Partial Restart of Neural Network

Bing Han(516030910523) and Yinqing Zhang(516030910560)
Shanghai Jiao Tong University

1. INTRODUCTION

Nowadays, the amount of parameters in neural networks are getting larger and larger. People use deeper and larger neural network to deal with the task of big data. And it also puts high demands on computing performance.

Mobile device, such as smart phone, is becoming more and more important in people's lives, but the computing performance of the mobile doesn't meet the needs of deep neuron networks. Considering that networks has been proved to have large sparsity. People try to change the neuron networks structure to reduce the scale of the neural network.

The most common method is pruning neuron networks to reduce a very large network to little without losing the accuracy of the task. Although pruning the networks can delete so many parameters in neural network, the computation speed is not accelerate so much as the irregular network. It is hard to accelerate computation on GPUs. In this paper, we proposal a new method to improve the CNN capacity to increase its accuracy in task without changing the structure of networks.

Using this method, we can train a smaller network and restart it to reach the same level of accuracy as a larger network. The smaller network requires less computation and can be deployed on devices with lower computing power to achieve better results.

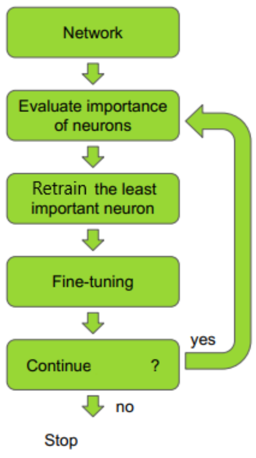
Network
Evaluate importance of neurons
Retrain the least important neuron
Fine-tuning
Continue ?
yes
no
Stop

Fig. 1. Process

2. BACKGROUND AND RELATED WORK

Appear and popularity of the Internet has brought a lot of information to users, and satisfaction their demand of information in the information age. Thanks to these big data, machine learning can develop more quickly and achieve good results on more tasks. To deal with larger and larger scale of data, people deepen the network, add more the parameters to improve the network's capacity to store more information.

Deep neuron networks require high computation ability when deployed in devices. In the past few years, people have put deep neural networks on the server to get better results. However, with the popularity of mobile terminals, the demand for neural networks on the mobile side is also increasing. High requirements of computation is an obstacle for deep neuron network to run in mobile device(like smart phone).

In order to better run the neural networks on the mobile device, people proposal some special networks(like MobileNet, ResNet, et.). These networks have special structure which is different from normal networks. And it's hard for users to design the scale when using these network. Though it can speedup the computing process and get a better performance, it's unsuitable to use it in some conditions.

Researchers proved that neuron networks have very large sparsity. It's said that so many parts in network don't work on fixed task though the network have so many parameters and layers. Then, people try a new solution which is pruning networks and many researchers have many contributes in this field.

Neural network pruning was pioneered in the early development of neural networks(Reed, 1993). Optimal Brain Demage(LeCun et al.,1990) and Optimal Brain Surgeon(Hassibi abd Stork, 1993)leverage a second-order Taylor expansion to select parameters for deletion, using pruning as regularization to improve training and generalization.

This method requires computation of the Hessian matrix partially or completely, which adds memory and computation costs to standard fine-tuning. In some paper, people said that good pruning can even improve the performance of networks. Though the pruning the scale of network can speedup the process training. But after training, the network have abnormal size, and it cannot speedup too much during training.

So we considering that why not reuse the sparsity in the network. Find these sparsity, restart and retrain it to improve the capacity of network. Finally, improve the performance of little scale network.

Convolution neural networks (CNN) are used extensively in computer vision applications, including object classification and localization, pedestrian and car detection, and video classification.

3. MOTIVATION

Thanks to the digitization of society, people's lives have generated a lot of data. Some of the data is beyond the fitting ability of traditional learning algorithms. As a result, people are focusing on the deep neural networks (DNNs). Given more data, we only need a deeper network. However, for DNNs, high demand for computation resources severely hinders deployment of large-scale DNN in resource constrained devices.

Recently,researchers have made effort to prune the large-scale neural network into a relatively small and compact neural network because of its sparsity. Inspirited by their great works, we consider they may help us to reset the pruned parts as the useless parts. If it works, there is no need for us to train a relatively large-scale network at the beginning and then prune it for compact. In this case, we only need to train the model as a small-scale network at the very beginning.

4. PROBLEM FORMULATION

Does the method of restarting neural network increase the accuracy of the network on image classification task and therefore generate a compact model with higher accuracy on the test set?

5. PROPOSED METHODS

The proposed method for restarting network consists of the following steps:

(1) Fine-tune the network until convergence on the target task.
(2) Alternate iterations of restarting and further fine-tune.
(3) Stop restarting after reaching the target performance.

```
Initialize the $model$;
$filters \leftarrow None$;
while Training do
    while $loss$ $decreases$ $in$ $recent$ $several$ $epochs$ do
        Train the $model$;
        $uselessfilters$ = the useless filters of $model$;
        if $uselessfilters == filters$ then
            $break$;
        end
        $filters \leftarrow uselessfilters$;
        Reset the weights of $uselessfilters$;
    end
end
```

Algorithm 1: Restart Neural Network

5.1 Fine-tune the network

Fine-tune the network is easy for everyone. Thus we will explain it briefly. After obtaining the data-set, pre-process the data-set so that it can be used in the network. Then construct the network structure and start to train it. At first, a larger learning rate for convergent quickly. Then lower the learning rate when training the network until convergence on the target task. And we need record the performance of original network. Because this score need to be compared with the score after restarting. At last, this step is done.

5.2 Locate the useless filters

In this step, the algorithm I use is mainly from Pruning Convolution Neural Networks For Resource Efficient Inference which is published as a conference paper at ICLR-2017. The procedure is simple, but its success hinges on employing the right searching criterion.

5.2.1 *Searching Criterion.* Consider a set of training examples $D = \{X = \{x_0, x_1, ..., x_n\}, Y = \{y_0, y_1, ..., y_n\}\}$, where x and y represent an input and a target output, respectively. The network's parameters $W = \{(w_1^1, b_1^1), (w_1^2, b_1^2), ..., (w_L^{C_t}, b_L^{C_t})\}$ are optimized to minimize a cost value $C(D|W)$. The most common choice for a cost function C(*) is a negative log-likelihood function. During the selecting, we refine a subset of parameters which preserves the accuracy of the adapted network. This corresponds to a combinatorial optimization:

$$min_{W'} |C(D|W') - C(D|W)| \quad s.t. \|W'\|_0 \leq B \tag{1}$$

where the l_0 norm in $\|W'\|_0$ bounds the number of non-zero parameters B in W'. If we search one by one, it will cost too much time, and it seems a impossible task.

5.2.2 *Taylor Expansion.* This problem can be seen as an optimization problem, that is, trying to find a W with a finite number of non-zero elements to minimize the loss change.$h_i = 0$ represent the loss if removing this weight. $|\Delta C(h_i)| = |C(D, h_i = 0) - C(D, h_i)|$. Using 1 order Taylor expansion to approximate $\Delta C(h_i)$, remove the highest order term, and finally get :

$$\Theta_{TE}(z_l^{(k)}) = \left| \frac{1}{M} \sum_m \frac{\delta C}{\delta z_{l,m}^{(k)}} z_{l,m}^{(k)} \right| \tag{2}$$

where M is length of vectorized feature map. For a mini-bath with $T > 1$ examples, the criterion is computed for each example separately and averaged over T.

5.3 Reset the weights of useless filters

For an unimportant filters, how to make it become important is our question. So we design several types reset methods to reset the weights of filters which have been selected. In the experiment, we will also show the result of several types restart.

5.3.1 *Uniform Randomly Initial.* A new neuron network is just uniform randomly initially at first. So we design this method to make these filters just like a new network.

$$W \sim U(-0.05, 0.05) \tag{3}$$

5.3.2 *He Randomly Initial.* This method is also one types to initial a new neuron network. We add this method to compare with the uniform randomly initial method.

$$W \sim U(-\frac{\sqrt{6}}{2n_j}, \frac{\sqrt{6}}{2n_j}) \tag{4}$$

5.3.3 *Uniform Randomly Initial between min and max.* We consider that maybe the initial have some relationship to the origin weights, the result will be better. Then we add method to randomly select one between min and max. For the minimum and maximum value of this value, we sample every weight between them. Then initial the filters.

$$W \sim U(min, max) \tag{5}$$

5.3.4 *Initial depend on the distribution of filters.* For each filter, it all has its own weights distribution. Why not compute the distribution of each filter, and sample new weight from the distribution gotten before.

5.4 Retrain the Network to Fine Tune

After initial the weights in filters, then retrain the network to fine tune the network. In this step, the method is just same with train the network at the first. First, we set a higher learning rate. When convergent, lower it, until it convergent at all. At last, compare the result as the benchmark before. For each network, we will restart three times to check the performance of restarting.

6. EXPERIMENTS

6.1 Dataset and CNN

We tested our algorithms mainly for image classification on the cifar-10 dataset: The CIFAR-10 dataset includes 60000 colour images in 10 classes, with 6000 images per class. We divided them into 50000 training images and 10000 test images.

We validated the efficiency of our method on two kinds of CNNs: AlexNet and MiniNet on the PyTorch platform. And the MiniNet's architecture was a small CNN as follows:

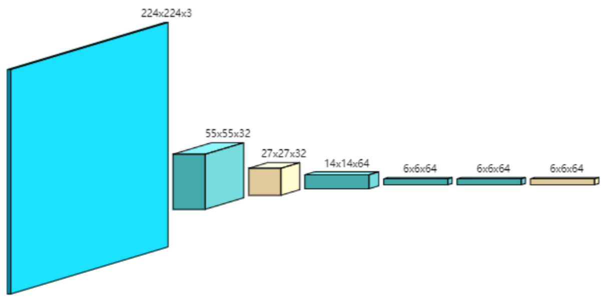

Fig. 2. MiniNet architecture

6.2 Fine-Tune Network

First we trained the networks. For each network, we trained it from higher learning rate to lower. Train them until they are convergent. The final accuracy on Cifar-10 has been shown in the table, and this accuracy will be regarded as baselines for next retraining procedure.

The final train accuracy of three types networks are shown in the following table. *This version of AlexNet was pre-trained on

Model	Accuracy
Pre-trained AlexNet*	90.45%
AlexNet	86.20%
MiniNet	75.66%

ImageNet.

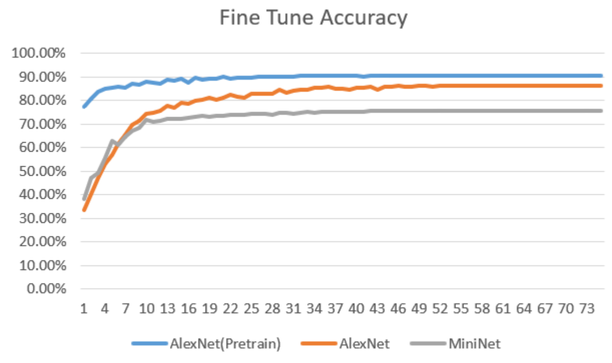

Fig. 3. Fine Tune Accuracy

6.3 Different Restart

After training the network for baseline, we would try to find which types of initialization has higher performance. We selected three types initialization methods and these methods have been introduced in the last section. For each network, we initialed them by three different methods. Train them 40 epoches, then compared them to see which is better. The result is as following:

	AlexNet(Pre-trained)	AlexNet	MiniNet
Uniform	91.60%	88.40%	79.66%
Xavier	88.72%	86.19%	76.76%
Uniform Between MinMax	74.59%	80.86%	74.88%

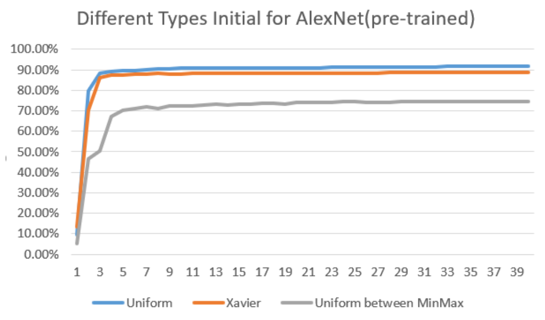

Fig. 4. Fine Tune Accuracy

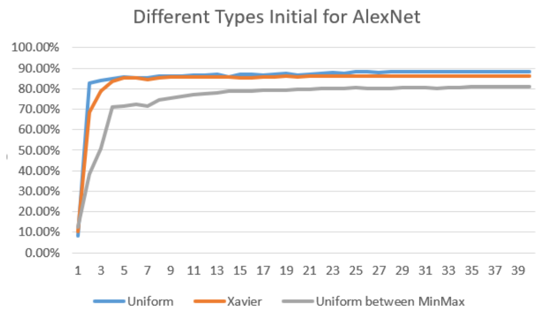

Fig. 5. Fine Tune Accuracy

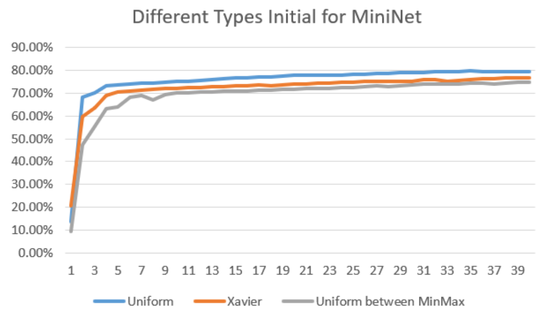

Fig. 6. Fine Tune Accuracy

6.4 Restart

Before, we selected the uniform initialization method as the best one. Then we will restart each network three times. And record its final accuracy as the result for analyzing.

	AlexNet(Pre-trained)	AlexNet	MiniNet
1	90.64%	88.09%	74.20%
2	**91.60%**	**88.41%**	75.79%
3	89.25%	88.35%	**79.66%**
Baseline	90.45%	86.20%	75.66%

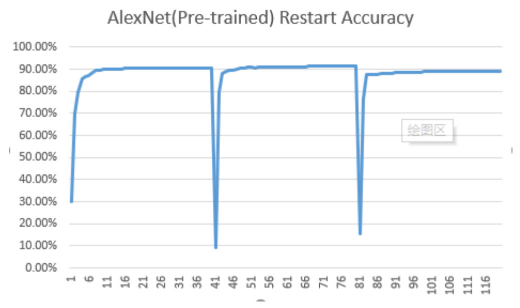

Fig. 7. Restart Three Times Accuracy for AlexNet(Pre-trained)

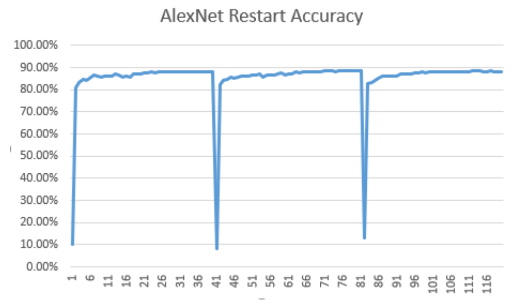

Fig. 8. Restart Three Times Accuracy for AlexNet

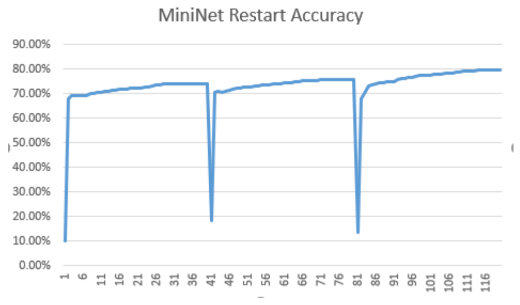

Fig. 9. Restart Three Times Accuracy for MiniNet

7. CONCLUSION AND DISCUSS

In this section, we discuss the result of experiments and analyze the result to obtain some conclusions.

7.1 Fine Tune Network

In the experiments of fine tuning, we trained three types of networks. Comparing the normal AlexNet with the AlexNet pretrained on ImageNet, pretrained AlexNet performe better thdan normal AlexNet.Obviously, the pretrained one carries some features of ImageNet, so it would be better.

As to MiniNet, because of its small-scale, it was trained much more quickly than AlexNets, but performed worst.

7.2 The Effectiveness of Different Methods of Initialization

We have done experiments to judge which one is best method when initialing the filters. For each network, we tested three methods to check the result. From the results, we figured out that the network initialized by randomly uniform converges more quickly than other two methods. When we train them for 40 epochs, the uniform method has converged sooner and its performance is better than other two methods.

As a result, we select the uniform initialization method for our restarting procedure.

7.3 Restarting

We restarted all networks and fined tune them for 3 times. From the table, we can see that the most of the restarted networks have reached the accuracy beyond the benchmark, which means that restarting is not always successful.

After restarting, the normal AlexNet increased accuracy more than pretrained Alexnet, which suggests that the restart method may help networks to reach its limitation.

Comparing the normal AlexNet and MiniNet, MiniNet's accuracy increased more than AlexNet. These two networks were not loaded pretrained weights, so the only difference between them is their scales, which means restarting on small-scale networks could gain more improvement on performance.

7.4 Discuss

Although we have significant results on most networks, the method may not work well on some networks, especially the large-scale networks. Moreover, on different stages of restarting, the performance may vary.

As a result, we consider that our method works like the restart-hill-climbing algorithm - when we change the value of some dimensions, the loss may jump out of the ridge, so we have more possibility to reach the better performance, and may reach it faster.

8. FUTURE WORK

The efficiency and effectiveness of our method is highly relative to the way of detecting useless neurons and filters. So that some effective pruning methods and researches of judging the uselessness of neurons or filters may bring inspiration to our work.

In our work, we created a universal function for restarting on any classic convolution layer. However, it may fail in some layers on special layers, such as residual blocks of ResNet. We will make an effort for restarting any layers with part of traditional architecture.

APPENDIX

REFERENCES

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014; arXiv:1409.1556.

Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, 2016; arXiv:1602.07360.

Alex Krizhevsky and Sutskever, Ilya and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks, 2012; NIPS2012.4824.

He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, 2015; ICCV2015.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila and Jan Kautz. Pruning Convolutional Neural Networks for Resource Efficient Inference, 2016; arXiv:1611.06440.

Steve Branson, Grant Van Horn, Serge Belongie and Pietro Perona. Bird Species Categorization Using Pose Normalized Deep Convolutional Nets, 2014; arXiv:1406.2952.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014; arXiv:1409.0575.

Song Han, Jeff Pool, John Tran and William J. Dally. Learning both Weights and Connections for Efficient Neural Networks, 2015; arXiv:1506.02626.

Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang and Dongjun Shin. Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications, 2015; arXiv:1511.06530.

Hengyuan Hu, Rui Peng, Yu-Wing Tai and Chi-Keung Tang. Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures, 2016; arXiv:1607.03250.

Sajid Anwar, Kyuyeon Hwang and Wonyong Sung. Structured Pruning of Deep Convolutional Neural Networks, 2015; arXiv:1512.08571.