Robustness to Noisy Synaptic Weights in Spiking Neural Networks

Chen Li[1]
Department of Computer Science
The University of Manchester
Manchester, UK
chen.li-12@postgrad.manchester.ac.uk

Runze Chen[1]
Department of Computer Science
The University of Manchester
Manchester, UK
runze.chen@postgrad.manchester.ac.uk

Christoforos Moutafis
Department of Computer Science
The University of Manchester
Manchester, UK
christoforos.moutafis@manchester.ac.uk

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

***Abstract*—Spiking neural networks (SNNs) are promising neural network models to achieve power-efficient and event-based computing on neuromorphic hardware. SNNs inherently contain noise and are robust to noisy inputs as well as noise related to the discrete 1-bit spike. In this paper, we find that SNNs are more robust to Gaussian noise in synaptic weights than artificial neural networks (ANNs) under some conditions. This finding will enhance our understanding of the neural dynamics in SNNs and of the advantages of SNNs compared with ANNs. Our results imply the possibility of using high-performance cutting-edge materials with intrinsic noise as an information storage medium in SNNs.**

***Index Terms*—spiking neural networks, artificial neural networks, noisy weights, Gaussian noise**

I. Introduction

Spiking neural networks (SNNs) are high-level biologically-inspired neural models that use spiking neurons as information processing units and 1-bit spikes as information carriers, and they aim to achieve high computational performance by mimicking the human brain. Spiking neurons are sparsely activated and their firing time is usually not synchronized. These two features can make SNNs energy-efficient on neuromorphic hardware [1], [2], [3], [4] which is often event-based, asynchronous and highly-parallel.

So far, SNNs are still in the early stage, and many researchers are attempting to discover the advantages of SNNs compared with artificial neural networks (ANNs) which are currently more successful on many benchmarks [5], [6], [7]. Two potential advantages of SNNs have been found: First, SNNs can achieve better power efficiency when more powerful encoding methods are applied [8], or when SNNs are optimized for short latency [9]. Second, the inference results in SNNs can be gotten faster than ANNs, though the precision may be lower. The inference precision would then rise with more evidence accumulated over time [10]. This feature is potentially useful for dealing with the challenge of real-time processing in self-driving vehicles. However, we have just scratched the surface of realizing these advantages [11] and we do not know yet how to use these characteristics properly to make SNNs more competitive than their artificial neural network counterparts.

Previous research suggests that neural network models of the brain inherently contain noise and rely on the presence of noise to carry out their functions [12], [13]. In the context of deep spiking neural networks for pattern recognition, the integrate-and-fire mechanism in spiking neurons introduces subthreshold noise and over-threshold noise to the neural network [10]. Moreover, when using rate coding as an encoding method in SNNs, the input signal is typically noisy as well [11]. Even if SNNs inherently contain these noises, they could still complete inference with high accuracy in many benchmarks [14]. Existing research has systematically investigated the impacts of several kinds of noises to the performance of spiking deep belief networks [15]. However, the inference accuracy of these neural networks is not competitive, and the comparisons in this paper are limited in the same SNNs with different noise levels. By contrast, better results have been achieved by the technique of ANN-to-SNN conversion; meanwhile, ANN-to-SNN conversion gives an opportunity to compare ANNs and SNNs with the same architecture and synaptic weights.

Though several kinds of noise are investigated in [15], no study has reported the impact of noisy synaptic weights in SNNs up to now. In this paper, we fill this gap by exploring the tolerance of SNNs to Gaussian noise in synaptic weights. Moreover, we compare the robustness to noisy weights on SNNs and ANNs with the same network architecture, and for the first time indicate that SNNs are potentially more robust to noisy weights than ANNs. This positive finding enhances the understanding of inherent advantages of SNNs when compared with ANNs and will contribute to research which aims to apply

[1]These authors contributed equally to this research.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

noisy cutting-edge materials such as memristors and magnetic skyrmions to SNNs as information storage components for weights.

This paper is organized as follows: Section II provides the necessary background on artificial neural networks, spiking neural networks, and related techniques. Section III presents the experimental methodology and results on noisy weights. Section IV is the discussion and section V is the conclusion.

II. Background

A. Artificial Neural Networks

We produced our results on two kinds of feed-forward neural network: fully connected networks (FCNs) which are the simplest neural network structure, and convolutional neural networks (CNNs) which are more powerful models for pattern recognition tasks. In fully connected networks, every two adjacent layers are fully connected in a feed-forward pattern. The output of spiking neurons in the same layer will be sent to all neurons in the subsequent layer without any feedback connection to other layers as shown in Fig. 1(a). This feed-forward structure is efficient to deal with tasks that do not involve memory such as handwritten digit recognition [5].

If the connections between two adjacent layers in a FCN become localized and share the same kernels, it will turn into a CNNs as shown in Fig. 1(b). CNNs could achieve better inference results than FCNs with relatively fewer connections, and more local structure information of inputs could be maintained by convolutional kernels. Usually, a pooling layer would be added after a convolutional layer to subsample feature maps and reduce the number of parameters. In this paper, average pooling layers rather than max-pooling layers are adopted to make the conversion to SNNs easier.

Impressive results were achieved by using 5 hidden layers on FCNs [16] and 7 hidden layers on CNNs [17]. However, to keep the network simple and make it easy to be repeated by other researchers, we use a shallow structure and limited optimization techniques for both FCNs and CNNs (details in section III-A). These make our results suffer about 0.9% accuracy drop on FCNs [16] and about 0.7% accuracy drop on CNNs [17] compared with the state-of-the-art results on the MNIST dataset.

The stochastic gradient descent (SGD) algorithm will be used to train FCNs and CNNs. The performance of ANNs will be slightly different due to different initialized weights and the randomness in the SGD algorithm, thus the final inference accuracy of ANNs is averaged over 5 trials.

B. Activation Function

The nonlinear and differential activation functions determine the input-output relationship of analog neurons. The nonlinearity of activation functions is vital for representations in deep structures, and their differentiability benefits the use of backpropagation algorithms in ANNs. The rectified linear unit (ReLU) is one of the most frequently-used activation functions, and it is inspired by the response curve of spiking neurons [18]. ReLU can be approximated easier by spiking

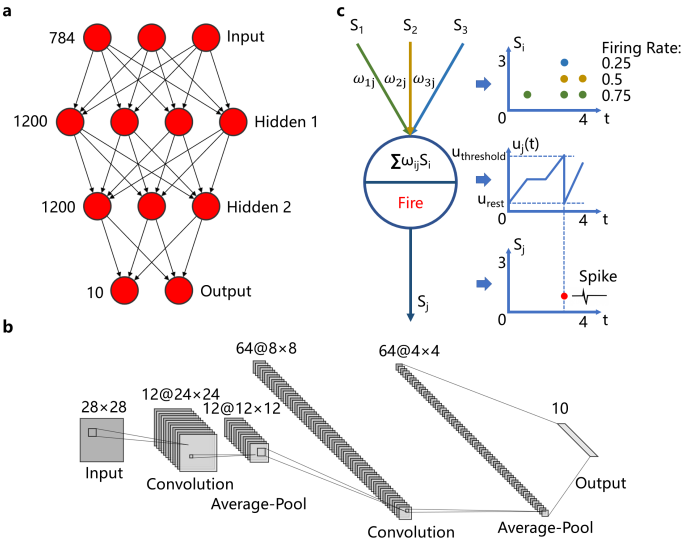

Fig. 1. The architecture of FCNs and CNNs, and the diagram of IF neural model.

neurons than other activation functions. The form of ReLU used in this paper does not contain biases, and is shown as follows:

$$y_j = max(0, \sum_i w_{ij} y_i) \tag{1}$$

where y_j is the output of an analog neuron, and y_i is the output of an analog neuron in the previous layer. w_{ij} denotes the synaptic weight from neuron i to j.

C. Spiking Neural Networks

Spiking neural networks are neural network models that closely mimic the information communication mechanism and neural dynamics in the human brain. Compared with ANNs, SNNs have an additional time dimension to process temporal information. Information in SNNs is represented by spikes, which are uniform electrical signals with precise spiking time. Spiking neurons are only activated when sufficient signals are integrated from other neurons, therefore the neural activities at the network level are usually sparse, which can make executing SNNs by matrix-based operation on a traditional CPU or GPU inefficient. Neuromorphic hardware is designed to accelerate the running of SNNs by topologically allocating spiking neurons and synapses to neuromorphic chips in analog [1], [2], [3] or hybrid [4] manners, and it can increase the power efficiency of SNNs by event-based computing.

D. Neuron Model

The integrate-and-fire (IF) model is one of the neural models used in spiking neural networks for pattern recognition tasks, and it is shown in Fig. 1(c). Spikes generated by presynaptic neurons are transmitted along synapses and then integrated by a postsynaptic neuron with different synaptic strengths which are represented by weights. The voltage in the postsynaptic neuron will be relatively stable if this voltage is below a threshold and no more electrical signals are integrated. When

the voltage inside the spiking neuron surpasses the threshold, a spike will be generated, and the internal voltage will drop down to the resting potential and wait for the next integration. The IF model used in this paper does not include a refractory period. The dynamics of IF model used in this paper are defined by:

$$u_j(t) = u_j(t - \Delta t) + \sum_i w_{ij} S_i(t - \Delta t) \tag{2}$$

where $u_j(t)$ is the internal voltage of spiking neuron j, and it is determined by two parts: The first part is the internal voltage of $u_j(t - \Delta t)$ at the previous moment $(t - \Delta t)$. Δt is the time resolution during SNN simulation. The second part is the input to the spiking neuron j in this moment. The range of the sum is all spiking neurons in the previous layer. w_{ij} denotes the synaptic weight of the neuron i in the previous layer to the neuron j in this layer. $S_i(t - \Delta t)$ has two states $\{0,\ 1\}$ and denotes whether a spike is generated in spiking neuron i at the time $(t - \Delta t)$. The time needed to transmit these weighted spikes to neuron j is Δt. The mechanism of spike generation is governed by:

$$S_j(t) = 1 \ and \ u_j(t) = u_{rest} \quad when \ u_j(t) > u_{threshold} \tag{3}$$

where u_{rest} is the resting potential of spiking neurons and it determines the stable state of these spiking neurons without external stimulus. $u_{threshold}$ is the threshold of spiking neuron j.

E. Input Encoding

Rate coding is used in this paper to generate input signals to SNNs. It is an encoding approach where the input firing rates are proportional to the intensity of a stimulus. The stimulus intensity in this paper represents the intensity of picture pixels in the handwritten benchmark MNIST. These pixels have an intensity between 0 to 1 as introduced in section II-H. The maximum pixel intensity has the highest firing rate $1/\Delta t$ where Δt is the time resolution of the SNN, and the minimum intensity 0 has the firing rate of 0. The corresponding relationship of firing rate and pixel intensity is linear.

F. Accuracy and Convergence Time

In the output layer, the inference result is the label of the neuron that has the highest firing rate. If more than one neuron has the highest firing rate, the neuron that has the higher voltage will be the final result.

Compared with ANNs, the output firing rate of SNNs usually changes over time so that their inference results will change over time. These fluctuations make it hard to record the inference accuracy and convergence time of SNNs precisely.

To deal with these difficulties, we adopt the following methods: First, the final accuracy of SNNs is the mean value of an accuracy range where the inference accuracy is converged in this range with some minor fluctuations. In our experiments, the fluctuation range is $\pm 0.03\%$. We believe that this method could record inference accuracy more precisely than directly recording the highest accuracy in the SNN simulation. Correspondingly, the convergence time is the time point when the inference accuracy is in this range for the first time and becomes stable after this time point. Second, to reduce the error in recording the convergence time, we use one more parameter to represent the convergence time. The time point when it is approaching the final accuracy, and within a certain percent accuracy loss from the final inference accuracy, will be recorded. For example, when the one percent accuracy loss convergence time is selected as the metric, if the final inference accuracy is 98.8%, the time point when its inference accuracy is higher than 97.8% for the first time would be recorded. Similarly, the 0.5% accuracy loss convergence time is the time point when the accuracy is higher than 98.3%. This method records the SNN convergence time more precisely than recording full accuracy convergence time, and gives a good indication of the convergence speed.

G. ANN-to-SNN Conversion and Threshold Tuning

ANN-to-SNN conversion is a relatively successful algorithm to achieve both high inference accuracy and short inference latency in SNNs [10]. Also, it provides an opportunity to compare the performance of ANNs and SNNs in the same network architecture and with the same trained weights. This algorithm will train ANNs by backpropagation, and then keep learned weights unchanged and replace analog neurons with spiking neurons. The analog inputs will be encoded as spike trains by rate coding.

After conversion, the threshold hyperparameter of the SNN needs to be tuned. There are many threshold normalization methods proposed to minimize the accuracy loss of ANN-to-SNN conversion [10], [19], [20]. These methods either adjust weights or tune thresholds. To keep the learned weights identical in ANNs and SNNs, we chose to tune thresholds by modifying *data-based normalization* in the previous paper [10]. The threshold in the layer l is set to D_l/D_{l-1}, where D_l is the maximum input of analog neurons in the layer l, D_{l-1} is the maximum input of analog neurons in the previous layer $l-1$. In particular, the threshold in the first hidden layer is D_2/D_1. Because D_1 is the maximum input in the input layer and it is equal to 1 in the MNIST dataset, the threshold in the first hidden layer is simplified to D_2.

In section III-D, we adopt different SNN thresholds to explore the influence of thresholds on inference accuracy and convergence time. The thresholds in this section are set by adding a scale factor σ to the threshold D_2 in the first hidden layer as $\sigma * D_2$ and keep thresholds in other layers unchanged. Through this weight scaling, the threshold in the first hidden layer will be σ times to the maximum input in this layer. According to the theory of *data-based normalization*, scaling the threshold in a layer will inversely scale the maximum inputs in all subsequent layers (This point is questioned by [20] because this theory is only correct when the activation in SNNs is the same as that in ANNs. However, this theory can still roughly estimate the maximum inputs, so we base our research on this theory to change thresholds and to test the

impact of different thresholds on network performance in noisy situations.). Hence thresholds in subsequent layers will be σ times to their maximum inputs as well. If σ is smaller than 1, the neurons in every layer will need less time to integrate information from the previous layer and generate a spike. On the contrary, if σ is bigger than 1, the neurons in every layer will need more time to integrate information from the previous layer and generate a spike.

H. MNIST Dataset

MNIST is a handwritten digit database for computer vision and pattern recognition [5]. It has a training set of 60000 examples and a testing set of 10000 examples. These examples are 28*28 pixel pictures of handwritten numbers between 0 to 9. Every pixel has a greyscale value between 0 and 1.

I. Noisy Weights

The information in neural networks is stored in synapses between neurons as weights. These weights can be made noisy either in the training phase, to increase the robustness of the neural network [21] or in the testing phase when the synapse storage materials is noisy at room temperature [22], [23]. The current storage techniques used in CPUs and GPUs do not result in noisy weights, but noise could become an issue when more advanced storage materials such as memristors and magnetic skyrmions are used to implement neural networks [22], [23]. These two materials have the excellent characteristics of non-volatility and nanoscale size so that they have the advantages of power efficiency and high integration. These advantages make memristors and magnetic skyrmions good candidates to deal with the challenges of high power-dissipation in neural networks and the continuation of Moore‘s law. Memristors and magnetic skyrmions have been applied to SNNs by theoretical simulation and experimental investigation [24], [25]. However, these cutting-edge materials may contain non-ignorable random noise at room temperature, which is typically Gaussian distributed.

Previous research has investigated how random variance in synaptic weights affects the inference accuracy of spiking neural networks [15]. This random variance originates from the physical quantity of materials and the fabrication of transistors therefore this variance is fixed and will not change over time. By contrast, the noise in synaptic weights discussed in this paper is the random noise that exists in future high-performance materials such as memristors and magnetic skyrmions at room temperature. This random noise has a certain distribution and its value will change over time.

The noise type investigated in this paper is Gaussian noise. The key parameters of Gaussian noise are its mean and standard deviation (SD). The mean value of Gaussian noise is zero in this paper. We considered three different ways to choose the standard deviation: SD is fixed in different synaptic weights; SD is proportional to the value of weights; SD is proportional to the square root of weights. This paper uses the second method to determine the standard deviation.

Adding noise to synaptic weights will introduce uncertainty to ANNs and SNNs, and make their inference results different for different trials. Hence, all results of ANNs and SNNs will be averaged over 5 trials in our experiments. This average operation is independent of the average operation during the training of ANNs as mentioned in section II-A. For example, when ANNs are averaged over 5 trials during training, the total trial number for noisy weights on ANNs and SNNs will both be 25.

III. Experiments and results

A. Experimental Setup

The FCNs are trained in MATLAB using a stochastic gradient descent algorithm [10]. These networks have a structure of 784-1200-1200-10 as shown in Fig. 1(a), their training parameters are shown in Tab. I. All results are averaged over 5 trials, and the average inference accuracy of FCNs is 98.77% on the test dataset.

After training, all analog neurons are replaced by spiking neurons, and the weights are kept unchanged. The time resolution of SNNs is set to 1ms and the max input firing rate is set to 1000Hz correspondingly. The thresholds are set to D_l/D_{l-1} as illustrated in section II-G. The simulation time is 1000ms.

The CNNs have the structure of 28x28-12c5-2s-64c5-2s-10o as shown in Fig. 1(b) and they are trained in MATLAB using a stochastic gradient descent algorithm [10]. The training parameters are shown in Tab. 1. All results are averaged over 5 trials, and the average inference accuracy of the CNNs is 99.09% on the test dataset.

After ANN-to-SNN conversion, all parameters and hyperparameters in the spiking CNNs are the same as those used in spiking FCNs.

TABLE I
Training parameters for FCNs and CNNs

Training parameters	FCNs	CNNs
Learning rate	1	1
Momentum	0.5	0
Dropout rate	0	0
Training epochs	20	30
Batch size	100	50

B. Inference Accuracy

The comparison of ANNs and SNNs for different noisy weights on FCNs is shown in Fig. 2. The X-axis represents the ratio of the Gaussian noise's standard deviation to synaptic weights. The noise level of zero percent on the X-axis represents noise-free synaptic weights. In this figure, the recognition accuracy of the SNNs is slightly lower than that of the ANNs when the noise level is 0%. However, with the increase of noise level, the inference accuracy of the ANNs drops dramatically. When the noise level is 100%, the inference accuracy of the ANNs is only 94.74% on average, and their standard deviation shows an increasing trend for

higher noise levels. By contrast, the inference accuracy of the SNNs keeps stable for all noise levels at around 98.76%, and their standard deviation is smaller than that of ANNs when the noise level is greater than zero percent. When the noise level is 20%, the accuracy of the SNNs has surpassed that of the ANNs.

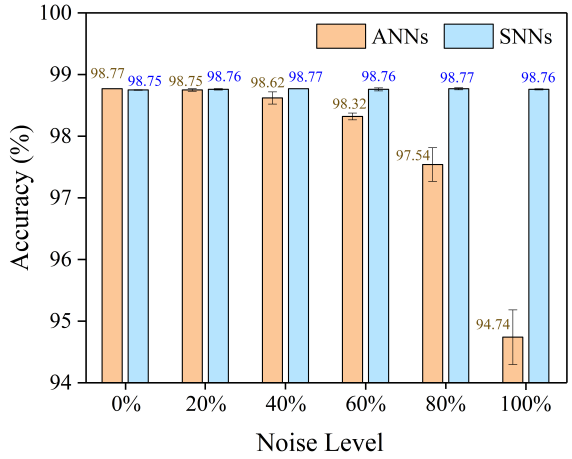

Fig. 2. Accuracy comparison of the ANNs and the SNNs in the architecture of FCNs for different noise levels.

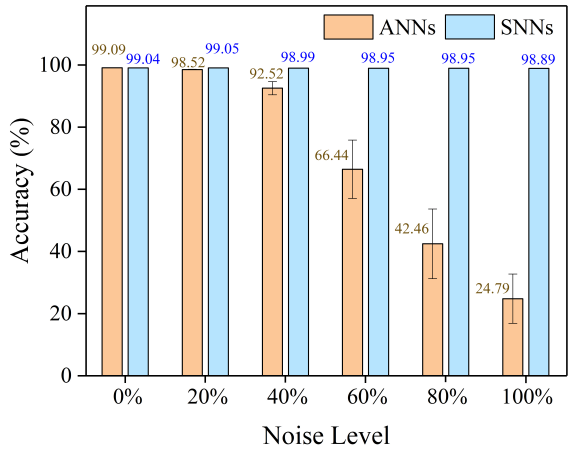

Fig. 3. Accuracy comparison of the ANNs and the SNNs in the architecture of CNNs for different noise levels.

Fig. 3 is the comparison of the ANNs and the SNNs for noisy weights on CNNs. The inference accuracy of the ANNs is bigger than SNNs when the noise level is 0%. When the noise level is 20%, the accuracy of the ANNs drops below 99%, but the accuracy of the SNNs is still above 99%. The accuracy of the ANNs decreases dramatically and their standard deviation increases to about 10% when the noise level is greater than 40%. By contrast, the accuracy of the SNNs is still greater than 98.8% even when the noise level is 100%, and their standard deviation remains small for all noise levels.

C. Convergence Time

The inference accuracy of SNNs represents their inference performance and the efficiency of ANN-to-SNN conversion, while the convergence time of SNNs reflects their energy efficiency. If the convergence time is too long, SNNs will lose their essential advantages of power efficiency and fast inference and will be not suitable to be applied to neuromorphic hardware and other cutting-edge materials. The convergence time of spiking FCNs for different noise levels with their standard deviation is shown in Tab. II.

In this table, the convergence time increases roughly with the noise level, as does the standard deviation of convergence time. The convergence time of one percent accuracy loss is relatively stable at around 16ms and shows minor increases for higher noise levels.

TABLE II
THE CONVERGENCE TIME OF THE SPIKING FCNS FOR DIFFERENT NOISE LEVELS

Noise level	Convergence time (ms)	Convergence time of 1% loss (ms)
0%	30.8±4.6	16±0.0
20%	40.2±3.1	16±0.0
40%	41.4±9.8	16±0.0
60%	60.2±26.3	16.8±0.3
80%	69.0±16.4	17.0±0.0
100%	64.8±18.2	17.6±0.9

Tab. III illustrates the convergence time of spiking CNNs. We can see from this table that the convergence time of the high noise level is significantly longer than that of the low noise level. The convergence time with one percent accuracy loss shows a similar trend. The standard deviation increases for higher noise levels as well.

TABLE III
THE CONVERGENCE TIME OF THE SPIKING CNNS FOR DIFFERENT NOISE LEVELS

Noise level	Convergence time (ms)	Convergence time of 1% loss (ms)
0%	206.5±32.5	28±0.0
20%	196.8±60.2	29.6±1.1
40%	222.0±57.2	39.6±1.7
60%	312.0±76.0	71.7±1.7
80%	654.8±168.6	111.4±9.1
100%	701.3±134.8	173.5±11.3

D. Different Thresholds

The thresholds in SNNs will significantly affect their inference accuracy and convergence time. The threshold in this paper is set by modifying *data-based normalization* [10], in the expectation that this threshold normalization could achieve fast inference and high accuracy. The optimal method to set thresholds may be different when the weights in the SNN are noisy. Driven by that, we are going to investigate how the

inference accuracy and convergence time are influenced by different thresholds with different Gaussian noise levels.

We choose the scale factor σ of 0.2, 0.5, 1 and 2 in this section, and scale the thresholds according to the method in II-G. The accuracy of the spiking CNNs is shown in Fig. 4. The accuracy of the spiking FCNs is stable at around 98.77% for all scale factors and noise levels so it will not be presented here. We can see from Fig. 4 that the inference accuracy for all noise levels rises with the increasing of scale factor σ in the first hidden layer. This indicates that the networks with high thresholds tend to have a higher inference accuracy. As for inference time, SNNs adopting scale factors of 0.2, 0.5 and

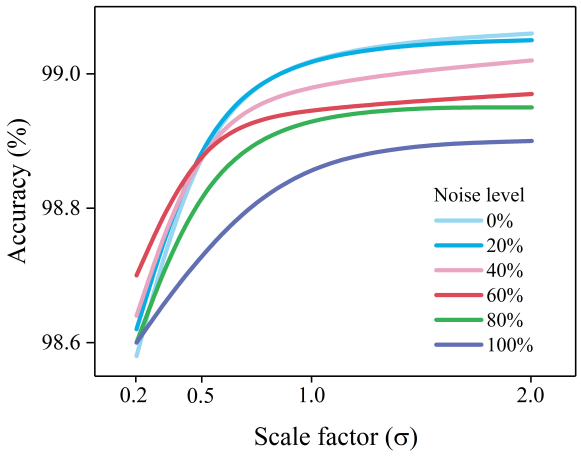

Fig. 4. The relationship of inference accuracy for different σ.

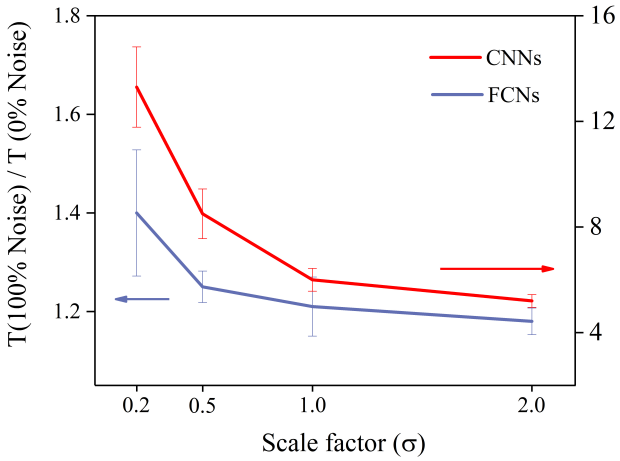

Fig. 5. The ratio of convergence time of 100% noise and 0% noise on FCNs and CNNs.

2 show a similar increasing trend for different noise levels as when the scale factor is 1. This has been shown in Tab. III and will not be summarised again here. Fig. 5 illustrates the relationship of the convergence time ratio to the scale factor σ. This ratio is calculated by dividing the convergence time at 100% noise level by the convergence time at 0% noise level, and it represents how much additional time is needed to cope with Gaussian noise on the weights before convergence. We use the 1% accuracy loss convergence time as the metric for CNNs and the 0.5% accuracy loss convergence time as the metric for FCNs. The reason to choose 0.5% accuracy loss but not 1% accuracy loss for FCNs is that the FCNs converge so fast that the 1% accuracy loss convergence time is similar at all noise levels and is hard to show differences. Meanwhile, the convergence time of 0.5% accuracy loss is big enough to show differences and is more stable than using full accuracy convergence time as a metric. In Fig. 5, the network with a higher scale factor has a lower convergence time ratio both in FCNs and CNNs.

IV. Discussion

A. Inference Accuracy

The inference accuracy of ANNs drops considerably with the increase of noise level in synaptic weights. The reason why ANNs are vulnerable to noisy weights is that the noisy weight in ANNs is similar to the weight variance investigated in [15]. During the inference phase, the noisy weights in ANNs are used once in the feed-forward propagation, and the Gaussian noise will only affect the value of synaptic weights once. This Gaussian noise is effectively a random offset which is fixed over time. A high noise level will dramatically change the value of synaptic weights and affect the function of ANNs, so the final inference accuracy of these ANNs will drop drastically. In addition, a high noise level will introduce more uncertainty in synaptic weights, so the standard deviation of they inference accuracy will increase too.

By contrast, SNNs have the concept of time, and weights with Gaussian noise in SNNs will be used several times during simulation, so the impact of Gaussian noise will be minimized with more spikes transmitted across this synapse. To illustrate it more clearly, we assume a weight with the value w and the Gaussian noise $G(0, \sigma)$. The number of spikes transmitted by this synapse is n. Hence, the total effective information transmitted by this synapse is $w{*}n$, and the total noise is $\sum_n G(0, \sigma)$. The value of $w{*}n$ will increase with more spikes transmitted. However, the value of $\sum_n G(0, \sigma)$ equals to $G(0, \sqrt{N}\sigma)$ according to standard statistics. It means that its standard deviation increases slower than accumulated weights $w{*}n$. As a result, the signal-noise ratio $w{*}n/\sum_n G(0, \sigma)$ will increase with the number of spikes transmitted.

B. Convergence Time

The accuracy of the SNNs investigated in section III-B is higher than 98.7% for all noise levels. While SNNs could achieve high inference accuracy at high noise level, their convergence time still goes up at high noise level both on the architecture of FCNs and CNNs as shown in section III-C. This indicates that though SNNs can minimize the impact of noise by averaging noise over time as illustrated in section IV-A, Gaussian noise on synaptic weights still poses difficulties for

SNNs and push SNNs to require more time to handle the noise before convergence.

C. Influence of Thresholds

In section IV-A, we explain why SNNs are more robust than ANNs to noisy weights from a signal-noise ratio perspective. However, the information transmitted by weights will not be completely obtained by postsynaptic neurons due to sub-threshold noise and over-threshold noise. Hence, the choice of thresholds will influence SNNs' robustness to noisy weights. Figures 4 and 5 illustrated the impact of the scale factor on inference accuracy and convergence time. We can see that the robustness to noisy weights is clearly affected by the scale factor on the threshold, and the network with a higher scale factor has better robustness to noisy weights. This is because in SNNs, the information in weights can only be transmitted to deeper layers through spikes. The information contained by a spike is always equivalent to the integrated voltage $(u_{threshold} - u_{rest})$ in a spiking neuron no matter how much voltage is cut off by the threshold. If the threshold is small, the relative error introduced by this over-threshold noise will increase. This error will make it difficult for spiking neurons to discriminate the original value of the weights to Gaussian noise. In this situation, the signal-noise ratio illustrated in section IV-A will not obviously increase with the number of spikes transmitted through a synapse.

D. Limitations

This study is limited in three ways. Firstly, the investigated neural network structures are shallow FCNs and CNNs, and the benchmark used in this paper is a basic computer vision benchmark. More experiments in deeper neural network architectures [26] and harder benchmarks [6], [7] are required. A difficulty in our experiments is that the ANNs and SNNs need to be repeated many times to cope with the noisy elements in the network and get the averaged results, which will make it extremely time-consuming in deeper neural network architectures and bigger benchmarks. Secondly, in this paper, we only investigated the robustness of neural networks to Gaussian noise with a standard deviation proportional to the value of the synaptic weights. In practice, the random noise found in advanced materials may have different distributions. Future work is required to establish systematic studies of different noise types. Also, to conduct more systematic studies, a potential research direction is expanding results to different spiking neuron models (e.g. leaky integrate-and-fire model) and different encoding methods (e.g. temporal coding). Thirdly, the inference accuracy and the convergence time are different for FCNs and CNNs for the same noise level. According to the results in section IV-A and section IV-B, the accuracy of CNNs drops faster than FCNs with the increasing of noise level for both ANNs and SNNs. Meanwhile, the convergence time of SNNs grows faster in CNNs than in FCNs. These different performances show the different tolerances to noisy weights in FCNs and CNNs and stem from their different neural network architectures, which has not been thoroughly explored in this paper.

V. Conclusions

Few practical advantages of SNNs have been found up to now. In this paper, we provide the first comprehensive assessment of the effect of Gaussian noise on synaptic weights and found that SNNs are very robust to Gaussian noise on synaptic weights. This interesting property suggests the possibility of the application of noisy materials such as memristors and magnetic skyrmions to SNNs as information storage components.

Research in SNNs is still in the early stages and a substantial amount of the research is driven by exploring the differences between SNNs and currently more performant ANNs. We show that SNNs have a greater tolerance to noisy weights than ANNs, which could be considered an advantage over conventional ANNs. This robustness to noise comes from the time dimension of SNNs as illustrated in section IV-A and the integrate-and-fire mechanism of spiking neurons as illustrated in section IV-C, which furthers our understanding of the characteristics of SNNs.

VI. Acknowledgments

We thank Edward Jones and Gengting Liu for their paper review. C. L. thanks Luhui Liu for encouragement and review for this paper. The research leading to these results has received funding from the EU Flagship Human Brain Project (H2020 785907).

References

[1] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[2] 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*, vol. 345, no. 6197, pp. 668–673, 2014.

[3] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[4] 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 *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, 2010, pp. 1947–1950.

[5] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[8] H. Mostafa, B. U. Pedroni, S. Sheik, and G. Cauwenberghs, "Fast classification using sparsely active spiking networks," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[9] J. Wu, Y. Chua, M. Zhang, Q. Yang, G. Li, and H. Li, "Deep spiking neural network with spike count based learning rule," *arXiv preprint arXiv:1902.05705*, 2019.

[10] 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)*. IEEE, 2015, pp. 1–8.

[11] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience*, vol. 12, 2018.

[12] L. Buesing, J. Bill, B. Nessler, and W. Maass, "Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons," *PLoS computational biology*, vol. 7, no. 11, p. e1002211, 2011.

[13] E. T. Rolls and A. Treves, "The neuronal encoding of information in the brain," *Progress in neurobiology*, vol. 95, no. 3, pp. 448–490, 2011.

[14] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Networks*, 2018.

[15] E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, and S.-C. Liu, "Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms," *Frontiers in neuroscience*, vol. 9, p. 222, 2015.

[16] D. Claudiu Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets excel on handwritten digit recognition," *arXiv preprint arXiv:1003.0358*, 2010.

[17] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[18] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[19] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017.

[20] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, 2019.

[21] G. Hinton and D. Van Camp, "Keeping neural networks simple by minimizing the description length of the weights," in *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*. Citeseer, 1993.

[22] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, p. 80, 2008.

[23] W. Jiang, P. Upadhyaya, W. Zhang, G. Yu, M. B. Jungfleisch, F. Y. Fradin, J. E. Pearson, Y. Tserkovnyak, K. L. Wang, O. Heinonen *et al.*, "Blowing magnetic skyrmion bubbles," *Science*, vol. 349, no. 6245, pp. 283–286, 2015.

[24] C. Sung, H. Hwang, and I. K. Yoo, "Perspective: A review on memristive hardware for neuromorphic computation," *Journal of Applied Physics*, vol. 124, no. 15, p. 151903, 2018.

[25] M.-C. Chen, A. Sengupta, and K. Roy, "Magnetic skyrmion as a spintronic deep learning spiking neuron processor," *IEEE Transactions on Magnetics*, vol. 54, no. 8, pp. 1–7, 2018.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.