The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

A Plasticity-Centric Approach to Train the Non-Differential Spiking Neural Networks

Tielin Zhang,[1,2] Yi Zeng,[1,2,3] Dongcheng Zhao,[1,3] Mengting Shi[1,3]
[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, China
[3]University of Chinese Academy of Sciences, Beijing, China
{tielin.zhang, yi.zeng}@ia.ac.cn

Abstract

Many efforts have been taken to train spiking neural networks (SNNs), but most of them still need improvements due to the discontinuous and non-differential characteristics of SNNs. While the mammalian brains solve these kinds of problems by integrating a series of biological plasticity learning rules. In this paper, we will focus on two biological plausible methodologies and try to solve these catastrophic training problems in SNNs. Firstly, the biological neural network will try to keep a balance between inputs and outputs on both the neuron and the network levels. Secondly, the biological synaptic weights will be passively updated by the changes of the membrane potentials of the neighbour-hood neurons, and the plasticity of synapses will not propagate back to other previous layers. With these biological inspirations, we propose Voltage-driven Plasticity-centric SNN (VPSNN), which includes four steps, namely: feed forward inference, unsupervised equilibrium state learning, supervised last layer learning and passively updating synaptic weights based on spike-timing dependent plasticity (STDP). Finally we get the accuracy of 98.52% on the hand-written digits classification task on MNIST. In addition, with the help of a visualization tool, we try to analyze the black box of SNN and get better understanding of what benefits have been acquired by the proposed method.[1]

Introduction

Neural network models are essential for developing Artificial Intelligence, since they are inspired by the brain at multiple levels of details and practically useful in many application areas. Deep neural networks (DNNs) and Spiking Neural Networks(SNNs) have attracted many attentions for different reasons. DNNs have shown their success on visual and audio recognition, natural language processing (LeCun, Bengio, and Hinton 2015), even on decision oriented learning tasks such as Atair 2600 games (Mnih et al. 2015) and the game Go (Silver et al. 2016). In different types of DNNs, the feed-forward type (e.g. convolutional DNN) is designed mainly for the abstraction of spatial information, and the recurrent type (e.g. long-short term memory DNN) is designed mainly for the process of temporal information. While SNNs are more biologically plausible, hence more natural to interpret the information processing mechanisms of the brain, and are relatively energy efficient.

The underlying theories of DNNs are considered to be inspired by the brain from various perspectives. The convolutional neural network is considered to be inspired by the visual cortical processing of mammalian brain (LeCun, Bengio, and Hinton 2015). The recurrent neural network learns from the Hopfield network which gets the inspiration from the hippocampus CA3 (Zhang et al. 2016). Many other versions of DNNs are also powered by different brain mechanisms, such as attention (Xu et al. 2015; Zhang, Zeng, and Xu 2016), memory (Sukhbaatar et al. 2015; Zhang et al. 2016), equilibrium states (Scellier and Bengio 2017) which have shown their effectiveness on many specific tasks.

The principles of neurons, synapses and networks in biological systems are more complex than these in DNNs (Hassabis et al. 2017; Helmstaedter 2015). For example, it has been proved that the function of a single biological neuron with complex dendritic branches needs a three-layered artificial neural network which is composed of hundreds of hidden-layer neurons to simulate (Hausser and Mel 2003). From this perspective, one possible way to improve DNNs is turning to biological neural networks.

The spiking neural networks are considered to be the third generation of neural networks which are more powerful on the processing of both spatial and temporal information (Maass 1997). Firstly, the neuron models are more biologically realistic, and the neuron will not be activated until the membrane potential reach the threshold (e.g. the leaky-integrated and fire model (LIF) makes SNNs more energy efficient). Secondly, the membrane potential will be nonlinear leaky or increased with different inputs, which makes SNN good at processing nonlinear information. Thirdly, the synaptic weights between neurons are time-sensitive and will be changed according to the relative arriving time of pre- and post-synaptic neuron spikes, which makes SNN easier to process both spatial and temporal information. Fourthly, different computing costs in neurons and synapses cause various kinds of time delays which will contribute to the asynchronous computation of SNN.

Due to the relevance with the biological brain from the

[1]Tielin Zhang and Yi Zeng contributed to this paper equally and should be regarded as co-first authors. Dongcheng Zhao and Mengting Shi contributed to this paper equally.

mechanism perspective, ideally SNNs should be powerful and efficient. However, since the nature of spiking neurons is non-linear and discontinuous, they are difficult to be tuned by current back propagation methodology. Many efforts have been taken to deal with this catastrophic training problem.

Diehl et al. trained an artificial neural network (ANN) with the limitation of minor range of weights and no mathematically additional biases. This effort limits synaptic weights in a linear range so that the ANN could be converted directly into a linear SNN classifier (Diehl et al. 2015). Even though this method could achieve the performance of 98.48% on 10-class hand-written digit classification task MNIST (i.e. Modified National Institute of Standards and Technology), the performance of SNN is actually contributed by ANN from back propagation instead of pure SNN learning. Similar to Diehl et al., Lee et al. think the discontinuities between spikes could be considered as noises, and the SNN without noises are continuous which could use back propagation for training. Lee et al. tested the idea on MNIST dataset and got 98.64% on classification accuracy (Lee, Delbruck, and Pfeiffer 2016).

SpikeProp is another effort which makes the equivalent exchange from spatial information (e.g. fire rates) in ANN to the temporal information (e.g. timing of inter-spike intervals) in SNN. This effort successfully converts the spatial back propagation in ANN into the temporal back propagation in SNN (Bohte, Kok, and La Poutre 2002). However, the problem of non-differential characteristics of SNN still exists, which has limited further improvements of SNNs on performance of both accuracy and efficiency.

Instead of tuning back propagation of DNN to fit for SNN, some challenging efforts try to directly use biologically plausible rules to train SNNs.

Zeng et al. (Zeng, Zhang, and Xu 2017) have proposed seven biologically plausible rules to train multi-layer SNN with LIF neurons. It has been proved that the synaptic weights in first few layers could be dynamically updated by STDP rules without any supervision, and the weights between the final two layers could be supervised and learned by Hebb's law. Diehl et al. have also trained a standard SNN by unsupervised STDP learning rule, but that effort could only train a two layers SNN and the performance is 95% on MNIST (Diehl and Cook 2015).

In this paper, we will firstly introduce the characteristics of spatial and temporal information processing in SNN, and then introduce an efficient Voltage-driven Plasticity-centric methodology to train non-differential SNN with LIF neurons. The proposed model contains four parts, namely: feed forward inference, unsupervised equilibrium state learning, supervised last layer learning and passively updating synaptic weights based on STDP. The first two parts are unsupervised, in which the membrane potential of neurons in each layer will be updated by feed forward streams and equilibrium conditions. The third part is supervised, in which the one-dimensional gradient descent in last two layers is calculated with the support of teaching signals. The fourth part is unsupervised, in which the synaptic weights are passively updated by STDP rule. Experimental results show that the model could get 98.52% on classification accuracy on MNIST dataset.

The Architecture of SNN

The complexity of SNN architecture is from its different scales. On the neuron scale, various details we considered for the construction of a single excitatory or inhibitory neuron will contribute different dynamics to SNNs. On the network scale, different proportions of connection types (e.g. recurrent or feed forward) will also affect the network characteristics. In this paper, we focus on the biological LIF neuron model and the multi-layer feed-forward SNN architecture.

Basic LIF Neuron

With the biological plausible characteristics of discontinuous membrane potential, the biologically inspired LIF model has been widely used in many computational neuroscience tasks. The LIF membrane potential will be dynamically increased when excitatory presynaptic spikes arrive, and the neuron will fire once the membrane potential exceeds the firing threshold, after that the neuron will turn back to the resting state of membrane potential until the end of refractory time, as shown in Fig 1.

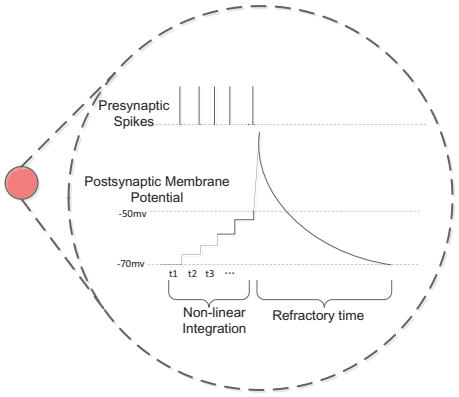

Figure 1: The LIF neuron with discontinuous spikes

$$C_m \frac{dV}{dt} = -g_L (V - V_L) + I_{syn} \tag{1}$$

The dynamical function of membrane potential in LIF is shown in Equation (1). The C_m is the membrane capacitance, the g_L is the leaky conductance, V_L is the leaky potential, and I_{syn} is the input stimulus from presynaptic neuron. Assume that the total excitatory conductance is g_E, the reversal potential is V_E, and make $\tau_m = \frac{C_m}{g_L}$, then the Equation (1) could be converted to Equation (2),

$$\tau_m \frac{dV}{dt} = -(V - V_L) - \frac{g_E}{g_L}(V - V_E) \tag{2}$$

$$\tau_E \frac{dg_E}{dt} = -g_E + \eta \sum_{j \in N_E} w_{j,i} \delta_t \tag{3}$$

The g_E in Equation (2) will be dynamically changed, as Equation (3) shows, whenever the pre-synaptic neurons fire, the g_E will be non-linearly increased by the number of input spikes. δ_t is the pre-synaptic inputs (in the next step it will be updated into non-differential potential V_j in Equation (5)), η is the learning rate, τ_E is the conductance decay of excitatory neurons, and $w_(j,i)$ is the connection weight from pre-synaptic neuron j to the target neuron i.

$$if\,(V > V_{th})\begin{cases} V = V_L \\ T_{ref} = T_0 \end{cases} \tag{4}$$

The increased membrane potential V will be reset to resting potential after firing, as shown in Equation (4), and the refractory time T_{ref} will be extended to a larger T_0.

The Multi-layer SNN

As shown in Fig 2, a multi-layer feed-forward SNN (for simplicity here we use three-layer SNN) could be separated into three parts: the first part is the input layer with sequential spikes input; the second part is hidden layers with the ability of transformation of non-linear and discontinuous features; the third part is the output layer for the signal selection with the help of teacher signal.

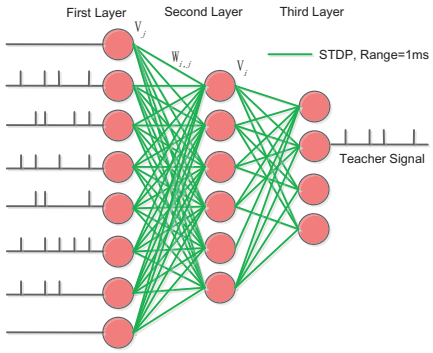

Figure 2: The architecture of an SNN

The Learning Methodology of SNN

Since the spikes and membrane potentials in SNN are usually non-linear and discontinuous, directly tuning their values by artificial back propagation will be challenging. Inspired by the tuning methodology in biological networks, in which the update of synaptic weights is passively changed by the temporal activities of connected neurons (e.g. pre- and post-synaptic neuron spikes), here we change the view of synaptic learning into the view of membrane potential learning. From this perspective, a new four-step SNN learning algorithm is proposed to deal with the catastrophic training problem in SNN. In the first two steps, the unsupervised membrane potential learning will be applied. In the third step, the neuron membrane potential in last two layers will be updated by teaching signals. In the fourth step, the STDP (or differential $Hebb's$ law) is used to passively update the synapses between dynamical neurons.

Feed Forward Inference in SNN

The input spikes will flow and pass three layers respectively. In each neighborhood layers, the information will convergent from pre-synaptic neurons into the post-synaptic neurons. As the first two layers for example, we set the states of pre-synaptic neurons in the first layer as V_j, and the states of post-synaptic neuron in the second layer as V_i.

$$\begin{cases} \tau_m \frac{dV_i}{dt} = -(V_i - V_L) - \frac{g_E}{g_L}(V_i - V_E) \\ \tau_E \frac{dg_E}{dt} = -g_E + \eta \sum_j^N w_{j,i} V_j \\ V_i = V_L, T_{ref} = T_0 \qquad if\,(V_i > V_{th}) \\ V_i^{FF} = V_i \end{cases} \tag{5}$$

As shown in Equation (5), the inputs will have feed forward transmission to the next layer and dynamically change the states of the neurons (i.e. membrane potential V_i^{FF}). Then the information will go on until it arrives at the output layer. In the training procedure, the propagation of feed forward information will integrate with equilibrium state learning and supervised learning as three main motivations of dynamically changing of neuron membrane potentials. While after training, only rule one will be left for the sequential inference tests.

Unsupervised Hierarchical Equilibrium State Learning in SNN

Actually there are two kinds of equilibrium states in SNN. For temporal equilibrium state, the membrane potentials will be attracted into equilibrium states as time goes on, for example the V_E in Equation (5). Another is spatial equilibrium state learning which describes the equilibrium of the input and output of the neurons. In the training procedure, the network will converge repeatedly until it stops at a static point, which means that the total updates of weights will be smaller and smaller as the training time going on, and finally the network will stop at equilibrium membrane potential states. The unsupervised equilibrium state learning will make the network ready for classification tasks in next two steps. The analysis of the role of it on network learning will be introduced in the next Section.

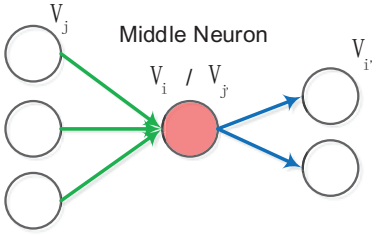

Figure 3: The neuron with equilibrium state in feed forward SNN

Fig 3 shows the building block of a feed forward SNN. We set the pre-synaptic neurons as neuron j and post-synaptic neuron as neuron i. The equilibrium state of neuron i (i.e. E_i) could be set as:

$$\Delta E_i = V_i - \left(\sum_j^N w_{j,i} V_j - V_{th,i} \right) \tag{6}$$

Here $\Delta E_i = \frac{dE_i}{dt}$ is defined as the differential equilibrium state. The first term after equal is the current membrane potential of neuron i. The second term is the future membrane potential of neuron i which integrates all of the inputs from pre-synaptic V_j. The network will learn dynamically towards network convergence which also means the equilibrium state for each neuron. With training time going by, the current states and next states of neurons will become equivalent, which means the ΔE_i will be around zero. Considering that the membrane potential V_i has already take the place of $w_(j,i)$ on network tuning, we update Equation (6) into Equation (7) according to the differential chain rule.

$$\frac{dE_i}{dV_i} = \frac{dE_i}{dt} \times \frac{dt}{dV_i} = \frac{V_i - \left(\sum_j^N w_{j,i} V_j - V_{th,i} \right)}{\frac{dV_i}{dt}} \tag{7}$$

Then we add non-linear Equation (2) into Equation (7), and we could get Equation (8).

$$\frac{dE_i}{dV_i} = \frac{V_i - \left(\sum_j^N w_{j,i} V_j - V_{th,i} \right)}{-(V_i - V_L) - \frac{g_E}{g_L}(V_i - V_E)} \tau_m \tag{8}$$

Here we use E_i to represent the equilibrium state of each single neuron. From Equation (8), the update of neuron membrane potential will decrease the whole network equilibrium state $E = \sum_i^N E_i$, and we could consider this state as one kind of the energy in SNN.

$$E_i = \int_{V_{min}}^{V_{max}} \frac{V_i - \left(\sum_j^N w_{j,i} V_j - \sum_j^N V_{th,i} \right)}{-(V_i - V_L) - \frac{g_E}{g_L}(V_i - V_E)} \tau_m dV_i \tag{9}$$

The continuous Equation (9) could be approximately represented by the discrete Equation (10) through Euler method. The V_i^{ES} is the updates of membrane potential on V_i in equilibrium states part. And the η_i is the learning rate which contains the variable τ_m.

$$\Delta V_i^{ES} = -\eta_i \frac{V_i - \left(\sum_j^N w_{j,i} V_j - \sum_j^N V_{th,i} \right)}{-(V_i - V_L) - \frac{g_E}{g_L}(V_i - V_E)} \tag{10}$$

Finally, we could get rules to update each neuron state V_i. With the integration of unsupervised feed-forward learning in Equation (5) and Equation (10), we could update the whole SNN membrane potential states V_i towards the convergence characteristics, as shown in Equation (11).

$$\Delta V_i = \frac{t}{T} \Delta V_i^{FF} + \left(1 - \frac{t}{T} \right) \Delta V_i^{ES} \tag{11}$$

The t is the training time slot and T is the total training time in SNN learning. ΔV_i^{FF} is the change of neuron states in feed forward step, while the ΔV_i^{ES} is the state change in equilibrium state step. The total ΔV_i will be the integration of these two steps, and the proportion of feed forward states will be increased while the equilibrium state is decreased with the training time pass by.

Supervised Learning in Last Two Layers in SNN

Both feed forward and equilibrium state procedures are unsupervised, and the network will converge into static states after unsupervised training. However, the network will not show any specific functions (e.g. recognition or classification) without the step of supervised learning. Here we add teaching signals into the last layer of SNN in the training procedure. The teaching signals is actually the fixed frequency stimulus, the intervals of them are the same with that of input signals. Since the error signals in the third layer could not back propagate to the first and second layers, it could be considered as a very weak supervised learning rule.

$$C = \sum_{i=1}^{L_3} (V_i - V_T)^2 \tag{12}$$

$$\frac{dC}{dV_i} = (V_i - V_T) \tag{13}$$

$$dV_i = -\eta^c (V_i - V_T) \tag{14}$$

As shown in Equation (12), the error of SNN is the integration of the divergence of realistic neuron state V_i with the teacher signal state V_T. η^c is the learning rate, L_3 is the number of neurons in the last layer. Different from the traditional ANN, in which the gradient descent will be calculated iteratively in each layer during the back propagation procedure, as shown in Equation (13) and Equation (14), we only use one time for differential calculation on final layer of SNN. This procedure could also be replaced by Hebb's law.

The Passive Update of Synaptic Weights based on STDP in SNN

There are various kinds of STDP rules (Dan and Poo 2004; Bi and Poo 2001). Take performance into consideration, we use a specific kind of STDP expression (Bengio et al. 2015a; 2015b) which describes an idea of STDP as presynaptic activity times rate of change of post-synaptic activity.

$$\Delta w_{j,i} \propto V_j V_i^{'} \tag{15}$$

$V_i^{'}$ is the derivative value of V_i. The reason why we use differential membrane potential STDP is that it could represent or process spikes or membrane potential states at the same time. With STDP rules the synaptic weights could be passively updated by the changes of the pre and post synaptic neuron states.

The Learning Architecture of SNN

The membrane potential states in SNN will be dynamically changed in the unsupervised and supervised training procedure, and then the weights of SNN will be passively updated based on the pre and post synaptic neuron states by STDP rules. The detailed procedure of training SNN is shown in Algorithm 1.

Algorithm 1 The Algorithm of SNN Learning.

1. Spatial and temporal data normalization and variables initiation. Initialize the multi-layer feed forward network, LIF neuron model, weight $w_{j,i}$ with random uniform distribution, and neuron membrane potential states V_i. Set iteration time I_{ite}, simulation time T, differential time Δt, learning rate η and η^c;
2. Start Training procedure:
(1). Load training samples;
(2). Update V_i^{FF} by Equation (5) during the feed forward inference learning;
(3). Update V_i^{ES} by Equation (10) during the unsupervised equilibrium state learning;
(4). Integrate V_i^{FF} and V_i^{ES} by Equation (11);
(5). Update V_i by Equation (14) with supervision;
(6). Passively update synaptic weights $w_{j,i}$ by Equation (15).
(7). Iterative training from Step (2) to Step (6), save final $w_{j,i}$.
3. Start test procedure:
(1). Test the performance of trained SNN with only feed forward step based on saved $w_{j,i}$.
(2). Output the performance of test samples without cross validation;
4. End SNN learning.

The Experimental Results

The proposed SNN learning model is a new kind of non-differential network tuning method. Its potential will cover both the spatial and temporal information processing. Many related works have shown that the SNN could be finally convergent by the restrictions from biological inspired rules (Zeng, Zhang, and Xu 2017). Here we use MNIST dataset to test the functional performance of the SNN with the proposed tuning method.

The MNIST Task Dataset

The Modified National Institute of Standards and Technology (MNIST) dataset with ten classes of hand-written digits (from zero to nine) is used to test the performance of the proposed SNN algorithm. As shown in the left of Fig 4, the original MNIST data contains 60,000 images for training and 10,000 images for test. Normally, the MNIST dataset will be directly added into the model without any pre-precessing. However, in SNN architecture, the membrane potential of neurons are dynamically changed as time goes by, and the synapses are also updated with the specific differential time of pre and post neural spikes. The traditional spatial-based MNIST dataset is no longer fit for the SNN task.

As shown in the right of Fig 4, with the random subsampling of original MNIST dataset, we could get different sub-image of original images. In addition, we package these sub-images as image streams which are with spatial and temporal characteristics and could be the input of SNN.

The Training and Test of SNN

We firstly train the SNN with sub-image streams as shown in the right of Fig 4 in a three-layer network architecture. The neurons in the first, second, third layers are 784, N and 10 respectively, in which N is the number of hidden neurons which is tested from 50 to 1000, as shown in Figure 6. The training method is shown in Algorithm 1. Here we use standard 60,000 MNIST data to train and another 10,000 to test (no cross validation). In test procedure, only feed-forward procedure is kept (i.e. equilibrium state learning and supervised learning are blocked), the tested images are added into the SNN one by one and the output neuron id with the largest number of spikes is selected as the output class.

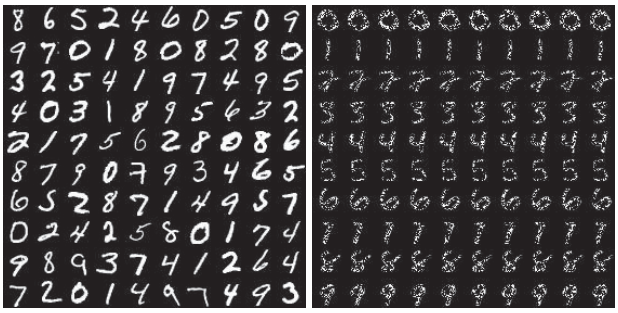

Figure 4: Original (left) and preprocessed (right) data

Figure 5: Samples for correctly (left) and incorrectly (right) classified images by SNN

After training and test on SNN, random samples of correct classification results on MNIST is shown in the left of Fig 5, from which we could observe that for most of the normal characters, SNN could recognize well. Further more, as shown in the right of Fig 5, incorrect results by SNN are also big challenges for human.

In addition, we also test the performance of three-layer SNN on different number of hidden neurons. We set iteration time as 100, the patch size as 10, and the learning rate as 0.05. The accuracies of SNNs on MNIST classification task are shown in Fig 6.

We observe from Fig 6 that: (1) the accuracy will generally be increased with more iteration times, and will be finally convergent and stop at a stable value; (2) the accuracies between different architectures are different, and usually the network with more hidden neurons will have better performance. So the accuracies of SNN will be dependent on both the specific iteration time and network structure. Here we will select the best accuracy value within iteration times as the performance of this kind of architectures.

In addition, for the different architectures, as shown in Fig 7, the best accuracy of them is 98.52% with 4,500 hidden neurons in the three-layer SNN architecture.

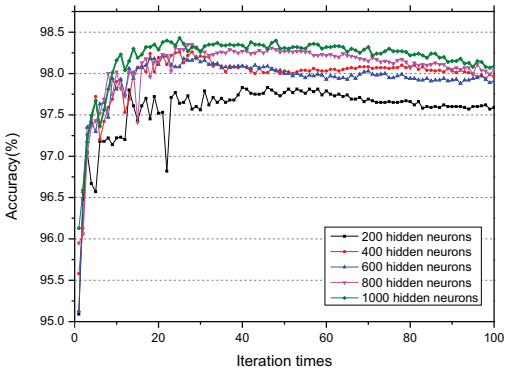

Figure 6: Test performance with different iteration times

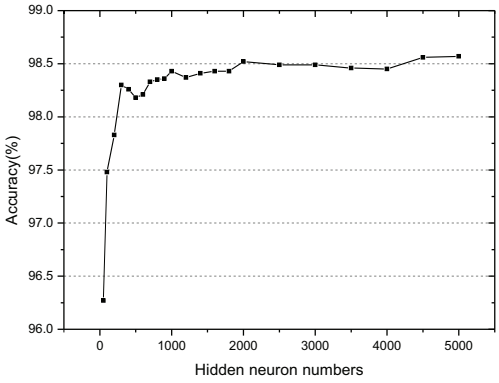

Figure 7: Test performance on different architectures

Comparative Studies with Other Models

Here we compare our model with other SNN models on the MNIST classification task, as shown in Table 1. Some of them use preprocessing strategies such as threshold judgment, training set enhancement, pre-edge detection, orientation detection, etc. For learning types, they are supervised or unsupervised. For neuron models, they are rate-based or spike-based. For learning methodologies, they are biologically inspired (e.g. morphology learning or STDP) or not (e.g. back propagation). From the table, we observe that the best performance of rate-based model with back propagation is 99.1%. However this method is actually converted from a traditional artificial neural network (not naturally spike based) with back propagation. Except for our model, the best performance of pure spike-based model with biological plausible learning rules (e.g. STDP) is 95%.

Our model is with both biological structures (LIF spiking neuron model, feed-forward architecture) and biological plausible learning rules (four step learning rules with STDP). Our method could reach the accuracy of 98.52% which outperforms 95%. To the best of our knowledge, this should be a new record on spike-based SNN with biological learning rules for MNIST task.

Analyzing the Functional Black Box of SNN

Deep analysis of connection weights of SNNs is a challenging task. However, some dimension-reduction method (e.g. SNE (Hinton and Roweis 2003), t-SNE (Maaten and Hinton 2008), LargeVis (Tang et al. 2016)) could decrease the dimension of the weight into two or three dimensions, by which human could far more easier to get better understanding. Here we use t-SNE to further analyze what happened in the procedure of SNN training, and could also answer that why the proposed model works on this kind of problem.

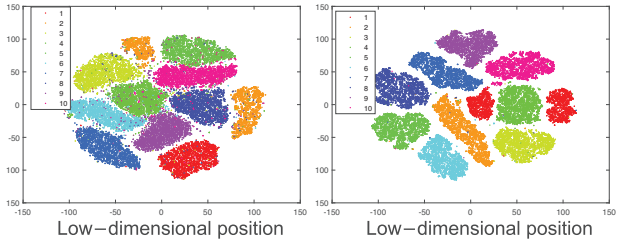

Figure 8: Visualization of input (left) and hidden (right) layers in SNN with t-SNE

The original dimension of MNIST dataset is 784 (which is the square of image width 28), and the number of images is 60,000 for training. In order to understand the distribution and relationships of these images, we directly use t-SNE to decrease the 784 dimensional data into the two dimensional data. As shown in the left of Fig 8, each point in figure stands for one 784 dimensional image sample, and different colors stand for different sample classes. The t-SNE will maintain the relationship of original datasets as much as possible. For example, any two points which are closer than other points in the 784-dimensional space would be similar in 2-dimensional space. Most but not all of the points in the left of Fig 8 could cluster into the correct classes. However, after the learning in SNN, the information in the second layer (with 400 hidden neurons) could have clearer and better clustering performance on t-SNE than the first layer, as shown in the right of Fig 8. This means the learning of synaptic weights in SNN is helping the network to perform better clustering and classification performances.

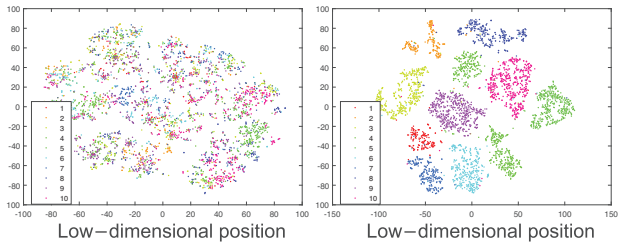

Figure 9: Unsupervised equilibrium state learning on hidden layers with 6 (left) and 20 (right) iteration times

In addition, Algorithm 1 shows the integration of unsupervised and supervised learning in SNN. However, the function of unsupervised procedure (especially the equilibrium

Table 1: Classification accuracy of SNNs on MNIST test set.

Architecture	Preprocessing	(Un-)Supervised	Training Type	Learning Rule	Performance
Dendritic neurons (Hussain, Liu, and Basu 2014)	Yes	Supervised	Rate-based	Morphology learning	90.3%
Spiking RBM (O'Connor et al. 2013)	Yes	Supervised	Rate-based	Contrastive divergence	94.1%
Convolutional SNN (Zhao et al. 2015)	Yes	Supervised	Spike-based	Temporal rule	91.3%
Spiking RBM (Merolla et al. 2011)	None	Supervised	Rate-based	Contrastive divergence	89.0%
Convolutional SNN (Diehl et al. 2015)	None	Supervised	Rate-based	Backpropagation	99.1%
Two-layer SNN (Diehl and Cook 2015)	None	Unsupervised	Spike-based	Exponential STDP	95%
Two-layer SNN (Querlioz et al. 2013)	None	Unsupervised	Spike-based	Rectangular STDP	93.5%
Voltage-driven Plasticity-centric SNN (Our work)	**None**	**Both**	**Spike-based**	**Equilibrium learning + STDP**	**98.52%**

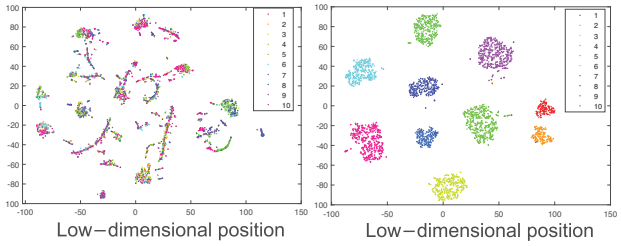

Figure 10: Unsupervised equilibrium state learning on output layers with 6 (left) and 20 (right) iteration times

state learning) and its role in network training is not very clear. How this procedure affect the classification task needs to be further explored.

Here we also use t-SNE to analyze the different proportions of unsupervised learning in SNN learning. We select different iteration times of unsupervised equilibrium state learning from 1 to 100, and in each iteration time we visualize the t-SNE images. As shown in Fig 9, with different repeated times of unsupervised learning, the SNN shows different degrees of clustering by t-SNE visualization method. In the left of Fig 9 the repeated time is 6 and the t-SNE result does not appear to be with clearly clustered phenomenon, however, with the increase of the repeated times of unsupervised learning, the cluster in the right of Fig 9 is clearer which shows the contribution of unsupervised learning will increase as the repeated time grows. Similar results could also be found in the output layers of SNN, as shown in Fig 10. This visualization result will help us better understand the functional role of the proposed model applied in SNN.

Conclusion

Although spiking neural networks are more biologically realistic to explore the nature of intelligence compared to other types of artificial neural networks. Understanding of their learning and training principles are still limited. In order to provide a more comprehensive framework, many different kinds of learning rules have been proposed (Zeng, Zhang, and Xu 2017; Zenke, Agnes, and Gerstner 2015), such as the different kinds of STDPs, long-term potentiation or depression, short-term facilitation or depression, hetero-synaptic plasticity, etc. Nevertheless, from a practical point of view, there is still a gap on network performance compared to deep neural networks.

This paper proposed the Voltage-driven Plasticity-centric SNN (VPSNN), which is a four-step learning model to create a new general learning architecture on the training of SNNs. It integrates supervised and unsupervised learning to train an SNN with non-differential neurons. The method overcomes the curse of direct relationship analysis between synaptic weights and output performance (e.g. energy function or cost function), which usually need smooth functions of neurons and networks. The architecture contains these main parts: Firstly, our method focuses on the dynamical change of membrane potential affected by feed-forward signals in LIF. Secondly, the model keeps the balance between the input and output of neurons and networks by tuning membrane potentials. Thirdly, in the supervised learning phase, the membrane potentials of the last two layers are dynamically changed. Finally, biological inspired STDP learning rule is applied on the SNN to change the synaptic weights.

In this architecture, LIF neuron model, bi-phasic STDP rule, the integration of supervised and unsupervised learning methodologies are synthesized together. This new learning methodology does not care about the characteristics of neurons which are non-differential or not, and also separates the unsupervised and supervised procedures. It is with potential on tuning other types of biologically inspired neural networks.

The good performance of the experimental result of the proposed model on MNIST makes a step forward to minimize the gap on accuracy between SNNs and DNNs. It shows the potential and efficiency of the architecture. In the future, with more biology inspired learning principles added into the network in a synthesized way, the potential and advantages of SNNs can be further extended and may be bigger than other types of artificial neural networks.

Acknowledgments

This study was funded by the Beijing Municipal Commission of Science and Technology (Z161100000216124).

References

Bengio, Y.; Lee, D.-H.; Bornschein, J.; Mesnard, T.; and Lin, Z. 2015a. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Bengio, Y.; Mesnard, T.; Fischer, A.; Zhang, S.; and Wu, Y.

2015b. Stdp as presynaptic activity times rate of change of postsynaptic activity. *arXiv preprint arXiv:1509.05936*.

Bi, G.-q., and Poo, M.-m. 2001. Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual review of neuroscience* 24(1):139–166.

Bohte, S. M.; Kok, J. N.; and La Poutre, H. 2002. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48(1):17–37.

Dan, Y., and Poo, M.-m. 2004. Spike timing-dependent plasticity of neural circuits. *Neuron* 44(1):23–30.

Diehl, P. U., and Cook, M. 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience* 9(99).

Diehl, P. U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.-C.; and Pfeiffer, M. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *Proceddings of the 2015 International Joint Conference on Neural Networks (IJCNN 2015)*, 1–8. IEEE.

Hassabis, D.; Kumaran, D.; Summerfield, C.; and Botvinick, M. 2017. Neuroscience-inspired artificial intelligence. *Neuron* 95(2):245–258.

Hausser, M., and Mel, B. 2003. Dendrites: bug or feature? *Current opinion in neurobiology* 13(3):372–383.

Helmstaedter, M. 2015. The mutual inspirations of machine learning and neuroscience. *Neuron* 86(1):25–28.

Hinton, G. E., and Roweis, S. T. 2003. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 857–864.

Hussain, S.; Liu, S.-C.; and Basu, A. 2014. Improved margin multi-class classification using dendritic neurons with morphological learning. In *Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS 2014)*, 2640–2643. IEEE.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553):436–444.

Lee, J. H.; Delbruck, T.; and Pfeiffer, M. 2016. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience* 10.

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

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9:2579–2605.

Merolla, P.; Arthur, J.; Akopyan, F.; Imam, N.; Manohar, R.; and Modha, D. S. 2011. A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm. In *Proceedings of the 2011 IEEE Custom Integrated Circuits Conference (CICC 2011)*, 1–4. IEEE.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

O'Connor, P.; Neil, D.; Liu, S.-C.; Delbruck, T.; and Pfeiffer, M. 2013. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience* 7(178).

Querlioz, D.; Bichler, O.; Dollfus, P.; and Gamrat, C. 2013. Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE transactions on nanotechnology* 12(3):288–295.

Scellier, B., and Bengio, Y. 2017. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience* 11(24).

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2440–2448.

Tang, J.; Liu, J.; Zhang, M.; and Mei, Q. 2016. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web (WWW 2016)*, 287–297. World Wide Web Consortium.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, 2048–2057.

Zeng, Y.; Zhang, T.; and Xu, B. 2017. Improving multi-layer spiking neural networks by incorporating brain-inspired rules. *Science China Information Sciences* 60(5):052201.

Zenke, F.; Agnes, E. J.; and Gerstner, W. 2015. Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks. *Nature Communications* 6:6922.

Zhang, T.; Zeng, Y.; Zhao, D.; Wang, L.; Zhao, Y.; and Xu, B. 2016. Hmsnn: Hippocampus inspired memory spiking neural network. In *Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2016)*, 002301–002306. IEEE.

Zhang, T.; Zeng, Y.; and Xu, B. 2016. Hcnn: A neural network model for combining local and global features towards human-like classification. *International Journal of Pattern Recognition and Artificial Intelligence* 30(01):1655004.

Zhao, B.; Ding, R.; Chen, S.; Linares-Barranco, B.; and Tang, H. 2015. Feedforward categorization on aer motion events using cortex-like features in a spiking neural network. *IEEE transactions on neural networks and learning systems* 26(9):1963–1978.