Towards a Brain-inspired Developmental Neural Network by Adaptive Synaptic Pruning

Feifei Zhao[1,2], Tielin Zhang[1], Yi Zeng[1,2,3] ⋆, and Bo Xu[1,2,3]

[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, China
Email: {zhaofeifei2014, tielin.zhang, yi.zeng}@ia.ac.cn

Abstract. It is widely accepted that appropriate network topology should be empirically predefined before training a specific neural network learning task. However, in most cases, these carefully designed networks are easily falling into two kinds of dilemmas: 1) When the data is not enough to train the network well, it will get an underfitting result. 2) When networks have learned too much patterns, they are likely to lead to an overfitting result and have a poor performance on processing new data or transferring to other tasks. Inspired by the synaptic pruning characteristics of the human brain, we propose a brain-inspired developmental neural network (BDNN) algorithm by adaptive synaptic pruning (BDNN-sp) which could get rid of the overfitting and underfitting. The BDNN-sp algorithm adaptively modulates network topology by pruning useless neurons dynamically. In addition, the evolutional optimization method makes the network stop on an appropriate network topology with the best consideration of accuracy and adaptability. Experimental results indicate that the proposed algorithm could automatically find the optimal network topology and the network complexity could adaptively increase along with the increase of task complexity. Compared to the traditional topology-predefined networks, trained BDNN-sp has the similar accuracy but better transfer learning abilities.

Keywords: brain-inspired developmental neural network, brain-inspired pruning rules, structural plasticity, network adaptability, synaptic pruning

1 Introduction

Recently, neural network models, such as deep neural networks (DNN), have been widely used in many different domains including computer vision (CV) and natural language processing (NLP)[1, 2], etc. Inspired by the hierarchical information processing mechanism in the brain, DNN has achieved big improvements

⋆ Feifei Zhao and Tielin Zhang contributed equally to this work and should be considered as co-first authors, and the corresponding author is Yi Zeng.

on the tasks of image classification[3], face recognition[4] and video prediction[5]. However, there are still many opportunities to improve the models. Firstly, it is hard to measure the complexity of data before processing by the model. There is no criterion to evaluate the complexity of the data, for example, the data with simple pattern types but large number of samples, or complex patterns and small number of samples. Secondly, the huge number of hyper-parameters on DNN leads the model time-consuming and easier to overfit. More importantly, the topologies of DNN are usually empirically set for given tasks which may not be suitable for other task complexities [6]. Thirdly, the training time of DNN mostly depends on the personal settings of specific variables such as iteration time, patch size or learning rate, and these variables are usually set by experiences or training tricks. The network usually could not be predefined at the best network state.

For the traditional DNNs, they only depend on the integration of lost function minimization and weights regulations[7] to avoid overfitting. As shown in Fig. 1(A), the precision error differences between training sample and test sample could be used to find the best fitting point, and the weights regulations will make the differences smaller. However, these methods are still with predefined network topology and even need more tricks to train a network well.

Some existing pruning neural network methods make the dynamical optimization possible, such as evolutionary artificial neural network (EANN). These kinds of methods usually calculate a fitness value to evaluate the performance of the network, and the fitness values usually include the descriptions of classification accuracy (e.g. the reciprocal of error, the mean square error[8, 9] or the cross entropy error[10]) and network scales (e.g., the number of neurons or connections[11, 12]). These methods focus on finding an appropriate model complexity which has the best balance between the network size and the test accuracy. However, the test accuracy is actually acceptable among a wide range of network size, as the solid black line shown in Fig. 1(B). The maximal accuracy could not reflect the comprehensive performance of network such as adaptability.

From information processing perspective, a complex neural network is considered to be a major part of the brain. In human brain, the process of eliminating synapses is important during the development period. Research has evidenced that the development period contains synaptic overgrowth in infant brain, then surplus synapses are gradually eliminated throughout childhood and adolescence[13]. Synapses that are rarely used are more likely to be eliminated during the pruning process[14]. The process of overgrowth first and then elimination is actually an efficient way for brain to achieve optimal development[13, 15].

Inspired by the pruning mechanism in human brain, we propose a version of brain-inspired developmental neural network algorithm to optimize network topology structure by iteratively pruning useless neurons until the fitness function reaches the peak (BDNN-sp for short). The most appropriate network size is the red dashed line in Fig. 1(B). The network could automatically find the optimal topology and the network complexity could adaptively increase along with the increase of the task complexity, as shown in Fig. 1(C).

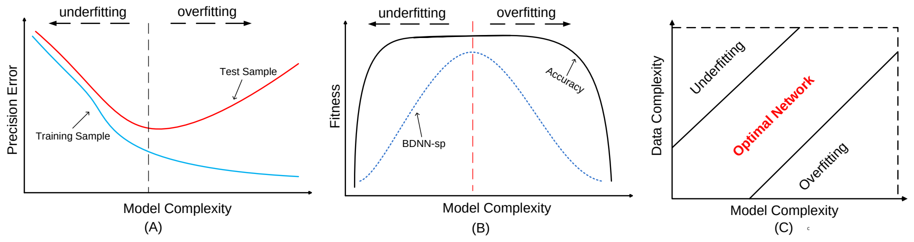

Fig. 1: Underfitting and overfitting in traditional DNN**(A)** and BDNN-sp**(B)**. The optimal network**(C)** has the very model complexity which is complex enough for the data complexity and will not cause overfitting or underfitting.

This paper is organized as follows. Section 2 introduces the detailed methodology of the proposed BDNN-sp algorithm. Section 3 verifies the adaptability and transfer learning ability of the BDNN-sp algorithm on various complexities of tasks. A conclusion is provided in Section 4.

2 Methodology of the BDNN-sp Algorithm

The BDNN-sp algorithm is a member and a branch of a series of Brain-inspired Developmental Neural Networks (BDNN). It mainly contains two parts, the first part is the brain inspired neuron pruning method on dynamically modulating network topology, and the second part is the main developmental learning process of the BDNN-sp network.

2.1 Brain-inspired Neural Network Pruning Rules

When learning a new task, a proportion of synapses are strengthened with increase of the volume of dendritic spines, and a proportion of synapses are weakened. Critically, these enlarged dendritic spines play an important role in performing this task, and they persist for a long time despite the subsequent learning of other tasks[16]. Meanwhile, the useless dendritic spines even neurons are gradually eliminated during the learning process, as shown in Fig. 2(A).

There are lots of synapses eliminated during the development process of the brain. The reasons can be summarized as the following hypotheses, such as saving spaces for storing new knowledge, or reducing the number of variables for better satisfying the outside environment, or increasing the response speed to some specific stimulus. The pruning process of the brain plays a main role during the procedure of brain and cognitive function development[13, 17, 18]. In this paper, inspired by the pruning mechanism of the human brain, in which the more weaker synapses even neurons are easier to be eliminated, we optimize the topology of artificial neural network (ANN) by pruning unimportant neurons for simplification. The basic process of pruning ANN is shown in Fig. 2(B).

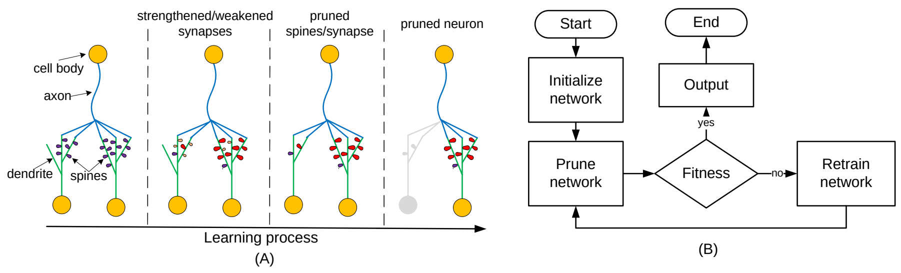

Fig. 2: The pruning process in human brain**(A)** and in BDNN-sp**(B)**.

We firstly build a three-layer ANN as initial network and ensure that the network is with enough variables (or enough complexity). Then, we shrink the complexity of the network by pruning the least important neurons. The reason is that unimportant neurons have little effect on the output. Then, we iteratively retrain the remaining network and prune unimportant neurons until the fitness function reaches the peak.

On the neuron level, every time we eliminate a proportion of the most unimportant neurons through pruning all the input and output connections of these neurons. We sum the total input and output weights of each neuron to evaluate its importance. Since presynaptic neuron may have both excitatory and inhibitory effects on postsynaptic neuron, then the importance degree of each neuron is calculated as shown in Equation (1), where N_{in} is the number of connections sent to neuron j, and N_{out} is the number of connections sent from neuron j. w_{ij} is the weight from neuron i to j, and w_{jk} is the weight from neuron j to k.

$$I_j = \sum_{i}^{N_{in}} |w_{ij}| + \sum_{k}^{N_{out}} |w_{jk}| \tag{1}$$

After pruning the proportion of unimportant neurons, we retrain the remaining network through back propagation (BP) method to update remaining weights. Then, we evaluate the adaptability of network by fitness function which is introduced in Section 2.2. This process will be iteratively executed until the network reaches the best fitness. The framework of BDNN-sp algorithm is shown in Algorithm 1.

2.2 Fitness Function in BDNN-sp

This subsection introduces the fitness function which is the criteria of the adaptability for the network. This fitness function contains some evaluative conditions of network stated as the following:

Algorithm 1 The framework of the BDNN-sp algorithm.

```
Input: Initial neural network with enough complexity;
Output: Pruned neural network;
1: Calculate the importance degree $I_j$ for each neuron $j$;
2: Prune a proportion of the most unimportant neurons;
3: Retrain the remaining neural network by BP;
4: Calculate the fitness function $F$;
5: Repeat Step 1 to Step 4 until fitness function $F$ reaches the peak;
```

(1) **Classification Performance.**
We measures the classification performance of network by test accuracy. It is calculated by the ratio of accurate number to the total number of test samples, as shown in Equation (2), where N_c represents the number of test samples that are correctly classified, and N_s is the total number of test samples. Classification accuracy is the benchmark to evaluate the classification performance, as well as the adaptability of the network.

$$A = \frac{N_c}{N_s} = 1 - error \tag{2}$$

(2) **Network Stability.**
Information entropy is one of the most widely used measurement theory of information [19]. Generally, entropy represents uncertainty or disorder. If the outcome of an event has big probability, the entropy is small because it gives little new information. If the outcome of an event is unpredictable (small probability), the entropy will be large because it may contain some new information[20]. This paper proposes an entropy-like measurement to calculate the stability of the neural network.
Firstly, we define the difference between un-retrained weights and retrained weights as probabilities distribution, as shown in Equation (3), where w_p represents the weights after pruning the most unimportant neurons in the network, and w_r represents the weights after retraining this pruned network. p represents the variation of the remaining weights. If the pruned neurons play little role in the network, then the p will be small and entropy will be large. After pruning to a certain extent, the remaining network is greatly influenced by the pruned neurons, then p will be fluctuant greatly and the entropy will be small. The equation of the entropy is shown in Equation (4), where N_c is the number of remaining connections.

$$p = |w_r - w_p| \tag{3}$$

$$H(p) = -\sum_{i}^{N_c} p_i \log_2 p_i \tag{4}$$

Then we normalize the $H(p)$ as stability function, as shown in Equation (5). S is effective for representing the stability of the network because it can reflect the change of overfitting and underfitting. At the beginning, the initial network complexity is so excess that it is easier to overfit. With the pruning of unimportant neurons, the network becomes smaller and the S will gradually increase until the network reaches the best stability. After that, if we go on pruning the network, it will be too small to learn the training data well. Then underfitting occurred, the S will start to decrease.

$$S = \frac{H\left(p\right)}{\max_i\left(\left|p_i \log_2 p_i\right|\right) * N_c} \tag{5}$$

(3) **Mean of Weights.**
The mean of weights is calculated by Equation (6), where n is the number of neurons in input layer, and m is the number of neurons in hidden layer, w_{ij} is the connection weight from neuron i to neuron j, w_{jk} is the connection weight from neuron j to neuron k. Mean of weights could reflect the change of connection weights. The more information it needs to represent, the larger weight value it will be with. Hence the mean of weights increases along with the increase of pruning ratio and task's complexity. Mean of weights is useful to balance the distribution of weights. When the network is complex enough, the mean of weights will be small, which leads the network easier to overfit.

$$E = \frac{\sum_{i=1}^{n}\left|w_{ij}\right| + \sum_{j=1}^{m}\left|w_{jk}\right|}{n+m} \tag{6}$$

Fitness function is a trade-off among the conditions introduced above, as shown in Equation (7), where α, β and γ are the learning rates. Fitness function plays the most important role in measuring the adaptability of network and will help to prune the neural network to appropriate network size. Supervised by this fitness function, the network will be minimized as much as possible, and the network after pruning will make the best use of the remaining connections for the best adaptability.

$$F = \alpha A + \beta S + \gamma E \tag{7}$$

3 Experimental Results

The performance of the proposed BDNN-sp algorithm, especially the adaptability, will be verified and tested on different complexities of the Mixed National Institute of Science and Technology classification datasets (MNIST). To verify the adaptability of our BDNN-sp algorithm, we test it on different complexities of tasks. For example, different number of training samples or different groups of classes are tested. Besides, we also compare the transfer learning ability of pruned network with un-pruned network on another different complexities of tasks. The activation function of neuron is sigmoid function, and the learning rate η is equal to 1. Here, we set $\alpha = \beta = \gamma = \frac{1}{3}$.

The MNIST dataset contains 10 classes of hand written digits from 0 to 9. The total number of images contains 60,000 training samples and 10,000 test samples. Each image is represented by a $28 * 28$ length of vector. As a result, the initial ANN is with 784 neurons in its input layer, and the number of neurons in output layer is equal to the class number. We give 1,000 neurons in the hidden layer at the beginning of the ANN training.

Here we test 600, 1200, 2400, 4800, 9600, 19200 training samples with 10 classes. Then we calculate the accuracy A, the stability S, the mean of weights E and the fitness F as shown in Fig. 3.

As Fig. 3(A) shows, the classification accuracy on different number of training samples are changing with the iterative hidden neuron pruning process. Accuracy represents the test accuracy with the 10,000 test samples for testing. From the Fig. 3(A), when the network is too small (i.e. 0 to 50 neurons are left) or too big (400 to 500 neurons are left), underfitting or overfitting occurred and the accuracy is low. The range of acceptable network size with proper accuracy is different from different number of training samples. As shown in Fig. 3(B), the performance of stability has clear peak which is corresponding to the network size with the best stability, and underfitting and overfitting are corresponding to the left and right side of the peak respectively. Fig. 3(C) shows the change of means of network weights. It falls sharply at first and then keeps steady with the pruning of the network, and with more number of training samples, the higher mean of weights will be. The overfitting occurred when the network is too large and the mean of weights is small. Fig. 3(D) shows the change of fitness F, it has clear peak and the most appropriate network size has the best fitness. The more training samples are involved, the larger size of the network will be required.

Then the most appropriate network size on different number of training samples will be obtained. Fig. 4(A) depicts the number of neurons in hidden layers after neural network pruning. With the increase of training samples, the number of neurons in hidden layer increases. The red line in Fig. 4(A) represents the training samples in 10 classes, and the black line represents the training samples in 5 classes. From the result, we could see that the network sizes of 5 classes are smaller than the ones in 10 classes which is consistent with the prediction. The result indicates that the pruned network complexity will increase along with the increase of task's complexity. In all, the model could automatically modulate network size for different complexities of tasks through the BDNN-sp algorithm.

Besides, we also test the transfer learning ability on another different complexities of tasks. We first train the initial network and pruned the network on 9,600 training samples. Then, we take out the pruned network and un-pruned network which both have been trained to the same accuracy 88.72%. Finally, we test pruned network and un-pruned network on untrained 1,200 and 19,200 samples. Fig. 4(B) depicts the change of error on 19,200 samples, and Fig. 4(C) depicts the change of error on 1,200 samples. The black line in Fig. 4(B) and Fig. 4(C) represents the error of pruned network, and the red line represents the error of initial un-pruned network. Obviously, pruned network performs better than un-pruned network on both simple and complex tasks. Thus we can conclude that the pruned network has better transfer learning ability than un-pruned network.

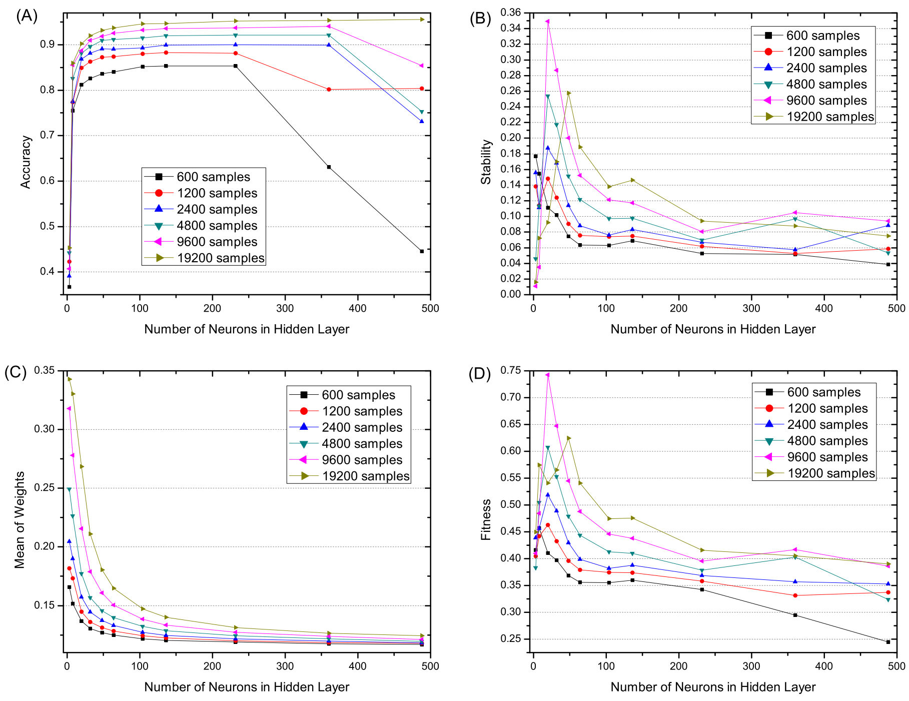

Fig. 3: The accuracy**(A)**, the stability**(B)**, the means of weights**(C)** and the fitness**(D)** of BDNN-sp on different number of MNIST training samples. The x axis shows the remaining number of neurons after pruning.

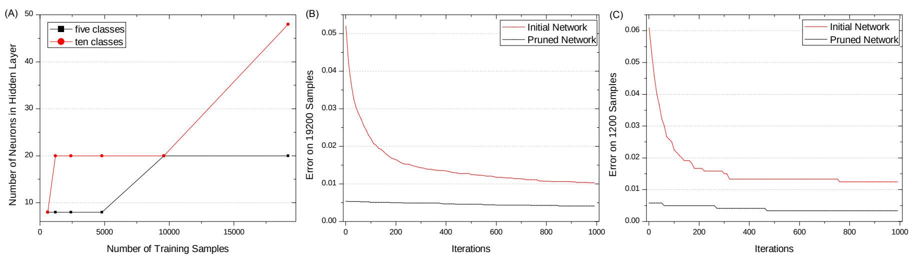

Fig. 4: **(A)**: The final network size on different complexities of tasks. **(B)**: The change of error on 19,200 samples during iteration. **(C)**: The change of error on 1,200 samples during iteration.

4 Conclusion

This paper proposes a neuron pruning method to adaptively modulate the topology structure of neural networks. Pruning process iteratively eliminates unimportant neurons and retrains the remaining network until its fitness reaches the peak. In order to determine the most appropriate network topology, we propose a new BDNN-sp algorithm with fitness function which integrates classification performance, stability and mean of weights. The experimental results show that the BDNN-sp model could reflect the network states of overfitting and underfitting. In addition, to verify the adaptability of BDNN-sp, we test it on different complexities of MNIST classification tasks. The experimental results show that the network complexity increases along with the increase of the task complexity. By comparing with the initial network, the size of network could be greatly reduced while the accuracy is with little reduction. On the transfer learning tasks, the BDNN-sp model performs better than un-pruned network.

Acknowledgment

This study was funded by the Strategic Priority Research Program of the Chinese Academy of Sciences (XDB02060007), and Beijing Municipal Commission of Science and Technology (Z161100000216124).

References

1. Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.
2. Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
3. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.
4. Steve Lawrence, C. Lee Giles1y, Ah Chung Tsoi, and Andrew D. Back. Face recognition: a convolutional neural network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997.
5. Li Deng, Geoffrey Hinton, and Brian Kingsbury. 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*, pages 8599–8603, 2013.
6. Zhi-Hua Zhou and Ji Feng. Deep forest: Towards an alternative to deep neural networks. 2017.
7. Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
8. Peter J. Angeline, Gregory M. Saunders, and Jordan B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65, 1994.

9. Xin Yao and Yong Liu. Evolving artificial neural networks through evolutionary programming. In *Proceedings of the 5th Annual Conference on Evolutionary Programming*, pages 257–266, 1996.
10. Joseph C. Park and Salahalddin T. Abusalah. Maximum entropy: A special case of minimum cross-entropy applied to nonlinear estimation by an artificial neural network. *Complex Systems*, 11, 1997.
11. E Vonk, L. C Jain, and R Johnson. Using genetic algorithms with grammar encoding to generate neural networks. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1928–1931, 1995.
12. Ileană Ioan, Corina Rotar, and Arpad Incze. The optimization of feed forward neural networks structure using genetic algorithms. In *Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics (ICTAMI)*, volume 8, pages 223–234, 2004.
13. Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Synaptic pruning in development: A computational account. *Neural Computation*, 10(7):1759–1777, 1998.
14. Michael Van Doren Johnston, Akira Ishida, Wako Nakajima Ishida, Hiroko Baber Matsushita, Akira Nishimura, and Masahiro Tsuji. Plasticity and injury in the developing brain. *Brain & Development*, 31(1):1–10, 2009.
15. Alvaro Pascual-Leone, Amir Amedi, Felipe Fregni, and Lotfi B. Merabet. The plastic human brain cortex. *Annual Review of Neuroscience*, 28(28):377–401, 2005.
16. Akiko Hayashi-Takagi, Sho Yagishita, Mayumi Nakamura, Fukutoshi Shirai, Y-i Wu, Amanda L. Loshbaugh, Brian Kuhlman, Klaus M. Hahn, and Haruo Kasai1. Labelling and optical erasure of synaptic memory traces in the motor cortex. *Nature*, 525(7569):333–338, 2015.
17. Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Neuronal regulation: A biologically plausible mechanism for efficient synaptic pruning in development. *Neurocomputing*, 26-27(98):633–639, 1999.
18. Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Synaptic pruning in development: a novel account in neural terms. In *Proceedings of Conference on Computational Neuroscience : Trends in Research*, pages 149–154, 1998.
19. Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.
20. C. E. Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 30:50–64, 1951.