Network Simplification: Dividing and Deleting Filters

Lu Guandong and Liu sichen
Shanghai Jiao Tong University

1. ABSTRACT

This passage is about an algorithm developed by us, motivated by papers submitted in recent years. Our algorithm is to simplify CNN using the strategy of deleting kernels. Before deleting the kernels, the classes are assembled into large classes. We develop a smaller network to determine the large class reducing the full connected layer from 4096 to 1024, and train it again. For each large class, delete the four-kernel sets whose removal causes no accuracy loss and get a new network for the large class. As a result, the problem is divided into two parts. To determine large class and to determine its actual class.

2. INTRODUCTION

Nowadays, image classification becomes a very popular area, since its the base of face recognition, environment recognition and automatic driving system. There are many methods developed to solve this kind of problem, such as KNN(K-Nearest Neighbors), linear classifier and so on. Unfortunately, some of the methods dont have a enough high accuracy. But so far, the best way to solve this kind of problem is using CNN(convolutional neural network). But we all know that the networks have a lot of parameters, and problems with small scale actually dont need such big networks, so if some of the parameters are deleted the calculation can be reduced to speed up and apply in real world. But if the network is modified, the accuracy changes. We have to trade off the simplification and accuracy.

3. BACKGROUND

There are many CNN models. We choose AlexNet[1] because its a basic and mature network considering our limited time and resources. AlexNet has five characteristics that previous networks dont have:

—1. Successful use of ReLU as an activation function of CNN solves the problem of gradient dispersion of Sigmoid when the network is deep.

—2. Use Dropout to randomly ignore a subset of neurons during training to avoid over-fitting the model, and in AlexNet, the last few fully connected layers use Dropout.

—3. In AlexNet, the step size is smaller than the size of the pooled core, so that there is overlap and coverage between the outputs of the pooled layer, which improves the richness of features.

—4. The LRN layer is proposed to create a competition mechanism for the activity of local neurons, so that the relatively large value of the response becomes relatively larger, and other neurons with less feedback are suppressed, and the generalization ability of the model is enhanced.

—5. Use CUDA to accelerate the training of deep convolutional networks, and utilize the powerful parallel computing power of GPU to deal with a large number of matrix operations during neural network training.

Fig. 1. The structure of AlexNet

4. RELATED WORKS

According to the paper Pruning Filters for Efficient Converts[3], for every kernel in CNN, it is assessed with the absolute sum of all weights of it. And the kernels with low score are deleted and the connections with them are also deleted.

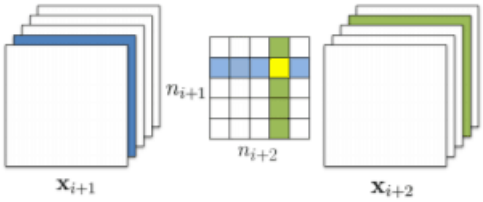

Fig. 2. Pruning filters across consecutive layers. The independent pruning strategy calculates the filter sum (columns marked in green) without considering feature maps removed in previous layer (shown in blue), so the kernel weights marked in yellow are still included. The greedy pruning strategy does not count kernels for the already pruned feature maps. Both approaches result in a $(n_{i+1}1) \times (n_{i+2}1)$ kernel matrix.

However, the accuracy of this pruning strategy doesn't maintain so well and the parameters don't decrease so significantly.

5. MOTIVATION

The paper Quantifying Interpretability of Deep Visual Representations motivates us with the interpretability[2] of the kernels in CNN, and it quantifies the interpretability of a kernel to a concept/class using a self-designed equation. But in our work, the interpretability is quantified using the accuracy change deleting a four-kernel set, and whether the sets are deleted depends on the value calculated previously.

Besides, in the paper mentioned in last section, the strategy can be adopted, we can perform operations in that order. But the definition of the score we give to every kernel is different. We use the accuracy change as the score, and smaller score represents better kernels. And kernels whose score is more than a threshold will be deleted.

6. PROBLEM FORMULATION

6.1 Input/output sequence and problem

—Common CNNs are large, but problems with small scale can use small networks as long as guarantees the accuracy to simplify the process.

—The input is a network and the output is a simplified network based on a given data base.

6.2 Our assumptions

—Problem of small scale can be solved using smaller networks

—Different filters have different responses to a class

6.3 Targets

—To simplify the CNN deleting kernels while guarantee the accuracy not decrease much.

6.4 Observed properties and conclusions

—The interpretability of CNN kernels

—The number of parameters proportional to the number of kernels

6.5 Methods

—Generally, we apply the idea of divide and conquer, dividing the problem into two subproblems and solving them successively. Assemble the small classes into some large classes, and try to use a smaller network to determine the large class. Besides, for every large class, deleting the kernels not affecting the accuracy after deleted. In each subproblem, the network is smaller than previous one, and in toal, the scale is also decreased.

7. PROPOSED METHODS

We proposed a network simplification method which we called Layered-Network-Simplification. We divide this part into three parts. First we introduce our process to the previous AlexNet kernels, then we introduce an algorithm to decide how we take superclasses and what kernels we want to reserve(and other kernels are to be deleted), and last we talk about the train and evaluate process.

7.1 Kernel evaluation

To evaluate how the kernels are effected the result, we use the following method to calculate every result y_i which means how important the i_{th} kernel is.

We define the kernels response to a concrete output class as the kernel can determine whether the kernel can recognize the input is in this class correctly, which can be defined as follows:

$$\hat{y_i} = y_i = m \quad or \quad \hat{y_i} = y_i \neq m \tag{1}$$

Here m means the output class we are examining. Since we have the evaluation methods, we can simply delete every kernel and examine whether the accuracy of every class dropped or not.

$$\delta_1 = p(\hat{y_{ij}} = y_i = m \quad or \quad \hat{y_{ij}} = y_i \neq m) \tag{2}$$

$$\delta_2 = p(\hat{y_i} = y_i = m \quad or \quad \hat{y_i} = y_i \neq m) \tag{3}$$

$$\Delta = \delta_1 - \delta_2 \tag{4}$$

Here $\hat{y_{ij}}$ means the output of the network after deleting the j^{th} kernel in the AlexNet. Due to limited time, we only examine the last layer(the 5^{th} layer) kernel.

The Δ references the response of each kernel to a certain output class. If one kernel is very important to one result, the Δ should be very large, as removing it will decrease the accuracy greatly. On the contrary, if a kernel is somehow useless to the class, the Δ will not decrease much. Actually, in our experiment, deleting some of the kernels will increase the accuracy, or what we called response, to the classes.

Due to the limited calculating resources, we grouped 4 kernels as a batch and examine batches response. The result shows in the Fig.3:

In conv5, there are total 256 kernels, and we take 4 for 1 batch and get the above 64 batches response.

7.2 Merge algorithm

In previous work, for every class, we can get a vector, indicating the useful and useless 4-kernel sets. Obviously, the vector is of 1 dimension and its length is 64. And the vector has only 1 and 0 values, with 1 indicating kernel response, while 0 representing kernel no response. And the aim of the algorithm is assemble them into 3 groups, meaning 3 superclasses.

And we'd like to introduce some concepts defined by us. vector A includes vector B means all the digs of A are greater or equal than those of B. The union of two vectors is the result of the bit operation 'or' between two vectors. Union penalty between vector A and B is the subtraction between the number of ones in union of A and B and the larger value of the number of ones in A or B.

And the group requirement is to make the union of every group of vectors have the smallest number of ones possible and try the best to make the number of vectors in every group close. The optimal solution is nonexistent or difficult to reach, so we design an algorithm to find a approximately optimal one.

First sort the vectors in ascending order of number of ones, and calculate the total penalty between two vectors. Here, we define merge penalty as:

$$mergePenalty = unionPenalty + distancePenalty \tag{5}$$

Distance penalty is the difference between number of ones of the two vector. Here we add a distance penalty because if there is a vector with all digits 1, then the union penalty is 0, and all the vectors will be grouped with it.

First build a dictionary, key and value are both a vector. Then, calculate the merge penalty for all key pairs, and find the minimum. Then merge the two keys. Build a new key, whose value is the union of the two current keys, and its value is a set which is a union set of the two current value sets. Perform the operation until the number of key value-pairs equals to the number of superclasses. And in our experiment, there are 3 superclasses.

And three remaining keys are three vectors, and these vectors implies the kernels deleted and conserved in every network.

7.3 Retraining

After we get the superclasses, we delete the network kernels and reserve only the kernels we get in the superclass algorithm. Then we retraining every mini network with the training data, the label of which in under the same superclass.

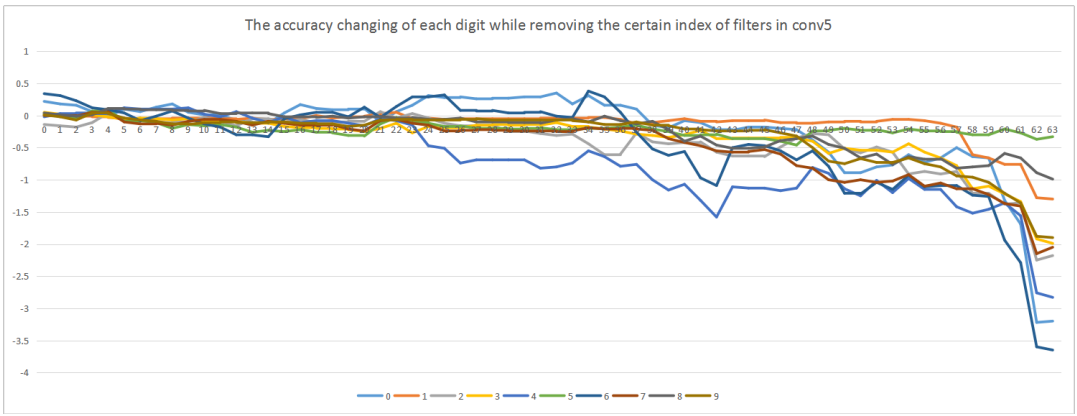

Fig. 3. Accuracy change after deleted

Then we train the first layer network with a smaller size network, whose FC reduced from 2048*2048 to 1024*1024. Here I use a convert dictionary to convert the original label to the superclass label.

8. EXPERIMENT

In the experiment, we use MNIST[4] as our dataset, and as they have 10 classes, we will merge them into 3 superclasses, which have 3, 3, 4 classes respectively.

8.1 Training

The training and retraining loss graph shows below, and the learning rate is 0.01.

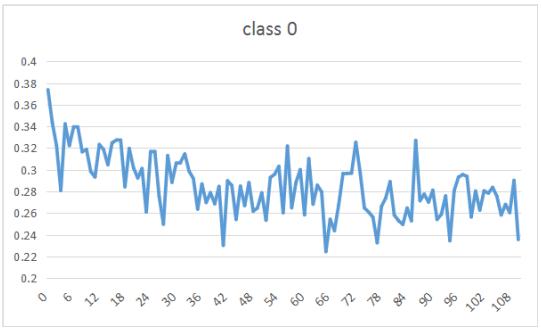

Fig. 4. loss change in class 1

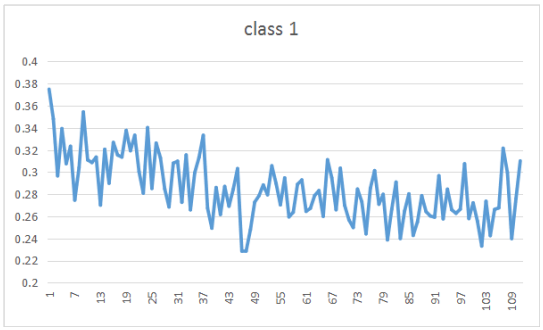

Fig. 5. loss change in class 2

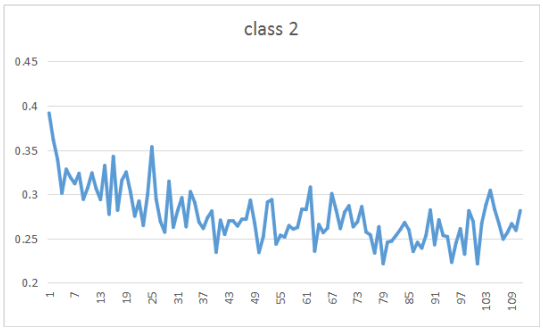

Fig. 6. loss change in class 3

8.2 Evaluation

Since there is no specific implementation of AlexNet using MNIST, we implemented by ourselves.

The accuracy rate of every network in the experiment is shown in the table. From the table, we can get that in our layered simplification, not only the number of parameters is decreased to about 40 percent, but the accuracy also has a slight ascension.

Network	Parameters	Accuracy(Top-1)
Alexnet[1](benchmark)	60M	87.53
Simplified Alexnet	24M	76.47
Simplified Alexnet + retraining	**24M**	**88.97**

To evaluate whether the algorithm developed by us is good, we calculate the accuracy for every subclass in the three networks.

Because the accuracy of the pure AlexNet is 87.53, and the accuracy of each superclass is about 90 percent, which is great than the benchmark. And this means the division is a good division.

9. CONCLUSION

In this project, we proposed a network simplification method based on AlexNet. We use two small network to replace a large network in order to decrease the total parameters of the network. In the ex-

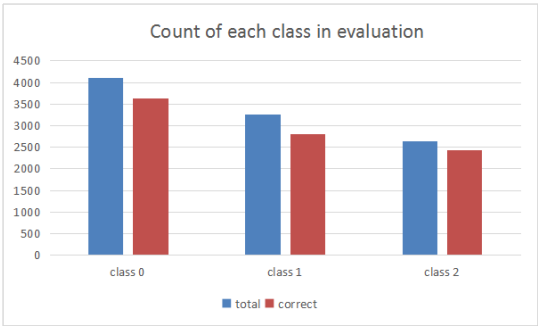

Fig. 7. count of every subclass

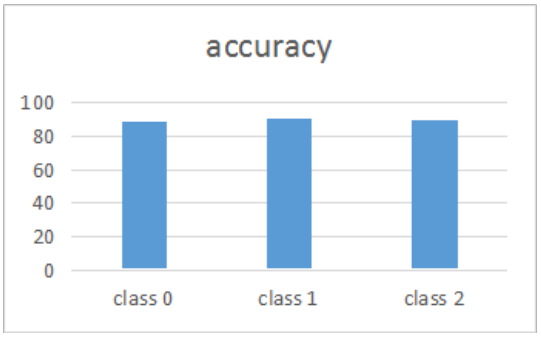

Fig. 8. the accuracy of every subclass

periment, we even managed to increase the accuracy of the network a bit by retraining the simplified network.

In our opinion, this is a new idea of network simplification and it can use in the further study. Due to the time limitation, we only tried on MNIST and AlexNet. Other networks can also be simplified using this method.

We hope our algorithm can be used as a new approach to simplify the network.

10. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton:ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

[2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba Network Dissection: Quantifying Interpretability of Deep Visual Representations, CVPR2017

[3] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, Hans Peter Graf:PRUNING FILTERS FOR EFFICIENT CONVNETS, ICLR2017

[4] Yann LeCun, Corinna Cortes, Christopher J.C. Burges: THE MNIST DATABASE of handwritten digits