Synaptic 1/*f* noise injection for overfitting suppression in hardware neural networks

NEUROMORPHIC
Computing and Engineering

RECEIVED
30 October 2021

REVISED
3 April 2022

ACCEPTED FOR PUBLICATION
5 May 2022

PUBLISHED
5 August 2022

PAPER

Synaptic 1/*f* noise injection for overfitting suppression in hardware neural networks

Yan Du[1,2,6], Wei Shao[3,4,6], Zheng Chai[1,2,*], Hanzhang Zhao[1,2], Qihui Diao[1,2], Yawei Gao[1,2], Xihui Yuan[1,2], Qiaoqiao Wang[1,2], Tao Li[1,2], Weidong Zhang[5], Jian Fu Zhang[5] and Tai Min[1,2,*]

[1] Center for Spintronics and Quantum Systems, State Key Laboratory for Mechanical Behavior of Materials, and School of Materials Science and Engineering, Xi'an Jiaotong University, Xi'an, People's Republic of China
[2] Peng Cheng Laboratory, Guangdong, People's Republic of China
[3] School of Computing Technologies, RMIT University, Melbourne, VIC 3001, Australia
[4] School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, United States
[5] School of Engineering, Liverpool John Moores University, L3 3AF Liverpool, United Kingdom
[*] Authors to whom any correspondence should be addressed.
[6] Authors with equal contribution.

E-mail: zheng.chai@xjtu.edu.cn and tai.min@mail.xjtu.edu.cn

Keywords: neuromorphic computing, 1/*f* noise, overfitting, hardware neural network

Abstract

Overfitting is a common and critical challenge for neural networks trained with limited dataset. The conventional solution is software-based regularization algorithms such as Gaussian noise injection. Semiconductor noise, such as 1/*f* noise, in artificial neuron/synapse devices, which is often regarded as undesirable disturbance to the hardware neural networks (HNNs), could also play a useful role in suppressing overfitting, but that is as yet unexplored. In this work, we proposed the idea of using 1/*f* noise injection to suppress overfitting in different neural networks, and demonstrated that: (i) 1/*f* noise could suppress the overfitting in multilayer perceptron (MLP) and long short-term memory (LSTM); (ii) 1/*f* noise and Gaussian noise performs similarly for the MLP but differently for the LSTM; (iii) the superior performance of 1/*f* noise on LSTM can be attributed to its intrinsic long range dependence. This work reveals that 1/*f* noise, which is common in semiconductor devices, can be a useful solution to suppress the overfitting in HNNs, and more importantly, further evidents that the imperfectness of semiconductor devices is a rich mine of solutions to boost the development of brain-inspired hardware technologies in the artificial intelligence era.

1. Introduction

In the artificial intelligence (AI) era, brain-inspired deep neural networks have demonstrated substantial potential in various neuromorphic tasks such as visual recognition, natural language processing, and autonomous driving [1–7]. Despite the remarkable progress, those neural networks often encounter the underfitting and overfitting problems, both resulting in unsatisfactory accuracy: underfit leads to high prediction errors for both training and test data, while overfit leads to a very low prediction error on the training data but a very high prediction error on the test data [8–11], as schematized in figure 1.

Underfitting happens because the neural network is too simple to capture all the features in the training data. The practical solutions can be simply training the network for a longer duration or just use a network with higher complexity. Overfitting happens because the neural network is too complex for a limited training data size, forcing the network to overly memorize the irrelevant detail and noise in the training data. Of course, increasing the size of training data could be the most straightforward solution. However, in real-world situations, the training data size is often limited by time, budget or technical constrains [12], making overfitting practically more difficult to deal with than underfitting [13].

Recently, a group of techniques, collectively referred to as 'regularization', which is the process of shrinking the coefficients in neural networks, have been used to select the networks' complexity by automatically

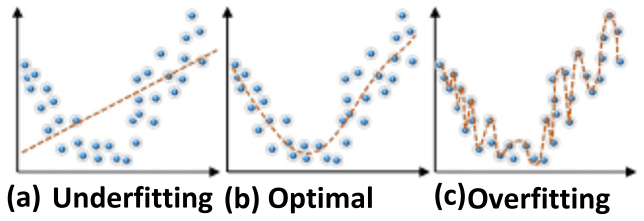

Figure 1. (a)–(c) Schematics of (a) underfitting, (b) optimal training and (c) overfitting of neural networks.

penalizing features that make the network too complex. Using regularization, the learning algorithms of neural networks are modified to reduce its generalization error but not its training error [14]. The most common regularization methods include [15]:

(a) Early stopping: stop training automatically when a specific performance measure stops improving;
(b) Weight decay: incentivize the network to use smaller weights by adding a penalty to the loss function;
(c) Dropout: randomly ignore certain nodes in a layer during training;
(d) Model combination: average the outputs of separately trained neural networks;
(e) Noise injection: allow some random fluctuations in the data through augmentation.

Among them, noise injection is a very popular method against overfitting [16]. The addition of noise during training has a regularization effect and improves the robustness of the neural network [17]. In practice, noise can be added in between training iterations and onto different parts of the neural networks, such as input signal, weights and activation functions, to make it difficult for the network to find a solution that fits precisely to the original training data, and thereby reduces overfitting. In software-based DNNs, noise injection is normally realized with the addition of a separate zero-mean Gaussian noise layer, such as the 'tf.keras.layers.GaussianNoise' in TensorFlow [18].

The software-based neural networks, which are still based on the traditional von Neumann architecture initially designed for sequential computing [19], are challenged by the proliferation of massive data in terms of computing ability and power consumption, driving people to look for alternative solutions. Again, inspiration comes from the brain: the power budget of the human brain is around 20 W, and its computation capabilities range in the 10^{17} FLOPS, equivalent to the best supercomputers [20]: the world's fastest supercomputer in 2021, Fugaku, has a computation capability is 4.42×10^{17} FLOPS, but with a power of 29 899.23 kW [21].

In recent years, there has been a large push toward a hardware implementation of artificial neural networks, i.e. hardware neural networks (HNNs), aiming to overcome the calculation complexity of software-based implementations by using semiconductor technology to directly emulate the behaviour of neurons and synapses [22–25]. Unlike the conventional von-Neumann architecture that is inherently sequential in nature [19], HNNs profit from massively parallel processing, and various architectures, such as multilayer perceptron (MLP), convolutionary neural network, recurrent neural network (RNN) and long short-term memory (LSTM) have been proposed using semiconductor devices (transistors, memristors, etc) and circuits.

Since HNNs are implemented with real-world devices, the natural-existing imperfectness of devices inevitably affects the performance of HNNs. Previously, such imperfectness was considered as detrimental factors that bring undesirable disturbance to HNN's parameters, causing variation and drift to the performance [22–24]. However, as the brain is of high error tolerance and so should be HNN, what is more attracting is that such intrinsic imperfectness of semiconductor devices might be utilized to, instead, improve the performance of HNNs. For example, the stochastic memristive switching behaviour has been used to realize the dropout function of HNN [25]. The intrinsic read noise of memristive devices has been used to prevent HNN from getting trapped into local minima and thus converge to sub-optimal solutions [26, 27].

Motivated by the previous explorations, it is natural to link semiconductors noise to overfitting suppression in HNNs. An obvious benefit is that the intrinsic noise in devices waives the necessity to design complex circuitry specialized for Gaussian noise generation using Zener diodes or other devices [28]. Various types of noise exist in semiconductor devices, such as thermal noise [29], random telegraph noise [30], $1/f$ noise [31], etc [32, 33], but a comprehensive study on the overfitting suppression effect of noise, at least for one or two types of noise, is still missing.

Among those noises, $1/f$ noise is the low frequency noise for which the noise power spectral density is inversely proportional to the frequency [34, 35]. It can be observed in a wide range of semiconductor devices, such as transistors [36] memristors [37–40], diodes [41], and photoelectric devices [42]. $1/f$ noise is also the 'background noise' of the brain [43]. For example, the channel noise in neurons, which is thought to arise from the random opening and closing of ion channels in the cell membrane, is seen to be $1/f$ [44]. Similarly,

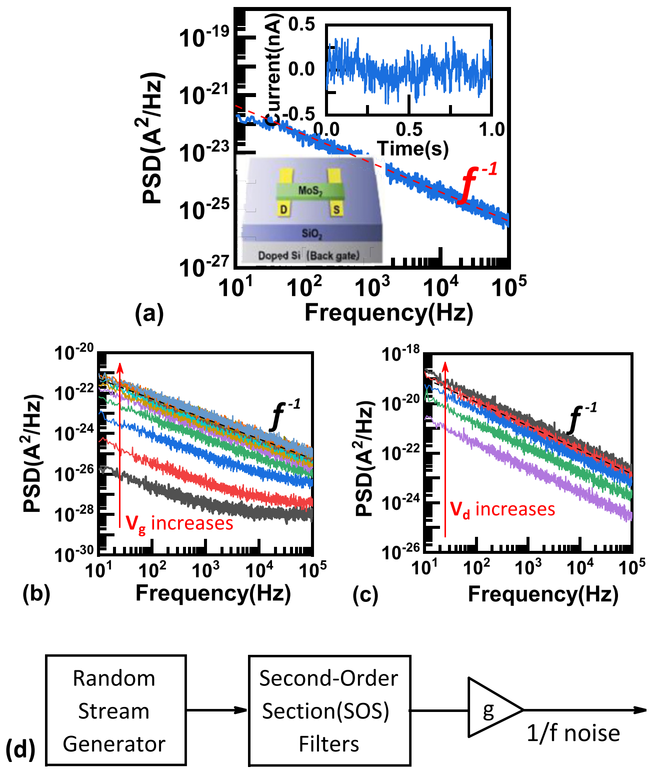

Figure 2. (a) Frequency domain and (upper inset) corresponding time domain $1/f$ noise measured via electrical characterization. (Lower inset) Schematic of the back-gated MoS$_2$ FET device, from which the demonstrative $1/f$ noise is measured. (b) and (c) Experimental frequency domain $1/f$ noise measured from this MoS$_2$ device, under (b) various V_{gs} and (c) various V_{ds}. $1/f$ noise can be easily tuned with V_{gs} or V_{ds}. (d) Schematic of time-domain $1/f$ noise generation in Matlab, using the pinknoise.m function.

it has been shown that both magnetoencephalography and electroencephalogram recordings of spontaneous neural activity in humans displayed $1/f$-like power spectra in the α, μ, and β frequency ranges [45]. $1/f$ noise is also an optimal communication channel for complex networks as in art or language and may therefore be the channel through which the brain influences complex process and is influenced by them [46]. This inspires people to wonder if the $1/f$ noise in real semiconductor devices could be used to mimic some natural neural behavior in human brains, and play a role in the HNN.

From the mathematical perspective, $1/f$ noise is well-known for its ‘memory’, or long-range dependence, which basically refers to the level of statistical dependence between two points in the time series [47]. More specifically, it relates to the rate of decay of statistical dependence between the two points if the distance between them increases. For example, if a time series has a short memory, it is predictable from only its immediate past. The memory of a time series can be expressed using the autocorrelation. Autocorrelation refers to the correlation of a given signal with itself at various points in time [48]. For a time series with short memory, its autocorrelations decay quickly as the number of intervening observations increases. $1/f$ noise is an intermediate between white noise (a process without memory) and brown noise (a process with an infinite memory) [49]. The long-term memory of $1/f$ noise can be quantified using the autocorrelation function (ACF).

In this work, we proposed the idea of noise injection on HNNs by using the intrinsic $1/f$ noise in semiconductor devices. We demonstrated the overfitting suppression ability of $1/f$ noise in MLP and LSTM for handwriting data recognition and weather prediction tasks, and attribute the superior performance of $1/f$ noise on LSTM, which is used to process time series data, to the long range dependence of $1/f$ noise. This work reveals that $1/f$ noise in semiconductor devices can be a useful solution to suppress overfitting in HNNs, and inspires that the imperfectness of semiconductor devices is a rich mine of solutions to boost the development of brain-inspired hardware technologies.

2. Noise measurement and simulation

For the purpose of demonstration, experimental $1/f$ noise is measured from the drain current in a back-gated MoS$_2$ field effect transistor (FET) (figure 2(a)), in both time domain and frequency domain. The channel length is 5 μm, width is 19.4 μm and the MoS$_2$ has nine layers. $1/f$ noise can be easily tuned with V_{gs} (figure 2(b)) or V_{ds} (figure 2(c)).

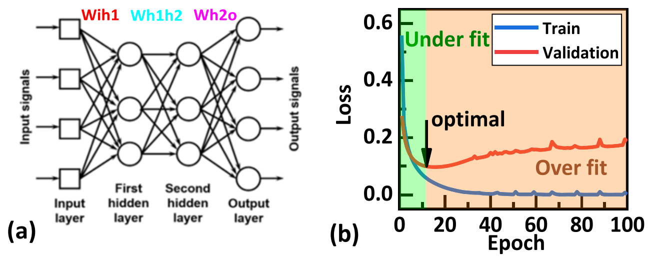

Figure 3. (a) An MLP neural network with 784, 50, 50 and 10 neurons in the input layer, 1st hidden layer (tanh), 2nd hidden layer (tanh) and output layer (relu). (b) Overfitting of the MLP showing increasing validation loss after the optimal point.

It should be emphasized that, due to the huge number of neurons and synapses in a neural network, in this work, noise simulated with software was used instead of experimental data measured from practical devices, to study the impact of noise injection on overfitting suppression. The 'pinknoise.m' function in Matlab [50], which includes a random stream generator, a series of randomly initiated second-order section (SOS) filters and a gain, was used to generate a time-domain $1/f$ noise, as schematized in figure 2(d). The Gaussian noise was simulated using the 'randn.m' function in Matlab [51].

3. Multi-layer perceptron (MLP) simulation

Multi-layer perceptron (MLP) is a popular and practical neural network model consisting of three types of layers—the input layer, output layer and hidden layer [52]. In a MLP the data flows in the forward direction from input to output layer, while the synapses in the MLP are trained with the back propagation learning algorithm. The major use cases of MLP are pattern classification, recognition, prediction and approximation. The MLP is sometimes called a 'memoryless' classifier because if one presents a pattern on its input units, the output units respond with an activation pattern, and those outputs depend only on the inputs at that moment, regardless of the previous input history.

In this work, an MLP with two hidden layers was simulated using Python, with 784, 50, 50 and 10 neurons in the input layer, 1st hidden layer (tanh), 2nd hidden layer (tanh) and output layer (relu), respectively. The MLP was trained and validated using the Modified National Institute of Standards and Technology handwritten digit database [53], in which 60 000 images were used for training and the other 10 000 were used for validation. During training and validation, the batch size is 100 and the learning rate is 0.0005. The loss function is cross entropy loss and the optimizer is Adam.

Overfitting is clearly realized in this MLP: after training starts, both the training loss and validation loss decreases (underfit), until at around 10th epoch when the training loss keeps decreasing but the validation loss reaches its lowest point (optimal) and started to increase. After that, the training loss keeps decreasing while the validation loss keeps increasing, which is a typical feature of overfitting, as shown in figure 3(b). To evaluate the impact of $1/f$ noise on overfitting suppression, a simulated time-domain $1/f$ noise, whose amplitude is calculated according to

$$\mathbf{SNR} = 10\mathbf{log}(\mathbf{P}_{\mathbf{signal}}/\mathbf{P}_{\mathbf{noise}}) \tag{1}$$

where the SNR refers to a fixed signal-to-noise ratio (SNR) and the signal is the weight value updated after back propagation in each epoch, is added to the weight before validation, as schematized in figure 4(a). For comparison, Gaussian noise with the same SNR is injected in the same way.

The simulated $1/f$ noise is applied onto the three layers of weights, i.e. the weights between the input layer and the 1st hidden layer (Wih1), between the 1st and 2nd hidden layer (Wh1h2) and between the 2nd hidden layer and the output layer (Wh2o), respectively (figure 4(b)). Obviously, the location of noise injection makes major differences: noise injection on Wih1 lead to converged training and validation curves but with higher final loss for both. For noise injection on Wh1h2 the overfitting is even worse. For noise injection on Wh2o, the training and validation curve are closer and the final validation loss is ~50% lower than the initial level, indicating that the overfitting has been suppressed with the injection of $1/f$ noise. The training and validation loss after 100 epochs are summarized in figure 4(c). As shown in figure 4(d), Gaussian noise with the same SNR, shows similar effect as the $1/f$ noise, probably due to the fact that since MLP is static and has a memoryless network architecture [54], it does not respond differently to $1/f$ noise or Gaussian noise, as shown in figure 5(b).

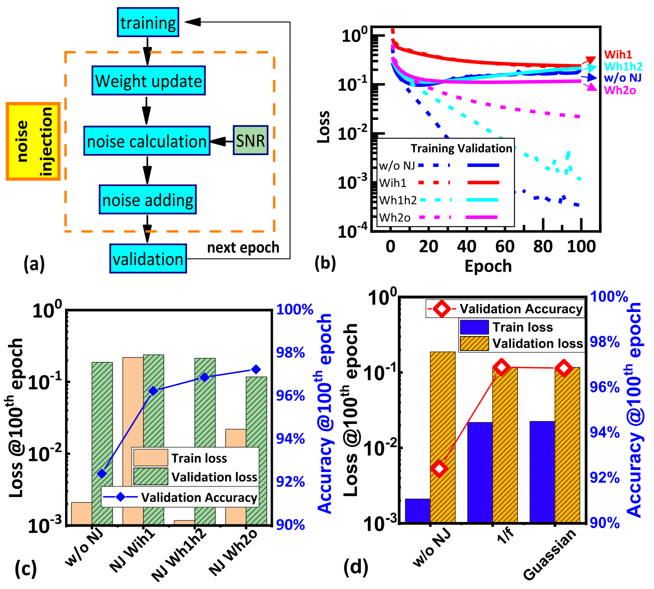

Figure 4. (a) The flowchart of noise injection in the MLP. (b) $1/f$ noise is added onto the weights between input layer and 1st hidden layer, between 1st and 2nd hidden layer, and between 2nd hidden layer and output layer, respectively. (c) The training loss validation loss and accuracy of different weights after 100 epochs. (d) The training loss, validation loss and accuracy of $1/f$ noise and Gaussian noise injection after 100 epochs. NJ is short for noise injection.

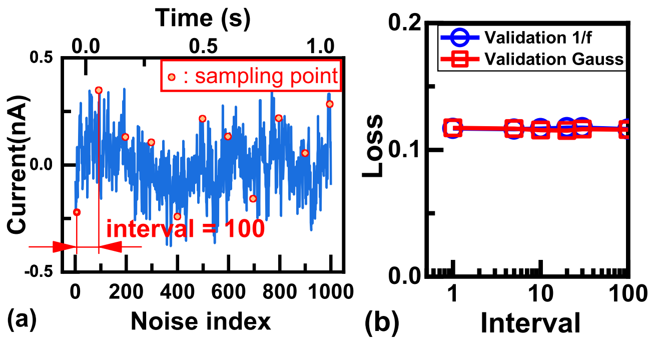

Figure 5. (a) Demonstration of noise sampling, with interval of 100. (b) Loss after using $1/f$ noise and Gaussian noise injection in an MLP. Noise is injected on to the weights between the 2nd hidden layer and the output layer.

4. Long-short term memory (LSTM) simulation

Artificial neural networks are expected to mimic the architecture and performance of human thoughts which have persistence. For example, the reader of this paper understands each word based on the understanding of previous words, instead of throwing everything away and start thinking from scratch again. However, traditional neural networks, such as MLP, cannot do this, which is a major shortcoming. RNNs, which has loops inside and allows information to persist, addresses this issue. In practice, it is found that RNN can learn to use the past information well if the gap between the relevant information and the place that it is needed is small. If such gap becomes large, which is entirely possible, RNNs become less capable of learning to connect the information, due to some fundamental reasons, such as vanishing or exploding gradient.

LSTM networks are a special kind of RNN explicitly designed to avoid the long range dependency problem. In standard RNNs, the repeating module has a very simple structure, such as a single tanh layer. For LSTMs, the repeating module has a different structure, consisting of a cell, an input gate, an output gate and a forget gate. The forget gate allows unneeded information to be erased and forgotten. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information associated with the cell, thus solving the long range dependence problem, as shown in figure 6(a). Since one of the key features of $1/f$ noise is its long range dependence, $1/f$ noise might play a special role in suppressing the overfitting in an LSTM for time serial data tasks such as weather prediction.

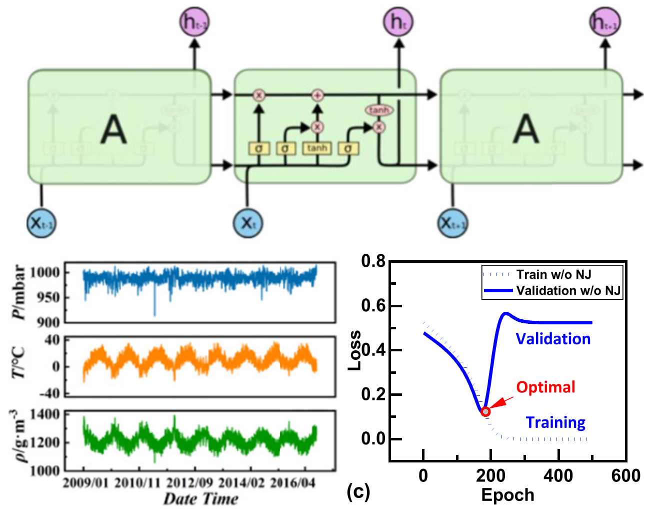

Figure 6. (a) Schematic of LSTM structure. A cell contains an input gate, an output gate and a forget gate. (b) Visualization of the selected dataset (temperature, pressure and air density) used to train the LSTM. (c) Demonstration of overfitting, in a LSTM.

In this work, the climate time-series dataset recorded by the Max Planck Institute for Biogeochemistry is used to train the LSTM. The entire dataset consists of 14 features, which were recorded once per 10 minutes at the Weather Station of Max Planck Institute for Biogeochemistry in Jena, Germany. Three features out of the 14 in the dataset, i.e. temperature, pressure and air density, are selected for a quick demonstration, as visualized in figure 6(b). The data of those three features are recorded once per 10 minutes, and is collectively defined as the data of 'one moment'. In other words, each moment contains three data points: temperature, pressure and air density. Ten consecutive moments are used for training and 500 consecutive moments for validation. Specifically, the first ten moments for training and moments from the 20000th to the 20500th for validation, to avoid overlap. Since every feature has values with varying ranges, normalization is carried out to confine feature values to a range of $[0, 1]$ before training the LSTM, by subtracting the minimum and dividing by the difference between the maximum and minimum of each feature. During training and validation, the batch size is 7 and 497 respectively. Therefore, the 8th, 9th, and 10th moments of training will be used as training labels while the 498th, 499th and 500th moments of validation will be used as validation labels. The optimizer is Adam and the learning rate is 0.0001. The loss function is the mean square error loss. During training, a simulated time-domain $1/f$ noise is applied onto the hidden state (ht) for overfitting suppression. The amplitude of noise is calculated according to (equation (1)) where the SNR refers to a fixed signal-to-noise ratio (SNR) and the signal is the hidden state value updated in each epoch. Gaussian noise is injected in the same way and of the same SNR for comparison.

Without using noise injection, overfitting can be clearly observed in figure 6(c). After training starts, both the training loss and validation loss decreases (underfit), until at ∼180th epoch when the training loss keeps decreasing but the validation loss reaches its lowest point (optimal) and started to sharply increase. After that, the training loss keeps decreasing while the validation loss keeps increasing. After ∼250th epoch, the training and validation loss finally saturate at around zero and 0.5, respectively.

$1/f$ noise and Gaussian noise with SNR $= 0$ dB is injected onto this LSTM for an initial demonstration (figures 7(b) and (c)). It can be clearly observed that the loss is controlled at the optimal point between the 280th and 470th epoch, while for the Gaussian noise injection, the loss function reaches the lowest point at the 270th epoch, sharply increases to a peak around 0.38 at the 380th epoch, and then gradually decrease afterwards. This supports that for the LSTM, $1/f$ noise and Gaussian noise could both suppress overfitting, but with different performance. Compared with the Gaussian noise case where although the loss is lower than the 'without noise injection' case, the loss fluctuates and could be as high as ∼0.38, the $1/f$ noise could effectively fix the loss at around the optimal point for around 200 epochs. The impact of SNR on loss is further demonstrated in figure 7(d), showing that the optimal SNR for overfitting suppression is -1 dB.

The $1/f$ noise shows ∼100% stronger overfitting suppression effect compared with the Gaussian noise: $1/f$ noise lowers the loss by 0.4 (from ∼0.5 to ∼0.1, which is the optimal level in figure 7(b)), while Gaussian noise only lowers the loss by 0.2 (from ∼0.5 to ∼0.3), as shown in figure 7(c). This is very different from the MLP and showing strong indication that there might be some 'coupling effect' that enhance the overfitting suppression effect of the long-term memory of $1/f$ noise on the LSTM. At lower SNR below 10 dB, $1/f$ noise shows stronger capability to suppress overfitting compared with Gaussian noise (figure 7(d)). For SNR below

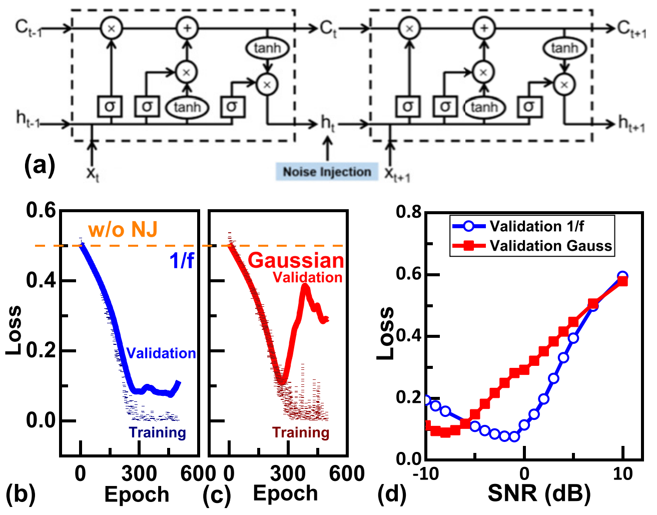

Figure 7. (a) Schematic of noise injection in the LSTM structure. Noise is applied onto the hidden state (ht) in each epoch. (b) and (c) Comparison of overfitting suppression, using (b) $1/f$ noise and (c) Gaussian noise, with SNR fixed at 0 dB. (d) Dependence of loss on the noise SNR.

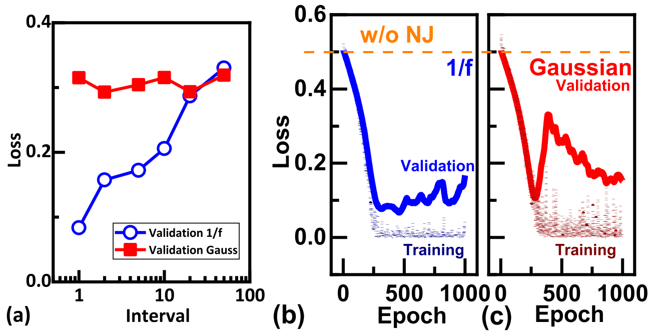

Figure 8. (a) Overfitting suppression effect using $1/f$ noise injected with different interval, or sampling frequency. (b) and (c) Comparison of overfitting suppression during 1000 epochs, using (b) $1/f$ noise and (c) Gaussian noise, with SNR fixed at -1 dB.

-5 dB, the noise will be far larger than the signal, and the network will be practically learning the noise instead of the signal.

To evaluate if the memory ability of LSTM really makes a difference, the $1/f$ noise is sampled at a fixed interval length to mimic the training time per epoch (default interval = 1). For comparison, a zero-mean Gaussian noise is also simulated and added.

This assumption is further confirmed in figure 8(a) when the noises are sampled at different intervals, i.e. using different sampling frequencies. For the $1/f$ noise, the loss is dependent in logscale on the sampling frequency, while the loss is almost not dependent on the sampling frequency of the Gaussian noise. If the interval is larger than 50, the loss of $1/f$ noise and Gaussian noise becomes similar. This is strong evidence, that the autocorrelation of $1/f$ noise plays an important role in the overfitting suppression of LSTM. Considering that LSTM's sequential structure. We can predict that the autocorrelation of $1/f$ noise could make it a special solution for the overfitting suppression in LSTM. However, for the Gaussian noise, which is memory-less, does not have such benefit.

Figure 8(b) compares the loss during 1000 epochs using the optimal SNR of -1 dB. For the $1/f$ noise injection, the loss function remains at the lowest point during the 280th and 470th epoch, and starts to gradually increase afterwards. For the Gaussian noise injection, the loss function reaches the lowest point at the 270th epoch, sharply increases to a peak at 380th epoch, and then gradually decrease afterwards. Although the $1/f$ noise injection does not eliminate totally the overfitting phenomenon, it still shows significant effect in suppressing overfitting and keeping the LSTM at the optimal condition for 190 epochs (from the 280th to the 470th), much better than the Gaussian noise.

The different performance between $1/f$ noise and Gaussian noise can be explained by using the 'xcorr' function in Matlab to calculate the ACF of $1/f$ noise and Gaussian noise [55], as shown in figure 9. The $1/f$ noise and Gaussian noise are sampled in the way as figure 5(a), with various intervals. Obviously, as the lag increases,

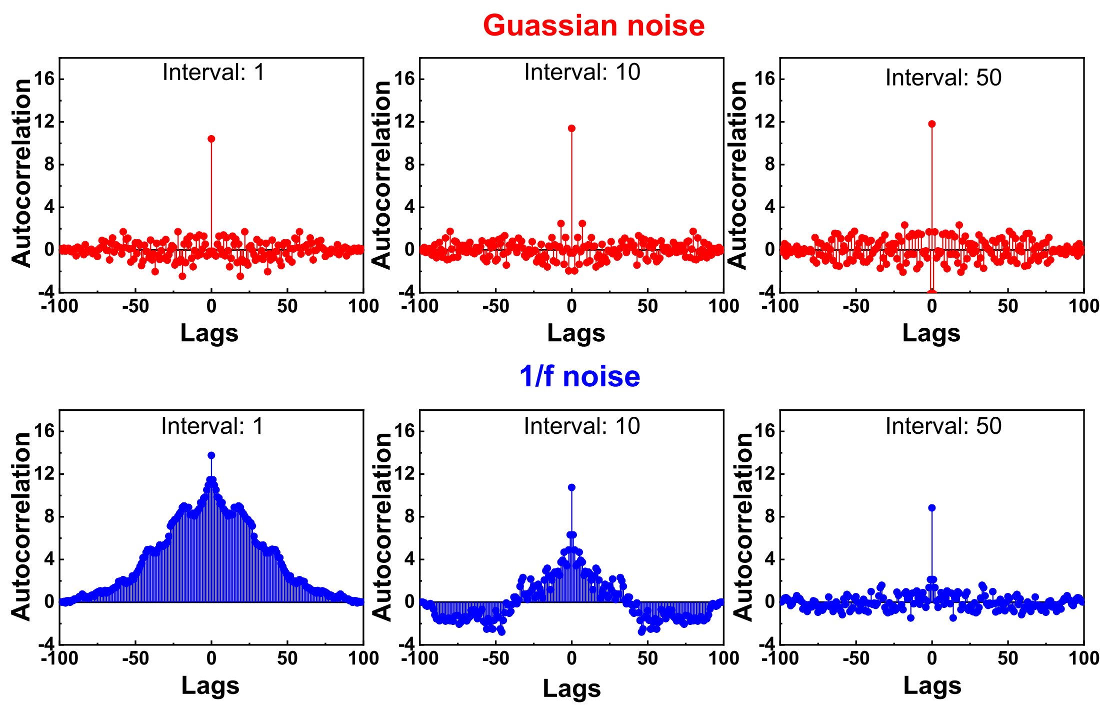

Figure 9. ACF of $1/f$ noise and Gaussian noise after being sampled with different intervals.

the ACF of $1/f$ noise decays gradually, supporting that $1/f$ has long memory with autocorrelation. However, for Gaussian noise, the ACF is almost independent from the lag, indicating that it does not have memory, or long range dependence. It should also be noted that if $1/f$ noise is sampled using very large interval, say 50, its ACF seems also independent from the lag, which is a strong indication that the long range dependence has decayed and now the sampled $1/f$ noise is very similar to the Gaussian noise, which also lead to similar effect in figure 8(a).

The aim of this paper is to give a preliminary demonstration that physics properties of materials, such as $1/f$ noise, shows some advantage over the software based approach, in the development of HNNs based on semiconductor devices such as transistors and memristors where implementing the software-based noise injection could be difficult: complex peripheral circuitry need to be designed to generate and modulate the Gaussian noise. For example, a conventional additive white Gaussian noise is to use a Zener diode in a reversed-biased circuit to produce Gaussian noise. This will bring additional area and power consumption to the HNN.

On the other hand, the semiconductor devices that form the HNN are naturally great sources of different noises, such as the thermal noise originated from the thermal agitation of the charge carriers, shot noise originated from the discrete nature of electric charge, random telegraph noise from the trapping of carriers, and $1/f$ noise originated from the carrier number fluctuation, in addition to the Gaussian noise. This is a significant advantage, as noise can be conveniently obtained from the semiconductor devices that form the HNN, without using the Zener diode or other devices/circuits for noise generation. Furthermore, another advantage is that those noises have different varies characteristics, such as the long-term memory/dependence of $1/f$ noise, which provides even better overfitting results, if they are properly selected and modulated, for some special architectures such as the LSTM.

5. Conclusions

In this work, we proposed the idea of using $1/f$ noise injection to suppress overfitting in different neural networks, and demonstrated that: (i) $1/f$ noise could suppress the overfitting in MLP and LSTM; (ii) $1/f$ noise and Gaussian noise performs similarly for the MLP but differently for the LSTM; (iii) the superior performance of $1/f$ noise on LSTM can be attributed to its intrinsic long range dependence. This work reveals that $1/f$ noise, which is common in semiconductor devices, can be a useful solution to suppress the overfitting in HNNs. This work could also provide strong support that the imperfectness of semiconductor devices can be exploited to provide solutions for the development of hardware AI technologies, mimicking the human brains which are not always precise but have been used efficiently and accurately for millions of years.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 62104188), the Major Key Project of PCL (Grant No. PCL2021A12) and the Engineering and Physical Sciences Research Council (EPSRC) of U.K. (Grant Nos. EP/M006727/1 and EP/S000259/1).

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Conflict of interest

The authors declare no conflict of interest.

ORCID iDs

Zheng Chai https://orcid.org/0000-0003-3446-7138
Tao Li https://orcid.org/0000-0002-3337-6202

References

[1] LeCun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521** 436–44
[2] Shao W, Salim F D, Song A and Bouguettaya A 2016 Clustering big spatiotemporal-interval data *IEEE Trans. Big Data* **2** 190–203
[3] Anthony M and Bartlett P L 2009 *Neural Network Learning: Theoretical Foundations* (Cambridge: Cambridge University Press)
[4] Shao W, Zhang Y, Guo B, Qin K, Chan J and Salim F D 2019 Parking availability prediction with long short term memory model *Green, Pervasive, and Cloud Computing* vol 11204 (Berlin: Springer) pp 124–37
[5] Ren H, Shao W, Li Y, Salim F D and Gu M 2020 Three-dimensional vectorial holography based on machine learning inverse design *Sci. Adv.* **6** eaaz4261
[6] Han S, Liu X, Mao H, Pu J, Pedram A, Horowitz M A and Dally W J 2016 EIE: efficient inference engine on compressed deep neural network *ACM SIGARCH Comput. Architect. News* **44** 243–54
[7] Shao W, Nguyen T, Qin K, Youssef M and Salim F D 2018 BLEDoorGuard: a device-free person identification framework using bluetooth signals for door access *IEEE Internet Things J.* **5** 5227–39
[8] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 Dropout: a simple way to prevent neural networks from overfitting *J. Mach. Learn. Res.* **15** 1929–58
[9] Lever J, Krzywinski M and Altman N 2016 Model selection and overfitting *Nat. Methods* **13** 703–4
[10] Samuel Y, Irene G, Matt F and Somesh J 2018 Privacy risk in machine learning: analyzing the connection to overfitting *2018 IEEE 31st Computer Security Foundations Symp.* pp 268–82
[11] Bartlett P L, Long P M, Lugosi G and Tsigler A 2020 Benign overfitting in linear regression *Proc. Natl Acad. Sci. USA* **117** 30063–70
[12] Carremans B 2018 Handling overfitting in deep learning models https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e
[13] IBM Cloud Education 2021 Underfitting https://ibm.com/cloud/learn/underfitting
[14] Shao W, Chan J and Salim F D 2020 Approximating optimisation solutions for the travelling officer problem with neural networks *Int. Joint Conf. Neural Networks*
[15] Molchanov D, Ashukha A and Vetrov D 2017 Variational dropout sparsifies deep neural networks *Proc. 34th Int. Conf. Machine Learning*
[16] Noh H, You T, Mun J and Han B 2017 Regularizing deep neural networks by noise: its interpretation and optimization *31st Conf. Neural Information Processing Systems*
[17] Matsuoka K 1992 Noise injection into inputs in back-propagation learning *IEEE Trans. Syst. Man Cybern.* **22** 436–40
[18] Brownlee J 2018 How to improve deep learning model robustness by adding noise https://machinelearningmastery.com/how-to-improve-deep-learning-model-robustness-by-adding-noise/
[19] Eigenmann R and Lilja D J 1999 Von Neumann computers *Encyclopedia of Electrical and Electronics Engineering* (Hoboken, NJ: Wiley) pp 384–400
[20] Drubach D 2000 *The Brain Explained* (Englewood Cliffs, NJ: Prentice-Hall)
[21] Fujitsu 2021 Supercomputer Fugaku https://fujitsu.com/global/about/innovation/fugaku/
[22] Chai Z, Freitas P, Zhang W, Hatem F, Zhang J F, Marsland J, Govoreanu B, Goux L and Kar G S 2018 Impact of RTN on pattern recognition accuracy of RRAM-based synaptic neural network *IEEE Electron Device Lett.* **39** 1652–5
[23] Du Y, Jing L, Fang H, Chen H, Cai Y, Wang R, Zhang J and Ji Z 2020 Exploring the impact of random telegraph noise-induced accuracy loss on resistive RAM-based deep neural network *IEEE Trans. Electron Devices* **67** 3335–40
[24] Kang J *et al* 2017 Time-dependent variability in RRAM-based analog neuromorphic system for pattern recognition *IEEE Int. Electron Devices Meeting*
[25] Huang H M, Xiao Y, Yang R, Yu Y T, He H K, Wang Z and Guo X 2020 Implementation of dropout neuronal units based on stochastic memristive devices in neural networks with high classification accuracy *Adv. Sci.* **7** 2001842
[26] Lu J, Wu Z, Zhang X, Wei J, Fang Y, Shi T, Liu Q, Wu F and Liu M 2020 Quantitatively evaluating the effect of read noise in memristive Hopfield network on solving traveling salesman problem *IEEE Electron Device Lett.* **41** 1688–91
[27] Cai F *et al* 2020 Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks *Nat. Electron.* **3** 409–18
[28] Yacoub M D 2019 *Foundations of Mobile Radio Engineering* (Oxfordshire: Routledge)

[29] Chen C-H 2021 Thermal noise measurement and characterization for modern semiconductor devices *IEEE Instrum. Meas. Mag.* **24** 60–71

[30] Claeys C, Andrade M G C D, Chai Z, Fang W, Govoreanu B, Kaczer B, Zhang W and Simoen E 2016 Random telegraph signal noise in advanced high performance and memory devices *2016 31st Symp. Microelectronics Technology and Devices (SBMicro)* pp 1–6

[31] Fang W *et al* 2016 Impact of the effective work function gate metal on the low-frequency noise of gate-all-around silicon-on-insulator NWFETs *IEEE Electron Device Lett.* **37** 363–5

[32] Bonani F and Ghione G 2001 *Noise in Semiconductor Devices* (Berlin: Springer)

[33] Konczakowska B M W A 2011 *Noise in Semiconductor Devices* (Boca Raton, FL: CRC Press)

[34] Bak P, Tang C and Wiesenfeld K 1987 Self-organized criticality: an explanation of the $1/f$ noise *Phys. Rev. Lett.* **59** 381

[35] Hooge F N 1994 $1/f$ noise sources *IEEE Trans. Electron Devices* **41** 1926–35

[36] Bloom I and Nemirovsky Y 1991 $1/f$ noise reduction of metal-oxide-semiconductor transistors by cycling from inversion to accumulation *Appl. Phys. Lett.* **58** 1664

[37] Bae S-H, Lee J-H, Kwon H-I, Ahn J-R, Om J-C, Park C H and Lee J-H 2009 The $1/f$ noise and random telegraph noise characteristics in floating-gate nand flash memories *IEEE Trans. Electron Devices* **56** 1624–30

[38] Amara-Dababi S, Sousa R C, Béa H, Baraduc C, Mackay K and Dieny B 2014 Breakdown mechanisms in MgO based magnetic tunnel junctions and correlation with low frequency noise *2014 IEEE Int. Reliability Physics Symp.* 6A.1.pp 1A.1.7–6

[39] Ambrogio S, Balatti S, McCaffrey V, Wang D C and Ielmini D 2015 Noise-induced resistance broadening in resistive switching memory—part: I. Intrinsic cell behavior *IEEE Trans. Electron Devices* **62** 3805–11

[40] Betti Beneventi G, Ferro M and Fantini P 2012 $1/f$ noise in 45 nm RESET-state phase-change memory devices: characterization, impact on memory readout operation, and scaling perspectives *IEEE Electron Device Lett.* **33** 1559–61

[41] Kleinpenning T G M 1985 $1/f$ noise in p–n junction diodes *J. Vac. Sci. Technol.* A **3** 176–82

[42] Guan J, Guo S, Wang J, Tao M, Cao J and Gao F 2016 Analysis of origin of measured $1/f$ noise in high-power semiconductor laser diodes far below threshold current *Microelectron. Reliab.* **59** 55–9

[43] Reinker S 2004 Stochastic resonance in thalamic neurons and resonant neuron models *PhD Thesis* University of British Columbia

[44] Buzsáki G 2006 *Rhythms of the Brain* (Oxford: Oxford University Press)

[45] Linkenkaer-Hansen K, Nikouline V V, Palva J M and Ilmoniemi R J 2001 Long-range temporal correlations and scaling behavior in human brain oscillations *J. Neurosci.* **21** 1370–7

[46] Allegrini P, Menicucci D, Bedini R, Fronzoni L, Gemignani A, Grigolini P, West B J and Paradisi P 2009 Spontaneous brain activity as a source of ideal *Phys. Rev.* E **80** 061914

[47] Wagenmakers E-J, Farrell S and Ratcliff R 2004 Estimation and interpretation of $1/f\alpha$ noise in human cognition *Psychon. Bull. Rev.* **11** 579–615

[48] Amirkhanov R N, Ghots S S and Bakhtizin R Z 1996 Autocorrelation function of $1/f$ current fluctuations in vacuum microelectronics devices *J. Vac. Sci. Technol.* B **14** 2135–7

[49] Stadnitski T 2012 Measuring fractality *Front. Physiol.* **3** 1–13

[50] MathWorks 2022 Pinknoise https://ww2.mathworks.cn/help/audio/ref/pinknoise.html

[51] MathWorks 2022 Randn https://ww2.mathworks.cn/help/matlab/ref/randn.html

[52] Simplilearn 2022 An overview on multilayer perceptron (MLP) https://simplilearn.com/tutorials/deep-learning-tutorial/multilayer-perceptron

[53] LeCun Y and Cortes C 1998 The MNIST database of handwritten digits http://yann.lecun.com/exdb/mnist/

[54] Tetko I V 2002 Associative neural network *Neural Process. Lett.* **16** 187–99

[55] MathWorks 2022 Xcorr https://ww2.mathworks.cn/help/matlab/ref/xcorr.html