improve the performance of the SNN.

Algorithm 1: Training in Deep Spiking Networks [22]

E : # of epochs, D : # of training data, T : Time period
phi_{bwd} : Membrane potentials on the backward path

for *e in E* **do**
  for *d in D* **do**
    $phi_{bwd} \leftarrow$ INITIALIZATIONFUNCTION;
    $del \leftarrow 0$;
    for *t in T* **do**
      $s_t \leftarrow$ SAMPLINGINPUTDATA(d);
      $err_t \leftarrow$ FORWARDPATHOPERATION(s_t);
      $del_t \leftarrow$ BACKWARDPATHOPERATION(err_t);
      $del \leftarrow del + del_t$;
    end
    UPDATEPARAMETERS(del);
  end
end



(a) (b)

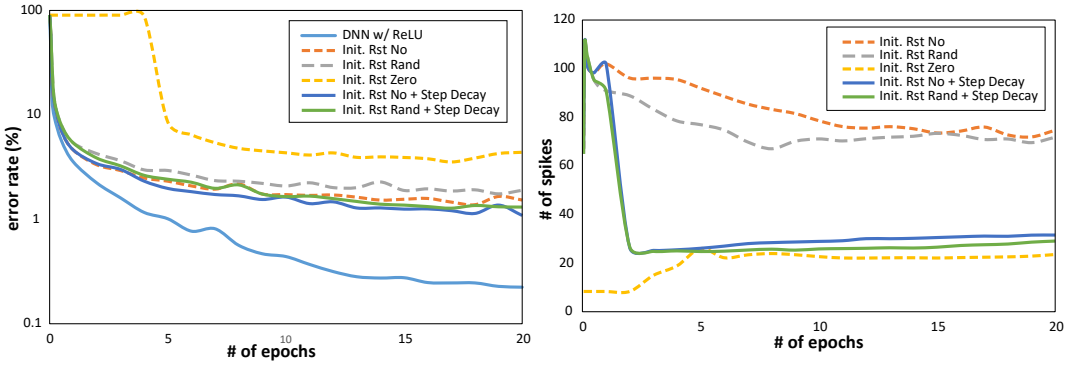
Figure 1: (a) Training error (%) and (b) the number of the backward spikes obtained from a DSN trained by SGD with the MNIST data using three different time periods for training (the initialization function of membrane potentials on the backward path was reset to zero).

2.1 Effects of Time Periods and Initial Potential on SNN Training

To verify the previous work and validate the effectiveness of the proposed approach, we performed a diverse set of experiments using the code under the same default hyper-parameters and conditions from the original work (available at https://github.com/petered/spiking-mlp).

Figure 1a shows the training results (error rates versus epochs) obtained from training an SNN implemented as a DSN using SGD with the MNIST data [15] for three different time periods. Evidently, we can observe that the time period used for training significantly affects training error rates. As expected, the longer a time period used for training the SNN, the more accurate result we could get. In the extreme case with an (vastly insufficient) time period of 5, the SNN could not be trained at all because of the lack of spikes, as shown in Figure 1b. Especially, in that case, the number of backward spiking instances was insufficient to stimulate a neuron on the backward path to fire a spike. As a result, there was no change in terms of the amount of spiking regardless of the number of epochs executed. Obviously, we should use a time period that is long enough to avoid this unsatisfactory result. On the other hand, as the time period is strongly related to the total training time, it is important to make the time period as short as possible for assuring efficient training.

The main problem encountered when training an SNN with insufficient time periods is that the neurons in the SNN cannot properly propagate error signal backwards. Each neuron in an SNN integrates the spikes from other neurons and fires a spike only when its membrane potential is greater than a certain threshold. Thus, if the neuron could not obtain a sufficient number of spikes during the



(a) (b)

Figure 2: (a) Training error (%) and (b) the number of the backward spikes during the training of a DSN using SGD with the MNIST data (decay constant $C_d = 1$ for the proposed step-decay method).

time period of a sampled input spike train, it will not fire. Even worse, as an SNN is getting deeper, this phenomenon of disappearing spikes between neurons would be more serious just as the vanishing gradient problem in gradient descent.

To resolve this problem and to improve training with relatively short time periods, the method of backwards quantization, a technique of precharging membrane potentials on the backward path, was introduced in the original DSN paper [22]. The main idea is that we should precharge each membrane potential on the backward path at the beginning of each training, in order to make it easily exceed the threshold with fewer spikes. By applying this method, the spikes for error backpropagation will be fired more easily. In other words, errors will be be propagated more effectively through the backward path. Consequently, we can potentially boost the effectiveness of training in terms of convergence time and accuracy through this precharge method.

O'Connor and Welling [22] empirically suggested three types of initialization methods: "no reset," "uniform random reset," and "zero reset." In the "no reset" approach, the membrane potential from the previous training sample is maintained at the beginning of training using the current sample without resetting the potential. The "uniform random reset" and "zero reset" approaches refer to resetting the membrane potentials at the onset of each training with uniformly sampled random values and all zeros, respectively. Although the "zero reset" approach is the one that is biologically most plausible, this option makes the training of an SNN inefficient (as seen before, a DSN needs an adequate amount of time periods to deliver satisfactory performance). Consequently, the "no reset" or "uniform random reset" approaches have been commonly applied to DSN training for precharging membrane potentials on the backward path.

2.2 Our Proposal: Step-Decay Precharge Method

In a typical SNN, the precharged membrane potential before the beginning of training would be considered as noise, which causes faulty learning results for the current training sample. The way of precharging the membrane potential therefore critically affects the training result.

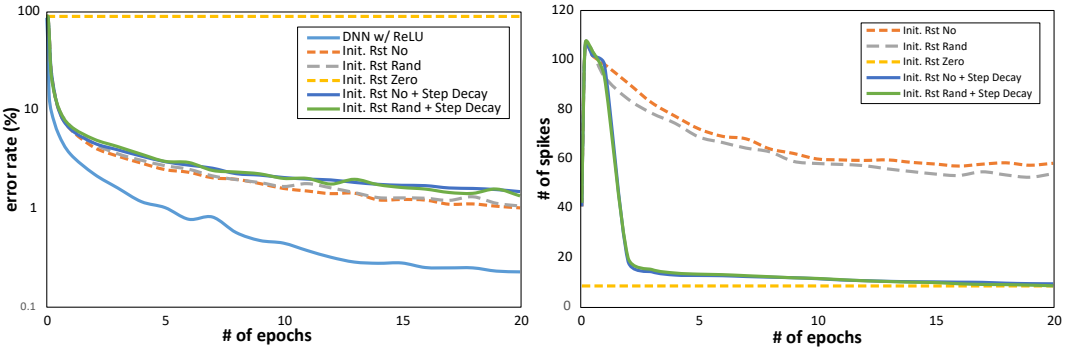
To put our proposal in a proper context, we show the SNN training results (error rates and the number of spikes versus training epochs) obtained by using each of the three aforementioned precharging methods, as shown in Figure 2. In the figure, the curve labeled "DNN w/ReLU" represents the error rate of the conventional DNN with ReLU activations, and three types of dotted lines indicate the measurements obtained from using the three precharging approaches (note that two types of solid lines represent the results from using our proposed approach, which will be elaborated shortly).

Initially in the training, we noticed a similar pattern of the effects of precharged membrane potential on training quality as in the DSN paper [22]. That is, the result using the "no reset" option gave the best result as shown in the figure. However, as the training continued, the precharged membrane potential started to become acting as noise, which hindered further training.

Algorithm 2: Training in Deep Spiking Networks with Proposed Step-Decay Method

E : # of epochs, D : # of training data, T : Time period
phi_{bwd} : Membrane potentials on the backward path, C_d : Linear decay constant

for *e in E* **do**
  for *d in D* **do**
    $phi_{bwd} \leftarrow$ INITIALIZATIONFUNCTION$/(C_d \times e)$; `// step-wise linear decay`
    $del \leftarrow 0$;
    for *t in T* **do**
      $s_t \leftarrow$ SAMPLINGINPUTDATA(d);
      $err_t \leftarrow$ FORWARDPATHOPERATION(s_t);
      $del_t \leftarrow$ BACKWARDPATHOPERATION(err_t);
      $del \leftarrow del + del_t$;
    end
    UPDATEPARAMETERS(del);
  end
end



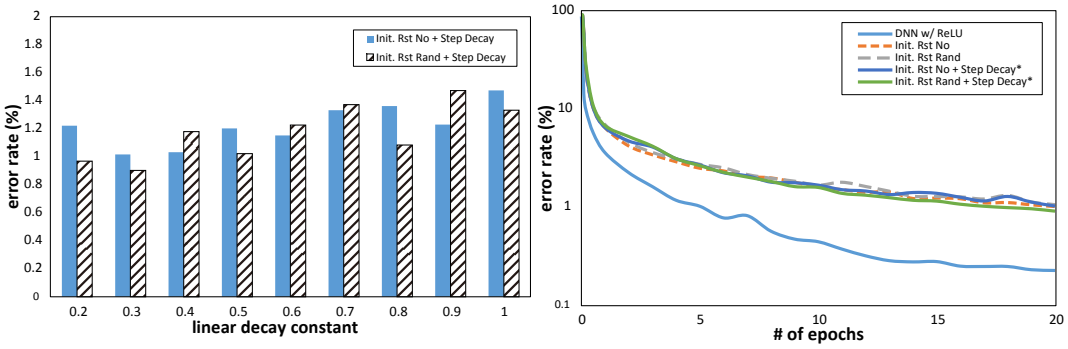
(a) (b)

Figure 3: (a) Training error (%) and (b) the number of the backward spikes during the training of a DSN using F-SGD with the MNIST (decay constant $C_d = 1$ for the proposed step-decay method).

To address this issue, we propose a new step-wise decay method for precharging the membrane potentials. Our method can boost the effectiveness of SNN training and reduce the negative effect of precharging when an SNN completes its training. Algorithm 2 outlines our approach, in which the amount of each precharged membrane potential decreases as training epochs proceed, effectively reducing its noisy effect on the final result. As proof of concept, in this paper, we use a linear function for implementing a step-wise decay (with a linear decay constant C_d); other types of (linear and non-linear) functions would work as well.

The solid lines in Figure 2a represent the measurement results obtained by applying the proposed decay approach to SGD-based training of the same DSN shown previously ($C_d = 1$). In this experiment, we could get better training results by adopting the proposed approach. For both of the "no reset" and "uniform random reset" options, our approach enhanced the training results. Compared with the original DSN, we were also able to reduce the number of backward spikes, owing to the decreased membrane potentials.

As shown in Figure 3a, we further tested our approach with F-SGD, an SNN training algorithm proposed in the context DSNs, which is different from SGD in that it updates the parameters whenever a spike is generated. In contrast to the SGD case, the improvement delivered by applying our method to F-SGD was marginal. This is because the number of spikes on the backward path for F-SGD was significantly smaller than that for SGD, as seen in Figure 3b. Based on this observation, we deduced that F-SGD could be more sensitive to the level of precharged membrane potentials and that we could get better training results if we reduce the effect of the decay at each epoch.



(a) (b)

Figure 4: (a) Training error (%) of a DSN trained by F-SGD using the MNIST data with various values of the linear decay constant C_d measured after 20 epochs (note: the case of $C_d = 0.1$ is omitted because of the unsatisfactory performance obtained with this value). (b) Training error (%) of the same DSN trained by F-SGD when $C_d = 0.3$, which is empirically the best value in our experiments (data: MNIST).

Table 1: Training Results on Various Configurations with the MNIST Dataset after 20 Epochs

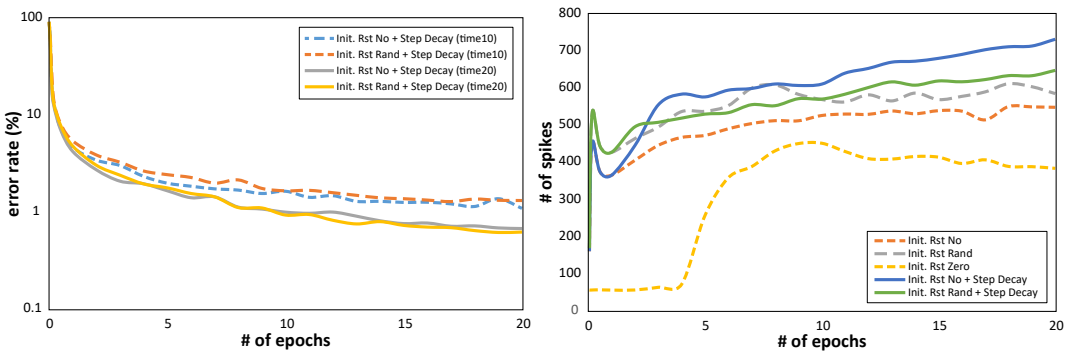
Training configurations of DSN				**Training error** (%)
Time period	Optimizer	Initialization[a]	Step decay[b]	
5	SGD	zero reset	n/a	90.136
10	SGD	zero reset	n/a	4.384
20	SGD	zero reset	n/a	1.002
10	SGD	no reset	n/a	1.530
10	SGD	uniform random reset	n/a	1.898
10	F-SGD	zero reset	n/a	90.126
10	F-SGD	no reset	n/a	1.006
10	F-SGD	uniform random reset	n/a	1.050
10	SGD	no reset	1.0	1.088
10	SGD	uniform random reset	1.0	1.310
10	F-SGD	no reset	1.0	1.474
10	F-SGD	uniform random reset	1.0	1.332
10	F-SGD	no reset	0.3	1.016
10	F-SGD	uniform random reset	0.3	0.902

[a] The initialization function of the membrane potentials on the backward path [22]
[b] The linear decay constant (see Section 2.2)

Finally, to find an (empirically optimal) linear decay constant that can reduce the precharged potential effectively, we tried to sweep the values of C_d from 0.1 to 0.9 (in the unit of 0.1), as shown in Figure 4a. As expected, the training accuracy varied by changing the linear decay constant, and using $C_d = 0.3$ gave the best result in this specific experiment. Figure 4b shows the error rates versus the number of training epochs obtained by applying our approach with this best value of $C_d = 0.3$ to F-SGD training of the same DSN used in the previous experiments. Table 1 lists additional experimental results training obtained by using various configurations.

3 Discussion and Conclusion

In this paper, we have proposed a new approach to boosting the effectiveness of DSN training with SGD and backpropagation. Our proposal is based on the fact that the initialization of membrane



(a) (b)

Figure 5: (a) Training error (%) of a DSN trained by using SGD with the MNIST data for the time periods of 10 and 20. (b) The number of forward spikes during the training of the same DSN (optimized by SGD using the MNIST data).

potentials on the backward path of a DSN significantly affects the training accuracy as previously discovered. The main idea of the proposed method is that decreasing the effect of initialization potentials can improve training accuracy as training progresses. We validated the effectiveness of our method through a diverse sets of experiments under various conditions.

As long as we train an SNN with SGD and backpropagation (both are widely used in DNNs), the precharge method is expected to be an inevitable prerequisite for boosting the effectiveness of SNN training. In this context, we believe that our approach makes a meaningful contribution to the field, providing a simple and effective means of improving SNN training quality. This paper presents our experimental results only with one type of SNN (*i.e.*, DSN) and one type of data (*i.e.*, MNIST). We expect that our method will be able to find applications in other types of SNNs and datasets, which will be a part of our future work.

It is possible to interpret the precharge effect as a type of transfer learning, especially when we do not reset the membrane potential at the beginning of each training. In this case, the maintained membrane potential acts to compensate the lack of spiking caused by insufficient time periods. However, depending on the training data and/or the degree of network training, this could merely indicate noise, even though the maintained membrane potential led to satisfactory results in our work and in the original DSN paper.

To scrutinize this interpretation further, we hypothesized that the randomized membrane potential with a step-wise decay could be the best combination among many initialization options. We then tried to empirically verify our hypothesis by carrying out experiments with sufficient time periods, as shown in Figure 5a. We also found out that the firing rate of the forward path on the network increased further when we applied the step-wise decay method as depicted in Figure 5b.

From a neuroscience point of view, these phenomena correspond to the long-term potentiation where a persistent strengthening of synapses occurs between neurons [4], resulting in higher spikes. In addition, if we interpret the initialized values as a trace of background activities of the network, they interfere with memory maintenance in synapses by leading a synaptic efficacy to faster decay [9], inhibiting the long-term potentiation, *i.e.*, successful learning.

Acknowledgments

This work was supported in by the BK21 Plus Project in 2016 (Electrical and Computer Engineering, Seoul National University), in part by a grant from Samsung Electronics, and in part by a grant from Naver.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv.org*, page arXiv:1409.0473, Sept. 2014.

[2] D. Chicco, P. J. Sadowski, and P. Baldi. Deep autoencoder neural networks for gene ontology annotation predictions. *BCB*, pages 533–540, 2014.

[3] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv.org*, page arXiv:1406.1078, June 2014.

[4] S. Cooke and T. Bliss. Plasticity in the human central nervous system. *Brain*, 129(7):1659–1673, 2006.

[5] 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 *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[6] N. Frémaux, H. Sprekeler, and W. Gerstner. Functional requirements for reward-modulated spike-timing-dependent plasticity. *The Journal of Neuroscience*, 30(40):13326–13337, 2010.

[7] W. Gerstner and W. M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[8] R. Gütig and H. Sompolinsky. The tempotron: a neuron that learns spike timing–based decisions. *Nature neuroscience*, 9(3):420–428, 2006.

[9] D. Higgins, M. Graupner, and N. Brunel. Memory maintenance in synapses with calcium-based plasticity in the presence of background activity. *PLOS Comput Biol*, 10(10):e1003834, 2014.

[10] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[11] E. M. Izhikevich. Polychronization: computation with spikes. *Neural computation*, 18(2): 245–282, 2006.

[12] W. M. Kistler and J. L. Van Hemmen. Modeling synaptic plasticity in conjunction with the timing of pre-and postsynaptic action potentials. *Neural Computation*, 12(2):385–405, 2000.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, 2012.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the Ieee*, 86(11):2278–2324, Nov. 1998.

[15] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.

[16] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[17] B. Lee, J. Baek, S. Park, and S. Yoon. deeptarget: End-to-end learning framework for microrna target prediction using deep recurrent neural networks. *BCB*, pages 434–442, 2016.

[18] J. Lee, T. Delbruck, and M. Pfeiffer. Training Deep Spiking Neural Networks using Backpropagation. *arXiv.org*, page arXiv:1608.08782, Aug. 2016.

[19] W. Maass. Networks of spiking neurons - The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[20] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[21] S. Min, B. Lee, and S. Yoon. Deep learning in bioinformatics. *Briefings in Bioinformatics*, 2016.

[22] P. O'Connor and M. Welling. Deep Spiking Networks. *CoRR*, 2016.

[23] 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*, 7, 2013.

[24] S. Park, S. Min, H. Choi, and S. Yoon. deepmirgene: Deep neural network based precursor microrna prediction. *arXiv preprint arXiv:1605.00017*, 2016.

[25] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. *Advances in neural information processing systems*, pages 3104–3112, 2014.

[26] D. Zambrano and S. M. Bohte. Fast and Efficient Asynchronous Neural Computation with Adapting Spiking Neural Networks. *arXiv.org*, Sept. 2016.