LETTER **Communicated by George Kastellakis**

Toward a Brain-Inspired Developmental Neural Network Based on Dendritic Spine Dynamics

Feifei Zhao
zhaofeifei2014@ia.ac.cn
Research Center for Brain-Inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Yi Zeng*
yi.zeng@ia.ac.cn
Research Center for Brain-Inspired Intelligence and National Laboratory of Pattern Recognition Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China; School of Future Technology and School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 10049, China; and Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai 200031, China

Jun Bai
jun.bai@ia.ac.cn
Research Center for Brain-Inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Neural networks with a large number of parameters are prone to overfitting problems when trained on a relatively small training set. Introducing weight penalties of regularization is a promising technique for solving this problem. Taking inspiration from the dynamic plasticity of dendritic spines, which plays an important role in the maintenance of memory, this letter proposes a brain-inspired developmental neural network based on dendritic spine dynamics (BDNN-dsd). The dynamic structure changes of dendritic spines include appearing, enlarging, shrinking, and disappearing. Such spine plasticity depends on synaptic activity and can be modulated by experiences—in particular, long-lasting synaptic enhancement/suppression (LTP/LTD), coupled with synapse formation (or enlargement)/elimination (or shrinkage), respectively. Subsequently, spine density characterizes an approximate estimate of the total number of synapses between neurons. Motivated by this, we constrain the weight to a tunable bound that can be adaptively modulated based on synaptic activity. Dynamic weight bound could limit the relatively redundant synapses and facilitate the contributing synapses. Extensive experiments

*Y.Z. is the corresponding author. F.Z. and Y.Z. contributed equally to this work.

Neural Computation **34, 172–189** (2022)
https://doi.org/10.1162/neco_a_01448

demonstrate the effectiveness of our method on classification tasks of different complexity with the MNIST, Fashion MNIST, and CIFAR-10 data sets. Furthermore, compared to dropout and L2 regularization algorithms, our method can improve the network convergence rate and classification performance even for a compact network.

1 Introduction

Deep neural networks (DNNs) bring about superior performance for various machine learning tasks, including image classification (He, Zhang, Ren, & Sun, 2015; Simonyan & Zisserman, 2015; Krizhevsky, Sutskever, & Hinton, 2012), face recognition (Lawrence, Giles, Tsoi, & Back, 1997), video prediction (Deng, Hinton, & Kingsbury, 2013), and speech recognition (Hinton et al., 2012). However, DNNs usually require a large number of parameters, which may cause overfitting. Several regularization methods have been developed to prevent overfitting during network training. These methods include L1 and L2 regularization, weight decay (Moody, Hanson, Krogh, & Hertz, 1995), max-norm constraints, and dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). One potential approach to reduce overfitting is to draw inspiration from the highly complex but efficient central nervous system. After hundreds of millions of years of evolution, the brain neural system shows a powerful ability of generalization learning for multiple cognitive functions. Such learning ability is attributed to adaptively regulate the growth and elimination of dendritic spines.

In the nervous system, changes in spine morphology are temporally associated with the formation and maintenance of memory. As shown in Figure 1, the structure of dendritic spines can be dynamically changed depending on synaptic activity, which plays a vital role in learning and memory (Nimchinsky, Sabatini, & Svoboda, 2002; Kasai, Fukuda, Watanabe, Hayashi-Takagi, & Noguchi, 2010). In particular, the long-lasting synaptic enhancement leads to an enlargement of thin spines into mushroom spines, even the growth of new dendritic spines after repeated inductions of long-term potentiation (LTP), whereas long-term depression (LTD) is accompanied by the elimination and shrinkage of spines (Bourne & Harris, 2007; Kasai, Matsuzaki, Noguchi, Yasumatsu, & Nakahara, 2003; O'Donnell, Nolan, & van Rossum, 2011; Zhou, Homma, & Poo, 2004). Thus, spine density reflects an approximate estimate of the total number of excitatory synapses onto the postsynaptic neuron (Chen, Lu, & Zuo, 2014).

Taking inspiration from the dynamic plasticity of the dendritic spine, we propose a brain-inspired developmental neural network based on dendritic spine dynamics (BDNN-dsd). Our method considers the density of dendritic spines as a tunable weight bound, which will be modulated based on

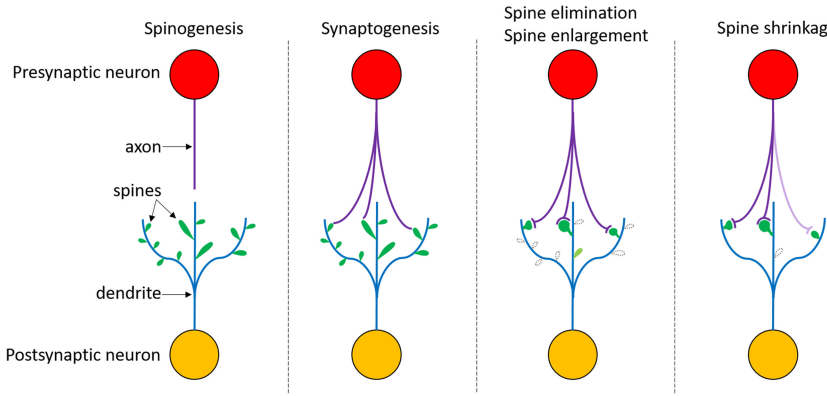

Figure 1: The changes of dendritic spines morphology during learning. The red circle represents the presynaptic neuron, and the yellow circle represents the postsynaptic neuron.

synaptic activities. During the training process, we constrain the weight to this weight bound to limit the relatively redundant synapses and facilitate the contributing synapses. In this letter:

- We propose a novel regularization method inspired by the dynamic plasticity of the dendritic spine, which is biologically inspired.
- We demonstrate the proposed method in different complexities of classification tasks on different data sets, including MNIST, Fashion MNIST, and CIFAR-10 data sets. Extensive experimental results show a consistent improvement in classification performance and convergence speed.
- We compare our method to other regularization methods, such as dropout and L2 regularization. The BDNN-dsd algorithm achieves better convergence performance.

The remainder of the letter is organized as follows. A brief overview of preventing overfit algorithms is given in section 2. The specific implementation details of our proposed approach are described in section 3. The experimental results are given in section 4, and we provide a analysis of our approach in section 5. A conclusion is provided in section 6.

2 Related Work

Much of the previous work on preventing overfit can be divided into two categories: regularization and structure sparse algorithms. Dropout (Srivastava et al., 2014) is used to improve the performance of multilayer

neural networks in various tasks. In dropout, each neuron is probabilistically dropped during training but comes back during inference. The error is then backpropagated through the remaining activated synapses. Extensive experiments show that dropout can reduce overfitting and improve test accuracy. Another regularization method, DropConnect, proposed by Wan, Zeiler, Zhang, Le Cun, and Fergus (2013), randomly selects the subset of weights within the network to zero. DropConnect is helpful for regularizing the neural network. It sometimes outperforms dropout but is slower than the initial network and the dropout network. The MeProp method (Sun, Ren, Ma, & Wang, 2017) updates only a small portion of the gradient information and the corresponding minimal portion of the parameters during each backpropagation step. These regularization methods aim at avoiding overfitting while ignoring the impact on convergence speed. They usually slow learning speed compared with the initial neural network.

Some structure-sparse algorithms such as synaptic pruning methods have been proposed in recent years. Synaptic pruning aims to dynamically compress the network into a very small model with small accuracy degrades. Minimal-value deletion is a simple synaptic pruning of all synapses whose weights are lower than a threshold (Han, Pool, Tran, & Dally, 2015; Chechik, Meilijson, & Ruppin, 1998a, 1998b). The pruned network can be greatly compressed without affecting accuracy (Han et al., 2015). He, Zhang, and Sun (2017) proposed a LASSO regression-based channel pruning and least square reconstruction to prune deep convolutional neural network. Zhuang et al. (2018) considered both discrimination-aware loss and reconstruction loss to minimize the loss of pruned network. He, Liu, Wang, Hu, and Yang (2019) proposed a filter pruning method based on geometric median to prune the most replaceable filters containing redundant information. Yu et al. (2018) proposed the neuron importance score propagation (NISP) algorithm to propagate the importance scores of final responses to every neuron in the network. Then the convolutional neural network is pruned by removing neurons with the least importance. Li, Kadav, Durdanovic, Samet, and Graf (2016) removed whole filters with relatively low weight magnitudes in the network together with their connecting feature maps, reducing computation costs significantly. He, Kang, Dong, Fu, and Yang (2018) proposed a soft pruning method that enabled the pruned filters to be updated when training the model after pruning. To adaptively learn the network topology, Zhao, Zhang, Zeng, and Xu (2017) took into account evolutionary optimization together with information entropy to evaluate the importance of neurons. Inspired by the neural development of the biological brain, which enables a very compact network to complete complex tasks, Zhao and Zeng (2021) proposed a brain-inspired synaptic pruning method to dynamically modulate the network architecture and simultaneously improve network performance.

These pruning methods always result in a significant reduction in network scale at the cost of accuracy. They also ignore the influence on

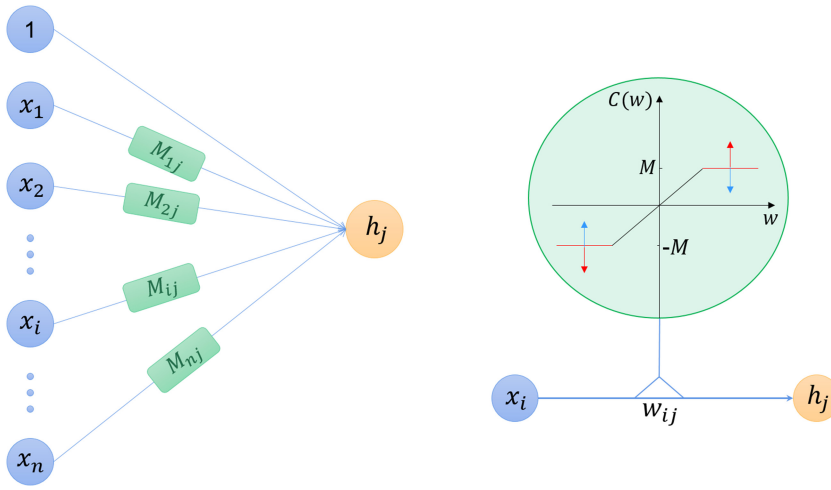

Figure 2: The BDNN-dsd algorithm.

convergence rate. With inspiration from the plasticity of dendritic spine in the brain, we propose a regularization algorithm for small networks and verify the improvement on convergence speed and classification accuracy compared to other regularization methods.

3 Method

In this section, we introduce our developmental neural network, BDNN-dsd. First, we present the overall framework. Next, we show the implementation details for a three-layer, fully connected neural network. Finally, we present a more detailed description of our bound modulation mechanism.

The core neural mechanism of the plasticity of dendritic spines is the dynamic change in spine intensity based on synaptic activity. Inspired by this, we consider the intensity of dendritic spines as a weight bound, which we call a bound mask for a neural network. During the training process, the weight will be optimized through backpropagation and constrained to the bound. The bound will be adaptively updated based on the changing of weight.

As shown in Figure 2, a nontrained neural network would be masked by a bound matrix M_{ij}, which is initialized by R_i: $M_{ij} = R_i$. For each connection w_{ij}, the upper bound is equal to M_{ij}, and the lower bound is equal to $-M_{ij}$: $-M_{ij} \leq w_{ij} \leq M_{ij}$. Consider the jth neuron in the hidden layer; x_i is the input to neuron h_j in the hidden layer; y_j denotes the output of neuron h_j; and w_{ij} and b_j are the weights and biases. The feedforward process in the BDNN-dsd algorithm is calculated in equation 3.1, and the feedback

function is updated as equation 3.2. Then the weights are limited to the bound M as shown in equation 3.3. If the connection weights exceed the bound mask M, they will be reset by the bound value:

$$y_i = f\left(\sum_{i=1}^{n} w_{ij}x_i + b_j\right), \tag{3.1}$$

$$w_{ij} = w_{ij} - \eta\frac{\partial E}{\partial w_{ij}}, \tag{3.2}$$

$$w_{ij} = C\left(w_{ij}\right) = \begin{cases} M_{ij} & w_{ij} > M_{ij} \\ -M_{ij} & w_{ij} < -M_{ij} \\ w_{ij} & otherwise \end{cases}, \tag{3.3}$$

where f is an activation function, such as a sigmoid function $f(x) = 1/(1 + e^{(-x)})$; E is any loss function, for example, the mean square error (MSE) function; and η is the learning rate.

During training, for each batch in each epoch, the bound mask M is dynamically updated through the following two steps:

1. Calculate the number of consecutive times that a synapse exceeds the bound or decays. Synaptic connections that exceed the bound include exceeding the upper bound M_{ij} ($w_{ij} > M_{ij}$) and exceeding the lower bound $-M_{ij}$ ($w_{ij} < -M_{ij}$). Then, for the connections satisfying $w_{ij} > M_{ij}$ or $w_{ij} < -M_{ij}$, the number of consecutive times will be accumulated: $Num^{+}_{ij} = Num^{+}_{ij} + 1$. Synapse decay refers to the decrease of the absolute values of weight in current time compared to the absolute values of weight in previous time: $\left|w_{ij}(t)\right| < \left|w_{ij}(t-1)\right|$. Then the number of consecutive times for the decayed connections will be accumulated: $Num^{-}_{ij} = Num^{-}_{ij} + 1$.
2. Update the bound value whose number of consecutive times exceeds the threshold T_{num}. When the connections satisfy $Num^{+}_{ij} \geq T_{num}$, the bound of those connections will be expanded by R_d: $M_{ij} = M_{ij} + R_d$. When the connections satisfy $Num^{-}_{ij} \geq T_{num}$, the bound of those connections will shrink by R_d: $M_{ij} = M_{ij} - R_d$.

The framework of the BDNN-dsd algorithm is shown in algorithm 1.

4 Experiments

In this section, we perform the experiments on CPU with a three-layer ANN. The activation function for neurons in the input and the hidden layer is a sigmoid function. We use softmax activation function in the output layer. The learning rate is equal to 0.1, and the number of iterations is 500. The batch size is set to 600.

Algorithm 1: Algorithm Description of BDNN-dsd.

Input: Training data X;

Given: Initialized bound mask $M_{ij} = R_i$; Changed value R_d; Threshold T_{num}; Initialized consecutive time $Num_{ij}^{+} = Num_{ij}^{-} = 0$; Time $t = 0$;

Initialize: Model parameter W.

for $epoch = 1; epoch \leq epoch_{max}; epoch++$ **do**
  for $batch = 1; batch \leq Num_{batch}; batch++$ **do**
    $t = t + 1$;
    Feed forward computation from equation 3.1.
    for each connection w_{ij} **do**
      Backpropagation computation from equation 3.2.
      if $w_{ij} > M_{ij}$ or $w_{ij} < -M_{ij}$ **then**
        $Num_{ij}^{+} = Num_{ij}^{+} + 1$;
      end if
      if $|w_{ij}(t)| < |w_{ij}(t-1)|$ **then**
        $Num_{ij}^{-} = Num_{ij}^{-} + 1$;
      end if
      if $Num_{ij}^{+} \geq T_{num}$ **then**
        $M_{ij} = M_{ij} + R_d$;
        $Num_{ij}^{+} = 0$;
      end if
      if $Num_{ij}^{-} \geq T_{num}$ **then**
        $M_{ij} = M_{ij} - R_d$;
        $Num_{ij}^{-} = 0$;
      end if
      Limit the weight to bound mask from equation 3.3.
    end for
  end for
end for

To illustrate the superiority of our proposed model, we test it on different tasks, including different data sets, training samples with different complexities, and different network scales. We compare our algorithm with two regularized algorithms, dropout and L2 regularization. Here, the parameters in both are selected empirically with the best performance. We verify our method on the improvement of testing accuracy and convergence speed compared with the initial neural network, the network with dropout, and the network with L2 regularization. The test accuracies of the initial network A_{init}, dropout A_{drop}, L2 regularization A_{L2} and the BDNN-dsd $A_{BDNN-dsd}$ methods are calculated by the average of five experiments.

To verify the improvement in convergence speed, we calculate the improvement of learning speed for each regularized method with respect to the initial network with the best performance. First, we get the time set T^b and T^c; the initial network and regularized methods (including the dropout, L2, and BDNN-dsd) have the same accuracy. As a result, for any ith element

Table 1: The Comparative Results of Testing Accuracy on 60,000 MNIST Training Samples.

Network \Acc(%)	A_{init}	A_{drop}	A_{L2}	$A_{BDNN-dsd}$
10 neurons	91.6 (± 0.27)	88.1 (± 0.41)	92.61 (± 0.3)	**93.16 (± 0.09)**
100 neurons	95.27 (± 0.08)	95.29 (± 0.085)	96.43 (± 0.11)	**96.94 (± 0.06)**
500 neurons	96.18 (± 0.13)	96.54 (± 0.06)	97.1 (± 0.06)	**97.5 (± 0.03)**

Note: The best performance on different networks is shown in bold.

Table 2: The Comparative Results of Convergence Rate on 60,000 MNIST Training Samples.

	L_{drop}	L_{L2}	$L_{BDNN-dsd}$
10 neurons	0.51 (± 0.093)	1.22 (± 0.164)	**5.78 (± 0.64)**
100 neurons	0.733 (± 0.033)	1.38 (± 0.083)	**6.41 (± 0.49)**
500 neurons	0.63 (± 0.045)	1.11 (± 0.11)	**3**

Note: The best performance on different networks is shown in bold.

in set T^b and T^c, the accuracy of the initial network a_i^b is equal to the accuracy of the regularized method a_i^c. Then we find the index i with maximal difference of learning time between the initial network T_i^b and the regularized methods T_i^c. Finally, the improvement of learning speed L is calculated as

$$L = \frac{T_i^c}{T_i^b},\ s.t.\ a_i^b = a_i^c, i = \max_i \left|T_i^b - T_i^c\right| \tag{4.1}$$

Then the improvements of learning speed of dropout L_{drop}, L2 regularization L_{L2}, and BDNN-dsd $L_{BDNN-dsd}$ methods are calculated by the average on five experiments. The detailed experimental results follow.

4.1 Experiments on MNIST. The MNIST classification data set contains 10 classes of handwritten digits from 0 to 9. It contains 60,000 training samples and 10,000 testing samples (Lecun, Bottou, Bengio, & Haffner, 1998). Each sample is represented by a 28×28 digital image. The initial ANN has 784 neurons in the input layer, and the number of neurons in the output layer is 10. To verify the robustness of our method on the MNIST data set, we conduct experiments on 1200 and 60,000 training samples respectively.

4.1.1 Results on 60,000 Training Samples. For 60,000 training samples, we test our method on 10, 100, and 500 neurons in the hidden layer. As shown in Tables 1 and 2, our BDNN-dsd achieves the highest accuracy and fastest

Table 3: The Comparative Results of Testing Accuracy on 1200 MNIST Training Samples.

Network \Acc(%)	A_{init}	A_{drop}	A_{L2}	$A_{BDNN-dsd}$
10 neurons	81.38 (± 1.08)	78.97 (± 0.57)	81.33 (± 0.86)	**83.76 (± 0.34)**
100 neurons	85.01 (± 0.31)	87.74 (± 0.19)	88.65 (± 0.13)	**89.56 (± 0.13)**

Note: The best performance on different networks is shown in bold.

Table 4: The Comparative Results of Convergence Rate on 1200 MNIST Training Samples.

	L_{drop}	L_{L2}	$L_{BDNN-dsd}$
10 neurons	2.489 (± 0.243)	4.98 (± 1.11)	**7.21 (± 1.265)**
100 neurons	2.22 (± 0.7)	2.822 (± 0.18)	**9.014 (± 0.74)**

Note: The best performance on different networks is shown in bold.

convergence speed compared to the initial network, dropout, and L2 regularization. For example, for 10 neurons in the hidden layer, dropout slows the convergence rate with 3.5% accuracy loss. For 100 and 500 neurons in the hidden layer, dropout has little improvement on accuracy, with nearly half of learning speed drop. In all cases, L2 regularization achieves comparable learning speed with only 1% accuracy improvements. Compared to dropout and L2 regularization, our method could accelerate convergence (by 3× to 6.41× speedup) with even better performances (more than 1% accuracy improvement) for different networks.

4.1.2 Results on 1200 Training Samples. A network with good generalization should work well on both large and small training samples. For a small data task with 1200 training samples, the comparative results of testing accuracy and convergence rate are shown in Tables 3 and 4. Our method outperforms the initial network, dropout, and L2 regularization in two aspects. First, our method achieves better performance than the others. Second, our method converges much faster than the others. Especially when the network is too small (with 10 neurons in the hidden layer), dropout and L2 regularization make no contribution to improving classification performance, while our method can improve testing accuracy on the initial network from 81.38% to 84.76% with 7.21× speedup. In addition, BDNN-dsd accelerates the network with 100 hidden neurons by 9.014 times with 4.55% accuracy improvement, which outperforms dropout and L2 regularization, which have less than 3 times speed-up.

Table 5: The Comparative Results of Testing Accuracy on 60,000 Fashion MNIST Training Samples.

Network \Acc(%)	A_{init}	A_{drop}	A_{L2}	$A_{BDNN-dsd}$
10 neurons	84.33 (± 0.13)	84.09 (± 0.16)	84.47 (± 0.046)	**85.16 (± 0.24)**
100 neurons	86.37 (± 0.25)	87.68 (± 0.073)	87.64 (± 0.07)	**88.64 (± 0.04)**
500 neurons	87.94 (± 0.31)	88.93 (± 0.09)	89.02 (± 0.07)	**89.56 (± 0.03)**

Note: The best performance on different networks is shown in bold.

Table 6: The Comparative Results of Convergence Rate on 60,000 Fashion MNIST Training Samples.

	L_{drop}	L_{L2}	$L_{BDNN-dsd}$
10 neurons	0.99 (± 0.1)	0.96 (± 0.11)	**3.29 (± 0.36)**
100 neurons	0.692 (± 0.063)	0.884 (± 0.295)	**3.2 (± 0.4)**
500 neurons	0.66 (± 0.055)	1.66 (± 0.21)	**2.83 (± 0.37)**

Note: The best performance on different networks is shown in bold.

4.2 Experiments on Fashion MNIST. The Fashion MNIST classification data set contains 10 classes: T-shirt, trouser, pullover, dress, coat, sandals, shirt, sneaker, bag, and ankle boots. It contains 28×28 grayscale images of 60,000 training samples and 10,000 testing samples (Xiao, Rasul, & Vollgraf, 2017). The detailed comparative results on 1200 and 60,000 training samples follows.

4.2.1 Results on 60,000 Training Samples. The comparative results of test accuracy and convergence rate are depicted in Tables 5 and 6. Not surprisingly, BDNN-dsd is superior to the initial network, dropout, and L2 regularization. From the perspective of performance, it achieves the best accuracy in different networks and nearly improves 2% accuracy. The additional advantage of our model lies in the improvement of convergence speed. L2 regularization accelerates the learning only when the network has 500 neurons in the hidden layer and slows learning when the network has 10 and 100 neurons in the hidden layer. Dropout could not improve the learning speed in all cases. Our method can improve the convergence rate by about 3 times speedup. These results demonstrate that BDNN-dsd can avoid overfitting to some extent with better performance and faster convergence rate.

4.2.2 Results on 1200 Training Samples. Tables 7 and 8 show the comparative results of testing accuracy and convergence rate on 1200 Fashion MNIST training samples. Similar to the observations from the experiment on MNIST, our method outperforms the other three algorithms. Even for

Table 7: The Comparative Results of Testing Accuracy on 1200 Fashion MNIST Training Samples.

Network \Acc(%)	A_{init}	A_{drop}	A_{L2}	$A_{BDNN-dsd}$
10 neurons	76.03 (± 0.42)	73.63 (± 0.97)	75.93 (± 0.26)	**78.54 (± 0.18)**
100 neurons	78.12 (± 0.3)	79.89 (± 0.17)	80.49 (± 0.06)	**80.77 (± 0.18)**

Note: The best performance on different networks is shown in bold.

Table 8: The Comparative Results of Convergence Rate on 1200 Fashion MNIST Training Samples.

	L_{drop}	L_{L2}	$L_{BDNN-dsd}$
10 neurons	0.75 (± 0.26)	1.133 (± 0.17)	**3.33 (± 0.65)**
100 neurons	1.32 (± 0.18)	2.01 (± 0.11)	**3.2 (± 0.11)**

Note: The best performance on different networks is shown in bold.

Table 9: The Comparative Results of Testing Accuracy on CIFAR-10.

Network \Acc(%)	A_{init}	A_{drop}	A_{L2}	$A_{BDNN-dsd}$
100 neurons	47.77 (± 0.46)	49.21 (± 0.85)	49.78 (± 0.23)	**50.98 (± 0.27)**
500 neurons	51.88 (± 0.61)	**55.53 (± 0.18)**	50.59 (± 0.32)	55.24 (± 0.11)

Note: The best performance on different networks is shown in bold.

the compact network with 10 hidden neurons, our method achieves 2.51% accuracy improvement with a 3.33 times speed-up on the initial network. L2 regularization and dropout have 0.1% and 2.4% accuracy drop, respectively, with no improvement in learning speed. For 100 neurons in the hidden layer, our method also achieves the best accuracy and fastest learning speed.

4.3 Experiments on CIFAR-10. The CIFAR-10 data set consists of 60,000 color images. There are 50,000 training samples and 10,000 testing samples. The CIFAR-10 data set contains 10 categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck (Krizhevsky, 2009). Tables 9 and 10 show the comparative results of testing accuracy and convergence rate. For 100 neurons in the hidden layer, the BDNN-dsd algorithm achieves the best performance: 50.98% compared with the initial network, dropout, and L2 regularization. L2 regularization achieves the fastest learning speed at the cost of 1% accuracy drops compared to our method. For 500 neurons in the hidden layer, the dropout method improves the performance from 51.88% to 55.53%, and our method achieves 55.24% accuracy, which shows

Table 10: The Comparative Results of Convergence Rate on CIFAR-10.

	L_{drop}	L_{L2}	$L_{BDNN-dsd}$
100 neurons	2.79 ($\pm$ 0.57)	**2.87 ($\pm$ 0.45)**	2.31($\pm$ 0.36)
500 neurons	2.33 ($\pm$ 0.21)	2.04 ($\pm$ 0.94)	**2.38 ($\pm$ 0.44)**

Note: The best performance on different networks is shown in bold.

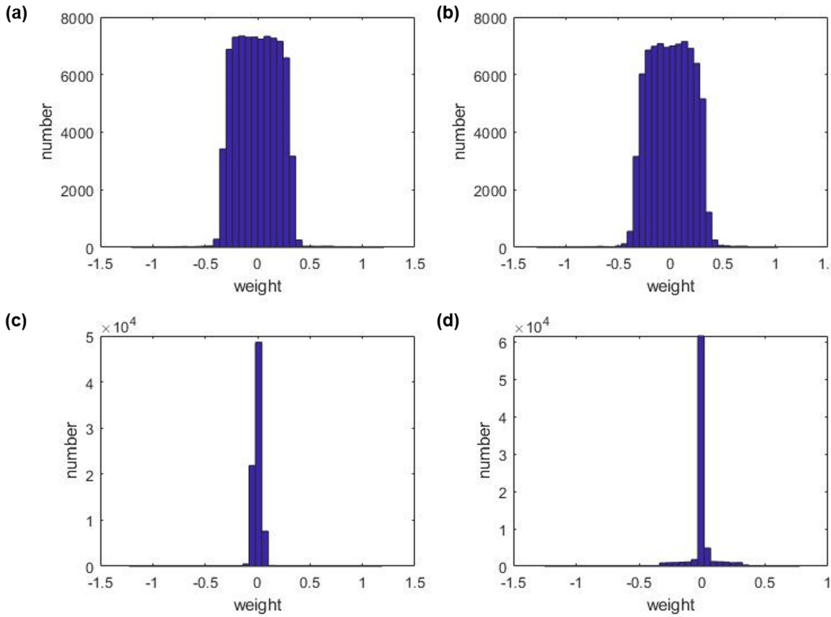

Figure 3: The weights distribution histograms of initial network (a), dropout (b), L2 regularization (c), and our method (d).

a comparable performance. In comparison to L2 regularization, which accelerates 2.04 times with a 1.3% accuracy drop, our method achieves 2.38 times speedup with 3.36% accuracy improvement.

4.4 Effect on Weights. We use 1200 MNIST training samples trained on 100 hidden neurons as an example. The histograms shown in Figure 3 represent the weights distribution of the initial network, dropout, L2 regularization, and our method. Though dropout randomly eliminates some neurons during training, it has little impact on reducing weights. L2 regularization as a kind of weight decay algorithm can extremely limit the weight to

zero. Compared to L2 regularization, our proposed method can bring about superior performance and a convergence rate with even smaller weights. Unlike L2 regularization, which prevents overfitting through supplementing a penalty term to the loss function, our BDNN-dsd method optimizes the bound of weights rather than the weights themselves, which leads to better convergence performance.

4.5 Influence of the BDNN-dsd Parameters. In BDNN-dsd, there are three tunable hyperparameters: the initial bound value R_i, the changed value of bound R_d, and the consecutive threshold T_{num}. Here we explore the effect of varying these hyperparameters. The comparative results, shown in Figure 4, are conducted on the MNIST data set with 100 neurons in hidden layer.

In this letter, we set $R_i = 0.45$, $R_d = 0.06$, and $T_{num} = 10$. If the R_i and R_d are held constant, having a larger T_{num} means a greater degree of limitation on connections, which leads to a decrease in performance. From Figures 4a and 4b, we see that as T_{num} increases, the convergence performance falls. It always outperforms the initial network and dropout, but it is inferior to L2 regularization when $T_{num} \geq 30$. From Figures 4c and 4d, when the R_i and T_{num} are constant, the BDNN-dsd method is always superior to others for $0.03 \leq R_d \leq 0.15$, which means that the proposed method is robust to the changing R_d. Moreover, as R_d increases, BDNN-dsd shows a balance accuracy and convergence rate. When R_d and T_{num} are fixed, our method consistently outperforms the other methods in both accuracy (see Figure 4e) and learning speed (see Figure 4f). Thus, we can conclude that BDNN-dsd has good robustness for various R_i and R_d, sensitive only to the T_{num}. Besides, through empirically defining T_{num}, our method can still outperform initial network, dropout, and L2 regularization.

5 Discussion

We have proposed a brain-inspired developmental neural network based on dendritic spine dynamics. The proposed BDNN-dsd algorithm considered the dynamic plasticity of the dendritic spine to limit the redundant synapses for avoiding overfitting and facilitating the contributing synapses for accelerating convergence. Extensive experiments verified the effectiveness of the BDNN-dsd algorithm on the improvement of classification performance and learning speed in classification tasks with different complexity. Compared with the dropout and L2 regularization methods, our proposed method achieved better accuracy and a faster convergence rate.

Unlike the dropout (Srivastava et al., 2014) and dropconnect (Wan et al., 2013) methods, which randomly dropped some connections or neurons in each epoch, our BDNN-dsd method regularized the network by optimizing a tunable bound to limit the activity of synapses. Network pruning methods

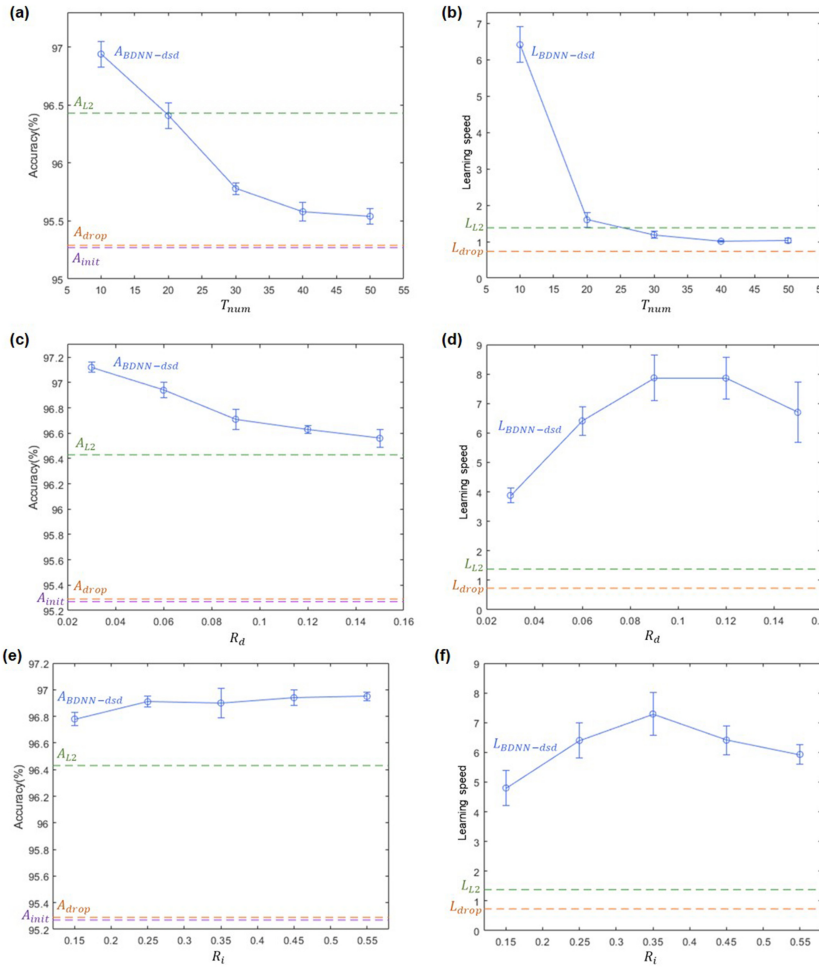

Figure 4: The effect of changing T_{num} on accuracy (a) and convergence rate (b). The effect of changing R_d on accuracy (c) and convergence rate (d). The effect of changing R_i on accuracy (e) and convergence rate (f).

aimed to compress a deep convolutional neural network by pruning unimportant filters or weights. Different from the network pruning methods, the BDNN-dsd method does regularization by adaptively limiting the weights to very small values. During the training process, the bound of the synapse may be modulated to zero, which will result in the zeroing weight. Whether these zeroing weights will be dead permanently and cause a problem similar to "Dead ReLU" is an important issue that needs to be addressed.

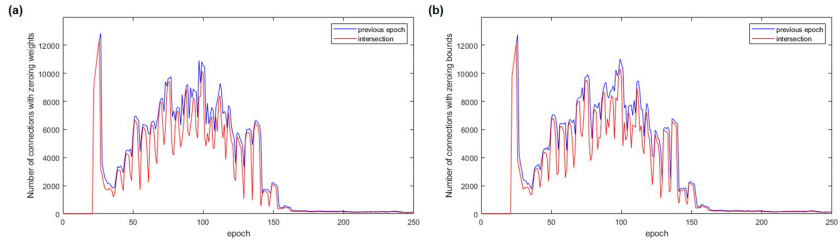

Figure 5: During the training process, the number of connections with zeroing weights (a) and bounds (b).

Take 1200 MNIST training samples trained on 100 hidden neurons as an example. For each epoch, we calculated zero-weight synaptic set and zero-bound synaptic set containing zeroing weight and zeroing bound respectively. Then we calculated the zero-weight (and zero-bound) intersection set of the previous epoch and the current epoch. The blue line shown in Figure 5a represents the number of zero-weight synaptics set in the previous epoch (k-1th epoch), and the red line represents the number of zero-weight intersection (of k-1th and kth epoch) set in the current epoch. Figure 5b depicts the number of zero-bound synaptic set and the intersection set.

The trends of lines in Figures 5a and 5b are similar, which verified the effect of the bound mask on limiting weight. In the first 20 epochs, there is no zero weight and bound. Then the bounds of some connections with $Num_{ij}^{-} \geq T_{num}$ suddenly become zero at the 21st epoch, which also results in the zeroing weights. From the 21st to the 25th epoch, the number of zero-weight intersection sets is the same as the number of the zero-weight synaptic sets (in the previous epoch), which means the zeroing weights continuously keep zero in this epoch interval. After the 25th epoch, the number of zero-weight intersection sets and synaptic sets sharply falls. This phenomenon demonstrates that zeroing weight could be optimized to nonzero by backpropagation and the tunable bound. And when the cumulative number of synapses beyond zero bound satisfies $Num_{ij}^{+} \geq T_{num}$, the bound will be modulated to nonzero and the weight will be nonzero as well. Here five epochs needed to update bound is consistent with the fixed-parameter $T_{num} = 10$ (since the batch size is 600). As shown in Figure 5, the number of zero-weight intersection sets are smaller than the numbers of zero-weight synaptic sets from the 25th to the 250th epoch, which indicates that the zeroing weight in the previous epoch will be nonzero in the current epoch. After the 250th epoch, the number of zero-weight and zero-bound intersection sets are always than 100. These results suggest that the zeroing weight will not be permanently deleted and will be adaptively optimized by backpropagation and the tunable bound.

6 Conclusion

This letter proposed a new approach to perform regularization for neural networks in an efficient manner. Inspired by the plasticity of dendritic spines, the proposed method introduces a dynamic weight bound to constrain synaptic activity. The bound increases for the synapses that constantly go beyond the bound and decreases for synapses that constantly decay. Experiments reveal that our method can speed up the network convergence rate and improve classification performance even for a compact network. Our results also demonstrate superior performance compared to other regularization methods. The brain developmental process has a complex and optimal learning and processing mechanism. To develop a more brain-like intelligence, we will do deep research on the more detailed and deep developmental mechanism and enlighten artificial intelligence on the typical tasks.

Acknowledgments

This work is supported by the National Key Research and Development Program (grant 2020AAA107800), the Strategic Priority Research Program of the Chinese Academy of Sciences (grant XDB32070100), the Beijing Municipal Commission of Science and Technology (grant Z181100001518006), the Key Research Program of Frontier Sciences, Chinese Academy of Sciences (grant ZDBS-LY-JSC013), and Beijing Academy of Artificial Intelligence.

References

Bourne, J., & Harris, K. M. (2007). Do thin spines learn to be mushroom spines that remember? *Current Opinion in Neurobiology*, *17*(3), 381–386. 10.1016/j.conb.2007.04.009, PubMed: 17498943

Chechik, G., Meilijson, I., & Ruppin, E. (1998a). Synaptic pruning in development: A computational account. *Neural Computation*, *10*(7), 1759–1777. 10.1162/089976698300017124

Chechik, G., Meilijson, I., & Ruppin, E. (1998b). Synaptic pruning in development: A novel account in neural terms. In J. M. Bower (Ed.), *Computational neuroscience* (pp. 149–154). New York: Springer Science+Business Media.

Chen, C.-C., Lu, J., & Zuo, Y. (2014). Spatiotemporal dynamics of dendritic spines in the living brain. *Frontiers in Neuroanatomy*, *8*, 28. 10.3389/fnana.2014.00028

Deng, L., Hinton, G., & Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: an overview. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 8599–8603). Piscataway, NJ: IEEE.

Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*, *28* (pp. 1135–1143). Red Hook, NY: Curran.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1026–1034). Piscataway, NJ: IEEE.

He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). *Soft filter pruning for accelerating deep convolutional neural networks.* arXiv:1808.06866.

He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Y. (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4340–4349). Piscataway, NJ: IEEE.

He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1389–1397). Piscataway, NJ: IEEE.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., . . . Sainath, T. N. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97. 10.1109/MSP.2012.2205597

Kasai, H., Fukuda, M., Watanabe, S., Hayashi-Takagi, A., & Noguchi, J. (2010). Structural dynamics of dendritic spines in memory and cognition. *Trends in Neurosciences*, *33*(3), 121–129. 10.1016/j.tins.2010.01.001, PubMed: 20138375

Kasai, H., Matsuzaki, M., Noguchi, J., Yasumatsu, N., & Nakahara, H. (2003). Structure-stability-function relationships of dendritic spines. *Trends in Neurosciences*, *26*(7), 360–368. 10.1016/S0166-2236(03)00162-0, PubMed: 12850432

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images.* University of Toronto technical report.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *25* (pp. 1097–1105). Red Hook, NY: Curran.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural network approach. *IEEE Transactions on Neural Networks*, *8*(1), 98–113. 10.1109/72.554195, PubMed: 18255614

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, *86*, 2278–2324. 10.1109/5.726791

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2016). *Pruning filters for efficient ConvNets.* arXiv:1608.08710.

Moody, J., Hanson, S., Krogh, A., & Hertz, J. A. (1995). A simple weight decay can improve generalization. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems*, *8* (pp. 757–763). Cambridge, MA: MIT Press.

Nimchinsky, E. A., Sabatini, B. L., & Svoboda, K. (2002). Structure and function of dendritic spines. *Annual Review of Physiology*, *64*, 313–353. 10.1146/annurev.physiol.64.081501.160008, PubMed: 11826272

O'Donnell, C., Nolan, M. F., & van Rossum, M. C. (2011). Dendritic spine dynamics regulate the long-term stability of synaptic plasticity. *Journal of Neuroscience*, *31*(45), 16142–16156.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the Conference on Learning Representations*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Sun, X., Ren, X., Ma, S., & Wang, H. (2017). *meProp: Sparsified back propagation for accelerated deep learning with reduced overfitting*. arXiv:1706.06197.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013). Regularization of neural networks using DropConnect. In *Proceedings of the International Conference on Machine Learning* (pp. 1058–1066).

Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms.* arXiv:1708.07747v2.

Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., . . . Davis, L. S. (2018). NISP: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 9194–9203). Piscataway, NJ: IEEE.

Zhao, F., & Zeng, Y. (2021). Dynamically optimizing network structure based on synaptic pruning in the brain. *Frontiers in Systems Neuroscience*, *15*, 55. 10.3389/fnsys.2021.620558, PubMed: 34177473

Zhao, F., Zhang, T., Zeng, Y., & Xu, B. (2017). Towards a brain-inspired developmental neural network by adaptive synaptic pruning. In *Proceedings of the International Conference on Neural Information Processing* (pp. 182–191). Berlin: Springer.

Zhou, Q., Homma, K. J., & Poo, M.-m. (2004). Shrinkage of dendritic spines associated with long-term depression of hippocampal synapses. *Neuron*, *44*(5), 749–757. 10.1016/j.neuron.2004.11.011, PubMed: 15572107

Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., & Zhu, J. (2018). Discrimination-aware channel pruning for deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*, *31* (pp. 875–886). Red Hook, NY: Curran.

Received February 20, 2021; accepted July 14, 2021.