Progressive Compression and Weight Reinforcement for Spiking Neural Networks

Hammouda Elbez, Kamel Benhaoua, Philippe Devienne, Pierre Boulet

Progressive Compression and Weight Reinforcement for Spiking Neural Networks

Hammouda Elbez, Kamel Benhaoua, Philippe Devienne, *Member, IEEE,* and Pierre Boulet, *Senior Member, IEEE*

I. Abstract

Neuromorphic architectures are one of the most promising architectures to significantly reduce the energy consumption of tomorrow's computers. These architectures are inspired by the behaviour of the brain at a fairly precise level and consist of artificial Spiking Neural Networks (SNNs). To optimise the implementation of these architectures, we propose in this paper a novel progressive network compression and reinforcement technique based on two functions, progressive pruning and dynamic synaptic weight reinforcement used after each training batch. The proposed approach delivers a highly compressed network (up to 75 % of compression rate) while preserving the network performance when tested with MNIST.

***Index Terms*—Spiking Neural Networks, neuromorphic computing, unsupervised learning, pruning, compression, dynamic threshold.**

II. Introduction

The human brain is considered as the most powerful biological computing machine, with the ability to do parallel processing for a long time with a relatively low energy consumption compared to the artificial computing machines and existing technologies. Composed of around 20 billion neurons and 200 trillion synapses, the human brain is able to process a huge amount of data by exchanging spikes between neurons and consuming only about 20 W of power. Recently, we have seen a rising interest in bio-inspired architectures such as neuromorphic architectures using Spiking Neural Networks (SNNs). These biologically plausible architectures imitate the brain for the purpose of exploring the advantages that the brain offers, especially low energy consumption.

Several works were presented in the past to propose a hardware implementation of such neuromorphic architectures, using different hardware components [1], [2], but the common observed characteristic of the presented SNNs is the huge size of the networks, in order to get better performance than non-Spiking Artificial Neural Networks (ANNs). Meaning having a large amount of neurons and synapses, and a rising complexity when it comes to the hardware implementation and analysis of such architecture, in [3] the authors had to increase the number of neurons in the 2-layer SNNs by more than 60 times, in order to improve the performance by 12 %, in [4] the authors used 2 hidden layers of 800 neurons each in order to get an average accuracy of 98.6 % and in [5] the authors implemented a network with two hidden layers of 1200 neurons each for an average accuracy of 98.64 %. From the presented examples, we can see that larger networks are needed for better performance which introduces the following challenges :

- Larger networks are usually harder to analyse.
- Having larger networks means longer simulations with more computational resources to use and a loss of the energy efficiency in such networks.
- Larger networks are harder to implement in hardware using known technologies like the memristive crossbars [6], [1], which are subjected to many design challenges seen when a large network is deployed [7].

Communication in SNNs is made using spikes, which are transmitted by the synapses that represent the channel of communication between the neurons. The number of synapses in a network scales with the number of neurons and from the hardware point of view of SNNs, more synapses means a higher cost of production, and the efficient use of such elements leads to a more energy efficient network. Pruning is one of the techniques widely used in ANNs to reduce the network size and complexity. This technique is inspired from the observation that during the development of the human brain, synaptic connections are pruned from the early stages [8], such phenomena inspired researchers in the neural networks field to adapt this technique to get a smaller network while preserving the same performance or with a small loss [9], [10], using this technique in Deep Neural Networks (DNNs) with the state-of-the-art AlexNet results in a 30-times reduced number of synapses than the baseline while preserving the performance [11]. The idea behind these researches is not new and based on the principle that "the simplest solution is most likely the right one" in other words "Entities should not be multiplied without necessity". The idea is attributed to English Franciscan friar William of Ockham (c. 1287–1347), a scholastic philosopher and theologian who used a preference for simplicity. As the Ockham's razor [12] says that when presented with competing hypotheses that make the same predictions, one should select in a neural network the most relevant inputs and connections, that is, the as simple as possible network. Similarly, in science, for each accepted explanation of a phenomenon, there may be an extremely large, perhaps even incomprehensible, number of possible and more complex alternatives. Since one can always burden failing explanations with ad hoc hypotheses to prevent them from being falsified, simpler theories are preferable to more complex ones because they are more testable. Furthermore, in genetics the idea is even stronger to be able to understand why the human genome is, for instance, 1.5 smaller that the genome of peas. The idea behind that there are a lot of irrelevant or redundant coding on genomes.

So according to Kolmogorov's complexity [13], to compress is to understand. The final word is often attributed to Einstein, himself a master of the quotable one liner: "Everything should be made as simple as possible, but not simpler."

SNNs have the capability to handle natural signals and can be implemented physically through ultra-low power devices based, for instance, on CMOS [14] or memristors [2]. CMOS artificial neuron is a very simple component (only six transistors operating in the subthreshold mode) and its energy consumption is several orders of magnitude lower than artificial neurons reported in the literature, but also two to three orders of magnitude lower than the energy consumption of living neurons.

As for recent works about pruning in SNNs, in [15] the author combined pruning while the network is learning with a weight quantization technique using a 2-layer SNN of 6400 neurons, in order to reduce the network size and parameters, and shows that we can achieve highly compressed networks, while preserving a good performance with a fixed pruning threshold, which is defined from the beginning, deactivating the synapses that are considered non critical, based on the non supervised local Spiking Time Dependent Plasticity (STDP) rule [16], used to update the weights of synapses with every training batch, until the end of the learning phase, where remaining non critical synapses are removed from the network. Another work [17], based on the locally connected Spiking Neural Network (LC-SNN), while using a 2-layer SNN of 900 neurons, resulted in 50 % of removed synapses and 90 % maintained accuracy, the pruning operation being made only once at the end of the learning phase. In [18], a soft-pruning method was used during the training process, which prevents the unnecessary update of the network parameters. This was considered as different compared to the existing approaches that use pruning on an already trained network. This approach was able to maintain 90 % accuracy, on the MNIST dataset with a 75 % of synapses removed.

The novelty of our work, is the use of a dynamic threshold for pruning, instead of a fixed one like the previous works, using a progressive pruning function, that computes a new threshold, with every batch based on the last pruning operation performance. During every learning batch, the produced network is able to keep the same performance compared to the baseline with up to 75 % smaller network, also we introduce a synapse reinforcement for the critical synapses, using dynamic reinforcement function, which helps the network to preserve an average spiking frequency, close to the baseline, and push the neurons to specialize in one class. The combination of these two techniques, is able to produce trainable compressed networks, that can be trained again for a better accuracy.

The rest of this paper, is organized as follows: Section III contains background about the used network, neuron and synapse models, with a presentation of the Spiking Time Dependent Plasticity (STDP) rule. Section IV, presents the contribution in details. In Section V, the results are presented, discussed and compared to the baseline and the related works and finally Section VI contains the conclusion.

III. Background

A. Network Topology

In our experiments, the used topology is based on Dielh and Cook work [3] to classify the MNIST handwritten digits [19]. As described in Figure 1, it is a 2-layer Spiking Neural Network composed of the input layer, where each neuron represents a pixel, which is feed-forward fully connected to the next unsupervised layer, using excitatory synapses. This unsupervised layer is fully connected with itself for lateral inhibition, in order to create competition between the neurons, and to prevent one neuron from dominating all the inputs, which is also known as the Winner-Takes-All (WTA) rule. We also employ Homeostasis [20], which is an adaptive membrane threshold technique, it is used to guarantee that every excitatory neuron, learns a unique feature and avoid neuron dominating the inputs. This technique is achieved using equation 1.

$$V_{\text{threshold}} = V_t + \tau \tag{1}$$

Where V_t is a constant, represents the initial threshold defined in the network. τ is dynamically changed, if the neuron fires often, the value of τ will increase which requires more inputs to fire again, and decays if the activity of the neuron is less often. Which ensures a close average spikes activity, between all the neurons in the network.

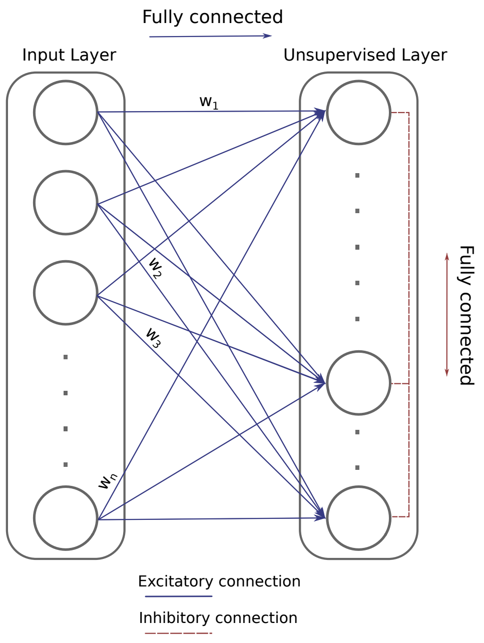


Fig. 1: SNN topology, with feed forward excitatory synapses from the input to the unsupervised layer, which are later pruned. Inhibitory synapses between neurons of the unsupervised layer.

In this work, we have used the N2S3 simulator [21] for all simulations. N2S3 is an open-source scalable spiking neuromorphic hardware simulator, written in Scala and based on the Akka actor system.

Those simulations are executed on an OpenStack cluster[1] node, with the following characteristics:

- RAM : 250 GB
- VCPU : 1 10-core Intel Xeon Silver 4114 CPU

B. Spiking Neuron Model

The Leaky-Integrated-and-Fire (LIF) neuron model [22] is used to simulate the neuron membrane potential. LIF is considered as one of the simplest models, that includes time in its operation, to update the membrane potential value of the neuron based on the input from other neurons in the network. The membrane potential keeps increasing until a specific threshold is reached, then the neuron sends a spike and enters a refractory period, before starting again to accumulate the received spikes. The membrane potential evolution is presented in Figure 2, and the LIF dynamic voltage equation is given by:

$$\begin{aligned} \tau_{\text{leak}} \frac{\partial v}{\partial t} &= [v(t) - v_{\text{rest}}] + r_m z(t), \\ v \leftarrow v_{\text{rest}} &\quad \text{when} \quad v \geq v_{\text{th}} \end{aligned} \tag{2}$$

Where v represents the membrane voltage, v_{rest} is the resting potential after a neuron fires a spike, τ_{leak} is the time constant of the leak with $\tau_{\text{leak}} = r_m c_m$, where r_m is the membrane resistance and c_m is the membrane capacitance.

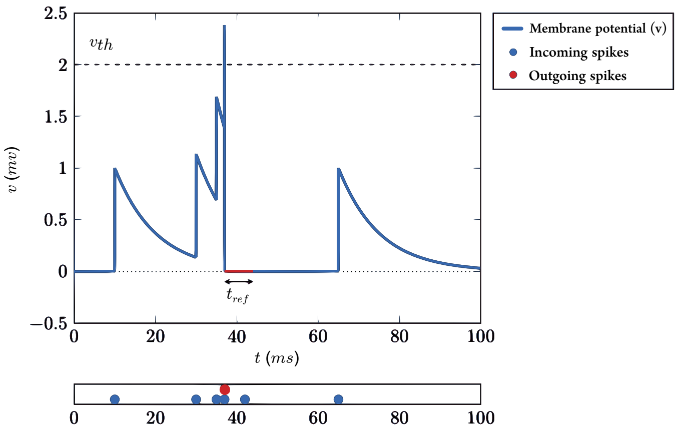


Fig. 2: Leaky-Integrated-and-Fire (LIF) neuron model membrane voltage activity

C. Learning Rule

Spiking Neural Networks usually use a local unsupervised learning rule (every synapse only has the information about its post- and pre-synaptic neurons), which is biologically plausible and suitable for hardware implementation. With SNNs, it is difficult to get high recognition rates, such as those of Deep Neural Networks (DNNs) or Convolutional Neural Networks (CNNs), that are based on global supervised learning rules (e.g., backpropagation). We use in our work the Simplified Spike Timing Dependent Plasticity (STDP) rule, which is easy to implement with nanodevices [23]. This simplified STDP rule is based on the t_{pre} pre-synaptic and t_{post} post-synaptic neuron firing times. When the pre-synaptic neuron fires before the post-synaptic one, the weight of the synapse is increased, and when the post-synaptic neuron fires before the pre-synaptic one, the weight of the synapse is decreased. This simplified STDP rule is modelled as:

$$\Delta_w = \begin{cases} +\eta_w e^{-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_{\text{STDP}}}} & \text{if } t_{\text{pre}} \leq t_{\text{post}} \\ -\eta_w e^{-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_{\text{STDP}}}} & \text{if } t_{\text{pre}} > t_{\text{post}} \end{cases} \tag{3}$$

τ_{STDP} represents the time constant that controls the leak, η_w is the learning rate, t_{pre} and t_{post} are times of spikes, for pre-synaptic and post-synaptic neuron respectively.

D. MNIST Dataset

We have used the MNIST handwritten dataset [19] for testing. This dataset is composed of 28x28 pixels images of handwritten digits with labels from 0 to 9. It contains 60,000 training images and 10,000 test images. For our experiments, training images were divided into 6 batches of 10,000 images in order to easily follow the network learning progress using the approach presented in this work, and to minimize training time. The images are presented to our network input layer and processed in the format of Poisson-distributed spike trains, the input layer neuron weights are randomly initiated based on a uniform distribution between 0 and 1. More parameters used in the network are presented in Table I.

TABLE I: Parameters used in the experiments

Input Stream			
MinFrequency	0Hz	MaxFrequency	22Hz
ExpositionDuration	350ms	PauseDuration	150ms
Neuron			
VoltageThreshold	35mv	RestingVoltage	0mv
STDP			
MaxWeight	1	MinWeight	0

IV. Contribution

Compressing Spiking Neural Networks (SNNs) results in a network with reduced parameters and complexity to analyze, which will make it easier to implement in hardware using the existing technologies [2], [1].

Pruning non-critical synapses is a widely used technique in ANNs and SNNs, and the recent works have used two different approaches to prune the network: at the end of the training, or while learning using a fixed threshold (which is defined from the beginning based on experiments). In [15], the author was able to achieve highly compressed networks by simply eliminating the non-critical synapses based on the STDP learning rule and a fixed threshold. We have observed from our tests using the MNIST dataset and based on the network model used in [3], that the neurons generally start to specialize in detecting a specific class early while learning, and the network performance starts to rise from the first

[1]http://hpc.univ-lille.fr/cloud-openstack

learning batch as demonstrated in Figure 3. This means that we can start removing synapses earlier, and the network can maintain the majority of its performance when removing a lot of synapses, based on STDP as shown in recent works [15], [17]. If we run another training epoch on the new reduced network, we will get a network with better performance, and an opportunity to prune based on a higher threshold, which should result in a more compressed network with an equal or better performance. Starting from this observation and based on the recent contributions, we propose *a progressive pruning function* to control a dynamic threshold, which is calculated and used to prune after a fixed training batch, and *a synaptic weight reinforcement mechanism*, used to maintain the average spiking frequency of the network, close to the baseline and to push the neurons that did not specialize in a certain class to do so, by reinforcement of the critical maintained synapses.

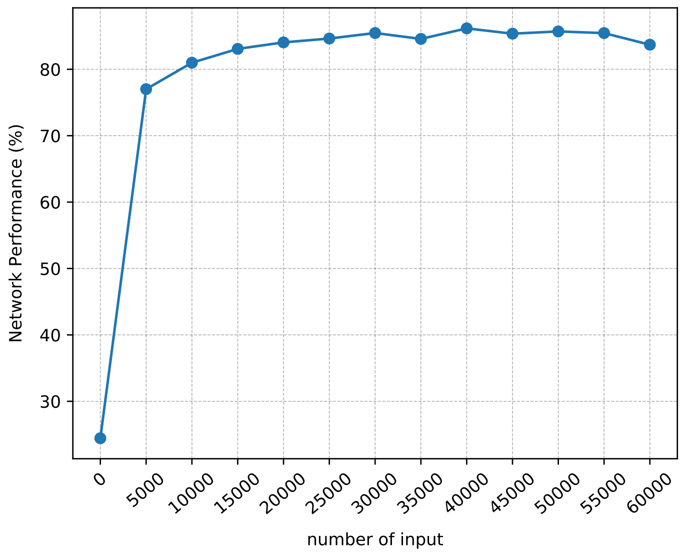


Fig. 3: Learning progress in SNN

A. Progressive Pruning

Progressive Pruning (PP) is based on reducing the number of synapses, between the input layer and the unsupervised layer. This reduction is performed after each batch while learning, using a dynamic progressively increasing pruning threshold $T_n, n \in \mathbb{N}$. This threshold is calculated using equation 4.

$$T_{n+1} = T_n + \alpha * (C_{r_n}/C_n) \qquad n \in \mathbb{N} \tag{4}$$

Where n represents the batch number, T_{n+1} is the threshold to use for the next batch, α represent a constant defined by the user, the choice of α is discussed in the next section ($\alpha = 0.05$ in this work), T_n is the old threshold used in the last batch ($T_0 = 0$), C_{r_n} represents the number of remaining synapses between the two layers at batch n and finally C_n represents the total number of synapses before applying pruning to this batch (which equals to the remaining synapses from the last batch, $C_n = C_{r_{n-1}}$).

Using equation 4, the threshold is defined based on the previous pruning performance (if the pruning rate using T_n from the last batch was large, $T_{n+1} - T_n$ will be small and vice versa), which prevents the network from a major performance loss, while reducing the network every time by eliminating the non-critical synapses.

B. Dynamic Synaptic Weight Reinforcement

After reducing the network size, the neuron will learn to react only to a specific pattern or class (in our case a digit). We observed that during the learning process, some neurons fail to learn a specific class, which affects the network performance resulting in false classifications. On the other hand, applying Progressive Pruning on the network results in a decrease of the average spiking network frequency, which will affect the energy consumption of our network positively, but may cause a frequency loss in multilayer networks, which is a known issue in Convolutional Spiking Neural Networks (CSNN) as described in [24]. Based on those observations, we propose a Dynamic Synaptic Weight Reinforcement (DSWR), which concerns the synapses that are conserved and considered as critical, to improve the network performance, by pushing the neurons that did not specialize in a specific pattern or class to do so, and keep the average spiking frequency of the network near the baseline. This DSWR procedure is executed after the pruning in each iteration, and it's done based on equation 5.

$$W_{n+1} = W_n + \beta * T_n, \qquad n \in \mathbb{N}. \tag{5}$$

Where n represents the batch number, W_{n+1} is the new weight of the concerned synapse, W_n represents the actual weight of that synapse, β is a defined constant, chosen based on experiments and discussed in the next section (in this work β= 0.1) and T_n represents the actual used threshold for pruning from equation 4. This equation means that when the compression ratio is large, the reinforcement will be too and vice versa.

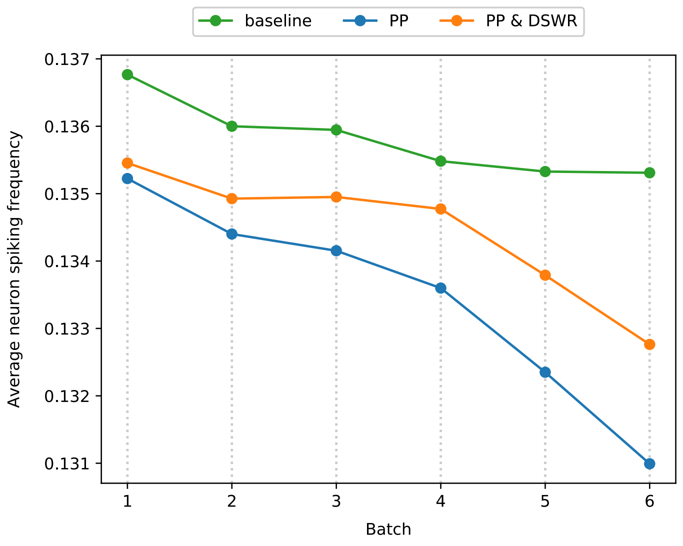


Fig. 4: Average spike frequency

Using the presented functions, we make sure to determine how much to increase the weight, and maintain the average spike frequency based on the network state. In Figure 4, using a network composed of 100 neurons, we can see the average spiking frequency of our network baseline and the proposed functions after each training batch.

A detailed flowchart concerning the proposed approach is presented in Figure 5.

C. α and β Selection

Choosing the values of parameters α and β is a multiobjective optimization problem, the two objectives being the network performance and the compression ratio. We did multiple experiments using different values, and compared them in terms of these two objectives, getting a Pareto front of non-dominated values. We finally chose the one that gave use the best performance with a compression ratio above 76 %.

In Figure 6, by trying different combinations of α and β, we can get a high compression rate but in the same time a higher loss in network accuracy, the selection of those parameters may vary from a case to another depending on the context of the application.

In Figure 7, we show the 10 possibilities from the 25, respecting the following conditions and draw the Pareto front of non-dominated solutions:

- The performance should be better than the baseline presented in [3] (82.90 %).
- The compression rate should be better than 76 %.

Based on these results, we decided to choose $\alpha = 0.05$ and $\beta = 0.1$, which gave us an accuracy of 85.10 % with a compression rate of 79.42 %, using a network composed of 100 neurons.

V. Experimental Validation & Discussion

In this part, we describe and discuss our experimental validation. The main tests are conducted using a network composed of 100 neurons, in the unsupervised layer connected to the input layer by 78400 excitatory synapses. The simulation on each case is executed 10 times using one epoch (of 60000 samples), and we report the average of these 10 simulations. Besides, tests were also conducted on 400, 900 and 1600 neuron networks to evaluate the performance of the presented approach on different sizes of networks.

A. Accuracy & Compression

We compare our results to the baseline and recent works on pruning in SNN in Table II. We can see that our methodology helps to compress the network and to get a higher accuracy in different network sizes, due to the progressive and iterative pruning and reinforcement operations. For 100 neurons, the work presented by Rathi et al. [15] could not preserve the same accuracy compared to baseline, and one of the possible reasons are the fact that there is no training after pruning, the one time pruning approach and the weight quantization makes the network lose information. No training after pruning and one epoch leads to an accuracy of 79.50 %, which is less than the baseline with a 75 % compression rate. We can see that with a 100 neurons network, using only Progressive Pruning (PP) gives better performance, compared to the baseline with more than 78 % of the synaptic connections removed, and when using both Progressive Pruning (PP) and Dynamic Synaptic Weight Reinforcement (DSWR), we get a slight improvement in accuracy and compression rate. Having a compressed network with a better performance than the baseline, or the work presented in [3] is possible, using the proposed functions and training after each batch, which helps the network to learn better and faster, while eliminating many synapses that are considered as noise to the network.

In the work presented by Diehl and Cook [3], tests were also conducted on a network composed of 400 neurons using 3 training epochs, the accuracy was 87 %, which is also close to our 400 neurons network baseline, without the presented approach, we can see from table II that we are able to compress the network up to 79.52 %, with 87.84 % accuracy using PP & DSWR with only one epoch.

From the observed evolution of the network, using a neural network of 100 neurons with PP and DSWR, we can see in Figure 8, that the network is compressed for more than half his initial synaptic connections in the first batch, the compression rate is increasing after each batch while the accuracy does the same thing but slowly.

B. Presented Approach On Larger Networks

While the selected parameters (α and β) give good results with small and medium networks, as shown in Table II, using the presented approach with larger networks, while preserving the same parameters and epoch number (only one), may have different impact on the network. In this part, we will present the test results we had, using a network of 1600 neurons similar to Diehl and Cook work [3], then compare it to the earlier results.

From Figure 9, compared to the previous experiments, we notice that we have a loss in performance (85.31 %), the network needs up to three batches to get a performance above 80 %, compared to smaller networks, where one batch is enough. For the compression rate, we can see also that the network this time is less compressed, compared to smaller ones (65.02 %), or to the results of Diehl and Cook [3], and starts from a small compression rate (4 %) in the first batch, which is not the case compared to earlier examples with smaller networks, which compress almost the half of the network in the first batch.

To justify and explain such a result when using the presented approach with a larger network, we propose two hypotheses:

1) The used constants (especially α) initial value may be too large for such a large network, and need to be adjusted properly.
2) The network with larger number of neurons needs more time to learn before pruning, which means the number of batches per epoch needs to be reviewed.

In order to get an idea on what is happening in the network, we can check in Figure 10, the heatmap of some neurons, captured after the first batch (10000 inputs) of training, as we can see the network had not enough time to start learning features, which is due to the competition between neurons to learn features, which takes more time compared to previous smaller networks. From the compression rate progress in Figure 9,

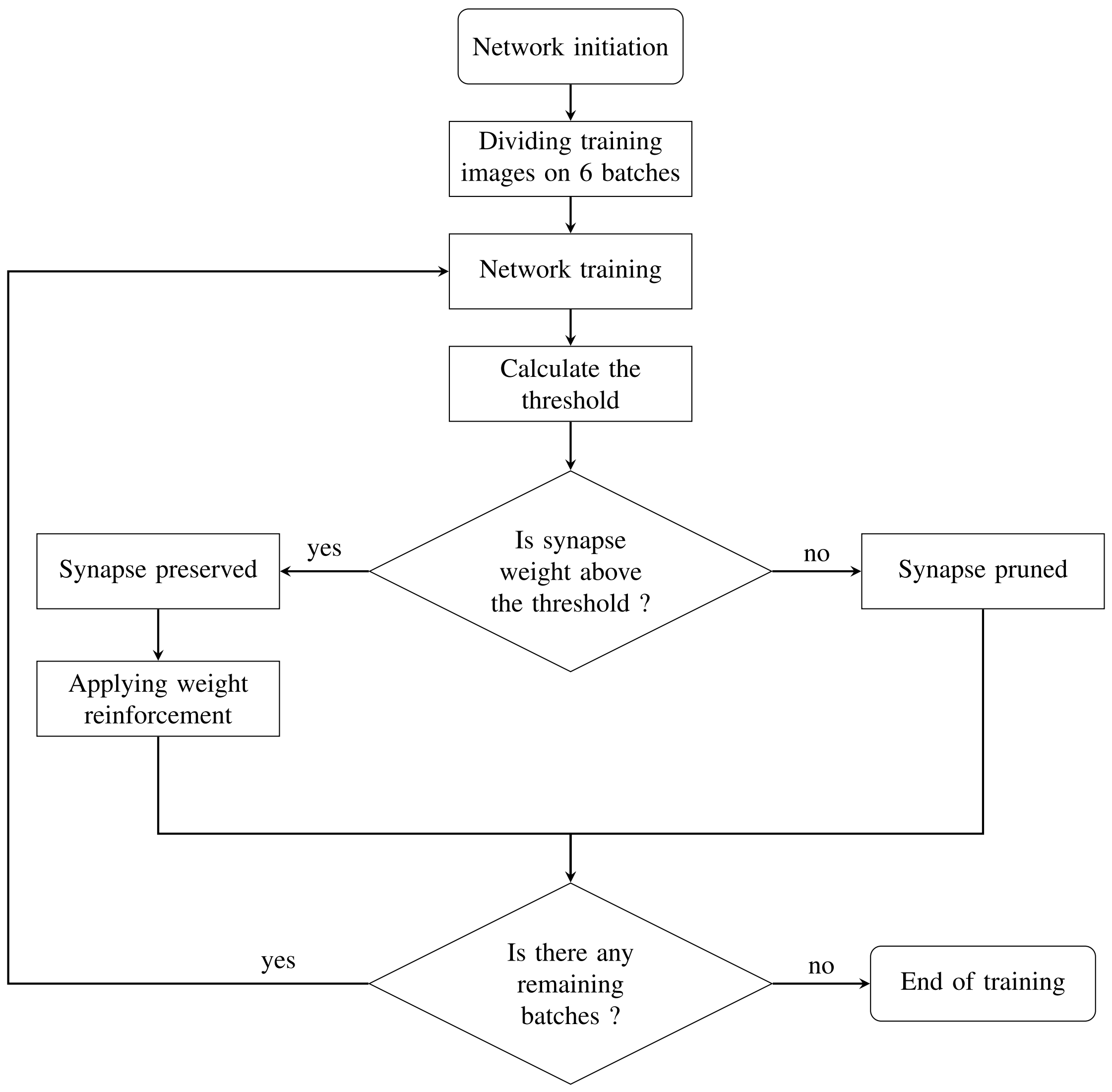

Network initiation
Dividing training images on 6 batches
Network training
Calculate the threshold
Is synapse weight above the threshold ?
yes
Synapse preserved
no
Synapse pruned
Applying weight reinforcement
Is there any remaining batches ?
yes
no
End of training

Fig. 5: Flowchart of the presented approach

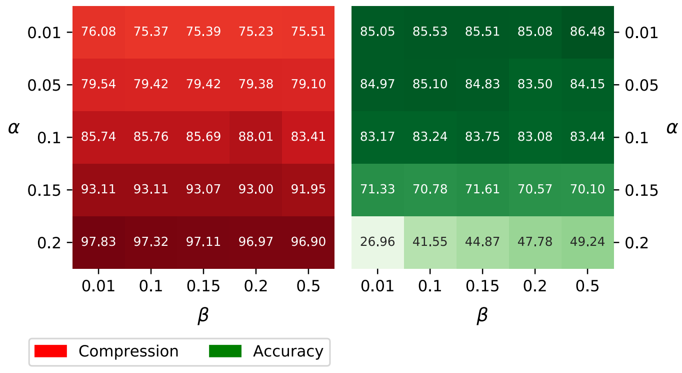


Fig. 6: α and β selection

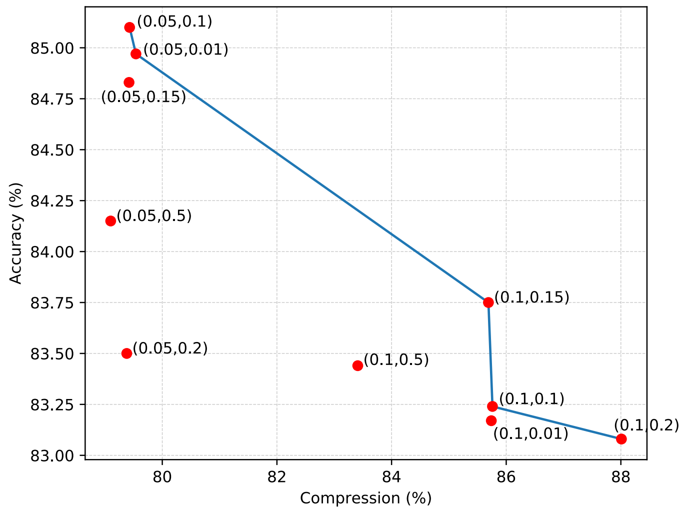


Fig. 7: Accuracy and compression in function of α and β (line = Pareto front).

we can discard the first hypothesis, since the problem is not related to pruning using high threshold, based on the low compression rate we had (4 %), which did not affect learning process of the network, but it is indeed related to the second hypothesis, about the time required for the network to start

Number of neurons	Paper	Learning rule	Pruning approach	Train after pruning ?	Number of epochs	Accuracy ± std	Compression ± std
100	Rathi et al. (2019)	Exponential STDP	Fixed threshold	No	1	79.50	75.00
	Diehl and Cook (2015)	Exponential STDP	–	–	1	82.90	–
	This work (baseline)	Simplified STDP	–	–	1	84.47 ± 1.55	–
	This work	Simplified STDP	PP	Yes	1	84.98 ± 0.37	78.71 ± 1.96
	This work	**Simplified STDP**	**PP & DSWR**	**Yes**	**1**	**85.10 ± 0.83**	**79.42 ± 0.06**
400	Diehl and Cook (2015)	Exponential STDP	–	–	3	87.00	–
	This work	Simplified STDP	–	–	1	87.77 ± 0.57	–
	This work	**Simplified STDP**	**PP & DSWR**	**Yes**	**1**	**87.84 ± 0.40**	**79.52 ± 0.03**
900	**This work**	**Simplified STDP**	**PP & DSWR**	**Yes**	**1**	**89.23 ± 0.91**	**77.28 ± 0.69**
1600	**This work**	**Simplified STDP**	**PP & DSWR**	**Yes**	**1**	**85.31 ± 0.24**	**65.02 ± 0.04**
	Diehl and Cook (2015)	Exponential STDP	–	–	7	91.90	–

TABLE II: Accuracy and compression test results compared to other similar works

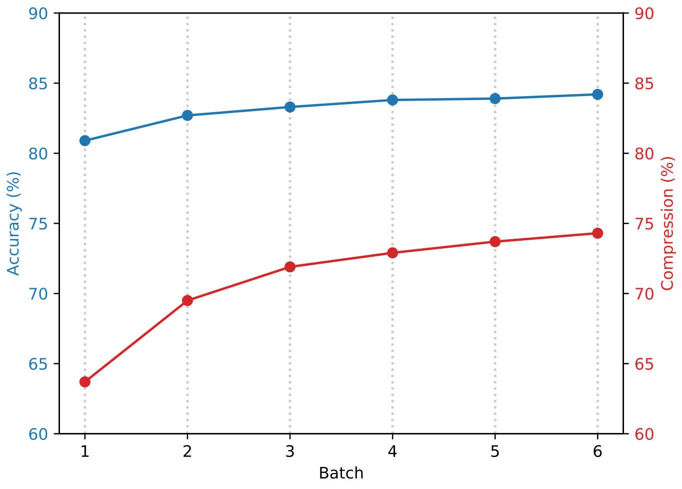


Fig. 8: Accuracy and Compression evaluation using PP and DSWR (100 neurons)

learning features, which justifies the small compression rate compared to smaller networks and the low performance.

In this case, when dealing with larger networks and based on our test results, it is better to reduce the number of batches per epoch, delay the prune operation and use more than one epoch for training, when applying the approach presented in this work, in order to give the network the required time to learn for a better performance and compression rate. Due to the fact that running this simulation with a large network takes a lot of time (days for 100 to 900 neurons and almost one month for 1600 neurons) and a significant amount of energy, we chose not run our tests for more than one epoch and present the results of the proposed modifications when working with large networks.

C. Trainable Compressed Network

Reducing the number of synapses in a network is helpful, for time and energy reduction, especially when implementing in hardware. One of the features that is commonly tested with reduced networks, is the possibility to have trainable networks after compression, because one of the reasons, that huge networks are used for training is to maximise the possibility to get good performance, which is not always possible if you start with small networks. In [25], [26], a lottery tickets hypothesis is presented for CNNs, saying that a randomly initiated dense neural network contains a subnetwork that, if isolated and trained alone, can match the accuracy of the original network. This subnetwork can be extracted using

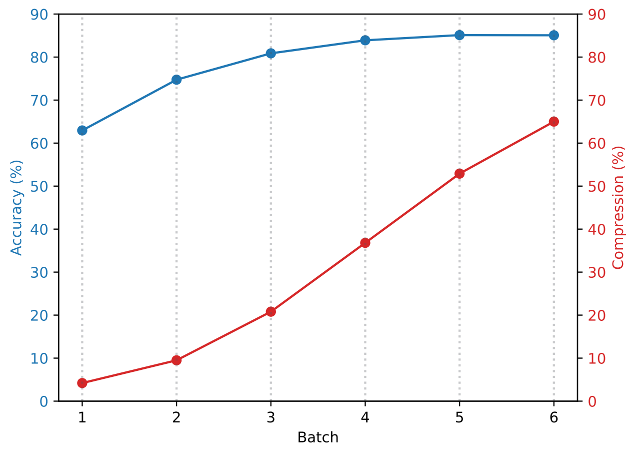


Fig. 9: Accuracy and Compression evaluation using PP and DSWR (1600 neurons)

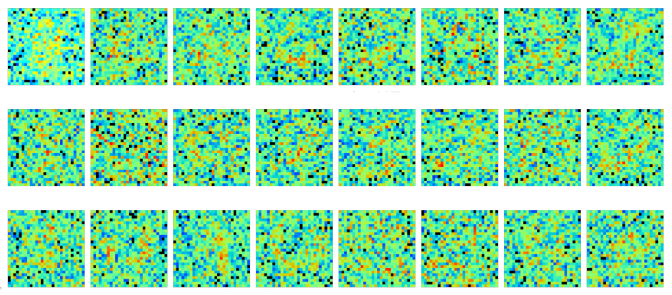
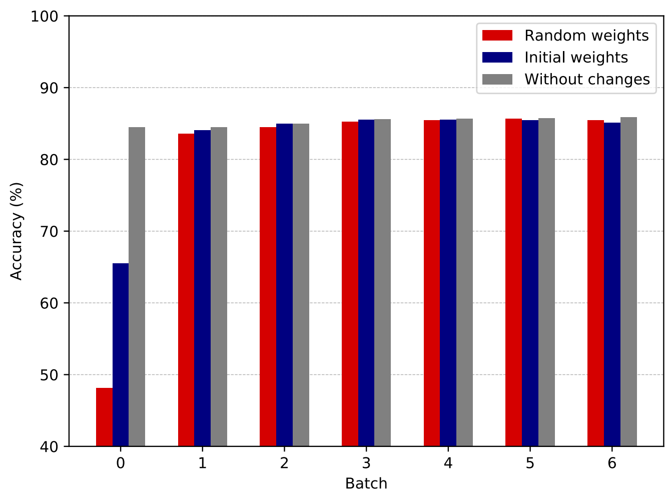
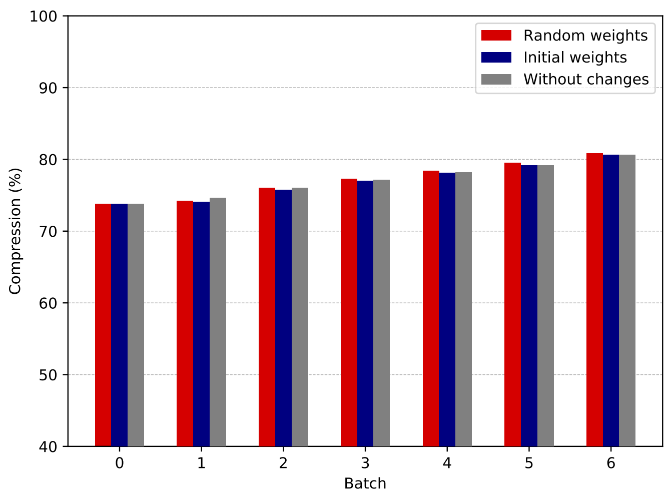


Fig. 10: Neurons Heatmap after one batch (black points represents pruned synapses)



(a) Accuracy



(b) Compression

Fig. 11: Accuracy (a) and Compression rate (b) during training batches

masks. It is hard to train a pruned network from the beginning in CNNs, which gives a worst accuracy than the original model, and is considered as not trainable. The solution was to train the pruned network with the initial synapse weights, and not randomly initiated in order for the network to learn well, and for gradient descent to be able to find a good solution. Such hypothesis was not yet tested in Spiking Neural Networks (SNNs). In Figure 11, we test using a 100 neuron network if the compressed network is trainable, using the new compressed network either without changing synapse weights, using random initial weights or using the initial weights of the original network. We can see after one batch that the network with initial synaptic weights without training reaches more than 60 % accuracy compared to the network with randomly initiated weights, that barely reaches 50 %, this observation was also presented in [25] for CNNs. At the end of the 6 batches of training, we get an accuracy near 85 % for the three use cases. For the compression rate, we can see that the network gets more compressed in the three use cases to reach a compression rate of 80 % while preserving a better accuracy compared to baseline.

This proves that many observations presented in [25] are true in SNNs too, and the trainable network gets a higher compression rate, when trained again irrespectively of the initialisation strategy, of the synaptic weights that only influences the speed of learning, not its quality.

D. Pruning Batch

During our tests, we used a single MNIST dataset epoch (60000 input samples), divided into 6 batches (of 10000 samples each), while applying the presented approach after each batch. We study here the influence of the size and number of batches, on the accuracy and compression of the network. In Figure 12, we compare one batch of 60000 samples, 3 batches of 20000 samples and 6 batches of 10000 samples, using a neural network composed of 100 neurons.

We can see that the accuracy of the network did not decrease, and the variation of the number of batches does not

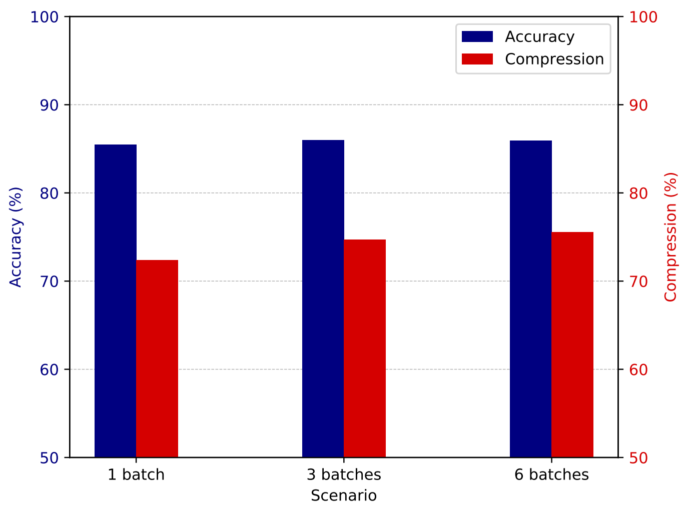


Fig. 12: Pruning batch variation

have a clear impact on the accuracy. However, we can see that the compression ratio increases, when the number of batches is increased, which is due to the presented approach, with the dynamic threshold used and the training of each compressed network. We conjecture that having more than 6 batches will have an effect similar to training the network again, for another epoch, but may results in a larger training time.

VI. Conclusion

Many researches focus on the unreasonable effectiveness of Neural Networks and our deep inability to understand causality, correlation in neural networks. To prune a neural network can be seen as a normal discriminant analysis, for dimensionality reduction before better classification and explanation.

In this work, we have presented a novel approach to reduce the size of Spiking Neural Networks, while preserving a similar or better accuracy, by the use of the Progressive Pruning, with a dynamic threshold and a Dynamic Synaptic

Weight Reinforcement, compared to the existing approaches of using a fixed threshold while pruning, or pruning at the end of training without training again. Our approach is able to get a better accuracy compared to the non-compressed networks, when applied to small or medium networks, using only one epoch. But when dealing with larger networks, more than one training epoch is needed with larger batches to provide sufficient time for the network to learn before applying the presented approach. The compressed networks were proved trainable to reach even better accuracy and a better compression rate, using the initial synapses weights of the original network, based on the presented idea in [25] for Convolutional Neural Networks. This also suggests that the Ticket Lottery Hypothesis presented in the same work may also be true for Spiking Neural Networks, to create masks for network pruning before training, which we will test and validate in future work. More datasets will be studied like colour images (CIFAR-10 [27]) or EMNIST [28]. Finally, it's worth mentioning that training compressed networks will result also in a reduced training time, which is very important especially for large networks.

Acknowledgments

This work was supported in part by IRCICA (Univ. Lille, CNRS, USR 3380 – IRCICA, F-59000 Lille, France) under the Bioinspired Project, PHC-CURIEN TASSILI 19MDU213 grant and PRIMA WATERMED 4.0 project.

References

[1] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, Sep. 2011, pp. 1–4, iSSN: 0886-5930.

[2] M. Shahsavari and P. Boulet, "Memristor nanodevice for unconventional computing:review and applications," *arXiv:1703.00331 [cs]*, Mar. 2017, arXiv: 1703.00331. [Online]. Available: http://arxiv.org/abs/1703.00331

[3] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, 2015. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2015.00099/full

[4] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training Deep Spiking Neural Networks Using Backpropagation," *Frontiers in Neuroscience*, vol. 10, 2016. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2016.00508/full

[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)*, Jul. 2015, pp. 1–8, iSSN: 2161-4393.

[6] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008. [Online]. Available: https://www.nature.com/articles/nature06932

[7] B. Liu, W. Wen, Y. Chen, X. Li, C.-R. Wu, and T.-Y. Ho, "EDA Challenges for Memristor-Crossbar based Neuromorphic Computing," in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, ser. GLSVLSI '15. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, May 2015, pp. 185–188. [Online]. Available: https://doi.org/10.1145/2742060.2743754

[8] H. P. R., "Synaptic density in human frontal cortex — developmental changes and effects of aging," *Brain Research*, vol. 163, no. 2, p. 195–205, 1979.

[9] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jun. 1990, pp. 598–605.

[10] B. Hassibi, D. G. Stork, and G. Wolff, "Optimal Brain Surgeon: Extensions and performance comparisons," in *Advances in Neural Information Processing Systems 6*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. Morgan-Kaufmann, 1994, pp. 263–270. [Online]. Available: http://papers.nips.cc/paper/749-optimal-brain-surgeon-extensions-and-performance-comparisons.pdf

[11] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding," *ICLR*, 2015.

[12] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's razor," Apr. 1987. [Online]. Available: https://doi.org/10.1016/0020-0190(87)90114-1

[13] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 3rd ed., ser. Texts in Computer Science. New York: Springer-Verlag, 2008. [Online]. Available: https://www.springer.com/gp/book/9781489984456

[14] I. Sourikopoulos, S. Hedayat, C. Loyez, F. Danneville, V. Hoel, E. Mercier, and A. Cappy, "A 4-fJ/Spike Artificial Neuron in 65 nm CMOS Technology," *Frontiers in Neuroscience*, vol. 11, 2017. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2017.00123/full

[15] N. Rathi, P. Panda, and K. Roy, "STDP-Based Pruning of Connections and Weight Quantization in Spiking Neural Networks for Energy-Efficient Recognition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 4, pp. 668–677, Apr. 2019.

[16] G.-q. Bi and M.-m. Poo, "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type," *Journal of Neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, Dec. 1998. [Online]. Available: https://www.jneurosci.org/content/18/24/10464

[17] D. J. Saunders, D. Patel, H. Hazan, H. T. Siegelmann, and R. Kozma, "Locally connected spiking neural networks for unsupervised feature learning," *Neural Networks*, vol. 119, pp. 332–340, Nov. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608019302333

[18] Y. Shi, L. Nguyen, S. Oh, X. Liu, and D. Kuzum, "A Soft-Pruning Method Applied During Training of Spiking Neural Networks for In-memory Computing Applications," *Frontiers in Neuroscience*, vol. 13, 2019. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2019.00405/full

[19] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[20] E. Marder and J.-M. Goaillard, "Variability, compensation and homeostasis in neuron and network function," *Nature Reviews Neuroscience*, vol. 7, no. 7, pp. 563–574, Jul. 2006. [Online]. Available: https://www.nature.com/articles/nrn1949

[21] P. Boulet, P. Devienne, P. Falez, G. Polito, M. Shahsavari, and P. Tirilly, "N2s3, an Open-Source Scalable Spiking Neuromorphic Hardware Simulator," Université de Lille 1, Sciences et Technologies ; CRIStAL UMR 9189, report, Jan. 2017, 00001. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01432133/document

[22] A. N. Burkitt, "A Review of the Integrate-and-fire Neuron Model: I. Homogeneous Synaptic Input," *Biological Cybernetics*, vol. 95, no. 1, pp. 1–19, Jul. 2006. [Online]. Available: https://doi.org/10.1007/s00422-006-0068-6

[23] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *The 2011 International Joint Conference on Neural Networks*, Jul. 2011, pp. 1775–1781, iSSN: 2161-4393.

[24] P. Falez, P. Tirilly, I. M. Bilasco, P. Devienne, and P. Boulet, "Mastering the Output Frequency in Spiking Neural Networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2018, pp. 1–8, iSSN: 2161-4407.

[25] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks." in *ICLR*. OpenReview.net, 2019.

[26] H. Zhou, J. Lan, R. Liu, and J. Yosinski, "Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 3597–3607. [Online]. Available: http://papers.nips.cc/paper/8618-deconstructing-lottery-tickets-zeros-signs-and-the-supermask.pdf

[27] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[28] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," *arXiv:1702.05373 [cs]*, Mar. 2017, arXiv: 1702.05373. [Online]. Available: http://arxiv.org/abs/1702.05373