Very Deep Convolutional Neural Network Based Image Classification Using Small Training Sample Size

Shuying Liu, Weihong Deng
Beijing University of Posts and Telecommunications, Beijing, China
liushuying668@163.com, whdeng@bupt.edu.cn

Abstract

Since Krizhevsky won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 competition with the brilliant deep convolutional neural networks(D-CNNs), researchers have designed lots of D-CNNs. However, almost all the existing very deep convolutional neural networks are trained on the giant ImageNet datasets. Small datasets like CIFAR-10 has rarely taken advantage of the power of depth since deep models are easy to overfit. In this paper, we proposed a modified VGG-16 network and used this model to fit CIFAR-10. By adding stronger regularizer and using Batch Normalization, we achieved 8.45% error rate on CIFAR-10 without severe overfitting. Our results show that the very deep CNN can be used to fit small datasets with simple and proper modifications and don't need to re-design specific small networks. We believe that if a model is strong enough to fit a large dataset, it can also fit a small one.

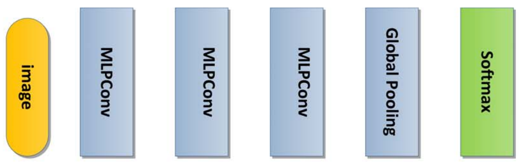
1. Introduction

Depth matters. CNNs were heavily used in 1990s(eg., [11]), but fade out of sight soon since the rise of support vector machine. Thanks to the rapid development of computing capacity, especially the development of GPU, in 2012, Krizhevsky won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 competition[1, 2] with the brilliant deep convolutional neural networks(D-CNNs)[10], which shows the great power of depth. The model used by Krizhevsky was trained on 1.2 million labeled images, together with 8 weight layers(5 convolution layers and 3 fully-connected layers) and ReLU nonlinearities and "dropout" regularization. Since D-CNNs' great success in 2012, researchers have proposed various kinds of D-CNNs. Zeiler and Fergus visualized the D-CNN with deconvolution network[17], and proposed a modified AlexNet according to the visualization results, which outperform AlexNet on the ILSVRC 2012. However, Oxford's 16-layer and 19-layer model[14] and the even deeper GoogleNet[15] confirmed that depth is the most critical factor which leads to the high performance of D-CNNs.

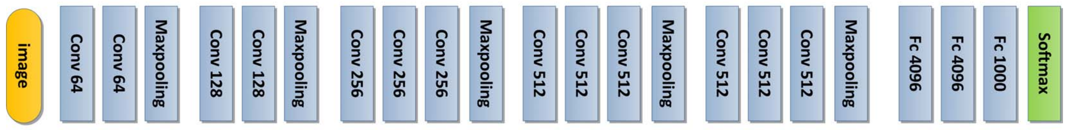
However, almost all the very deep convolutional neural networks were trained on the giant ImageNet datasets. In fact, we are lack of labeled data and there is only one ImageNet datasets in the world. When people want to train a CNN on small datasets like CIFAR-10[9], they always tend to use models that are much shallower than AlexNet. Krizhevsky used a 4-layer CNN[10] to fit the CIFAR-10 datasets and got 11% error rate. Lin proposed the novel Network In Network(NIN)[12] which contains 3 MLP convolution layers and 1 global pooling layer, and NIN got 8.81% error rate. Many researchers believe that deeper CNNs should only be used to fit large datasets and shallower CNNs are better to fit small datasets. Figure 1 gives an explanation of the way people think about the depth of CNN.

In 2014, Ross proposed that very deep models can be used when lack of labeled data with the help of finetune[3]. However, finetune needs a D-CNN model pre-trained on large datasets like ImageNet dataset. Meanwhile, if we don't finetune all the fully-connected layers, the small dataset which needs to be finetuned should meet the require of the pre-trained model, say, it should be able to give the same size input of the deep model, or the number of weights in the fully-connected layers might miss match. For CIFAR-10, it's not good to use finetune. Because resizing 32x32 images to 227x227 would not increase any information and would make the input extremely blur. However, now we provide another way to take full advantage of deep model.

In this paper, we modified the very deep VGG-16 network and use it to fit the small CIFAR-10 datasets. In order to make the very deep model work for the small datasets, we made the following modifications: a. shrink the dimension of fully-connected layers to reduce the number of parameters; b. use the brilliant batch normalization[6] to accelerate convergence, reduce error rate and overfitting; c. use more "dropout" and stronger weight decay to reduce overfitting. Our model avoid severe overfitting on the small CIFAR-10 dataset and got 8.45% error rate. Although we



(a)



(b)

Figure 1. Two famous CNNs used to fit small datasets and big datasets, respectively. (a) Network in Network. Used to fit CIFAR-10 and CIFAR-100 (b)VGG-net. Used to fit ImageNet. People always think that very deep network should only used to fit giant dataset like ImageNet.

didn't achieve state-of-the-art performance on CIFAR-10, our goal is to show that a very deep convolutional neural network can be used to fit small dataset like CIFAR-10 with simple but proper modifications, small datasets can also enjoy the power of very deep model. Once we've designed a powerful D-CNN for a large dataset, we don't need to redesign a specific small CNN for a small dataset.

The remainder of this paper is organized as follows. In Section 2, we introduce our modified model's network configuration and some methods to adopt deep model from large dataset to small ones in detail. The details of training are discussed in Section 3. Experiments results and analysis are given in Section 4. And finally we draw our conclusions in Section 5.

1.1. The Very Deep Network

We use a very deep CNN to fit CIFAR-10 dataset. Below, we describe the overall architecture and some features of the model. Section 3.2-3.6 are sorted according to the importance we observed from our experiment results.

1.2. Overall Architecture

The model is a modified version of VGG net(Figure 1.(b)). We choose VGG net as the base net because the 32x32 input would not vanish through the VGG's architecture, and the VGG architecture is very simple so that we can analysis the result more easily. What's more, VGG net is one of the most powerful deep convolutional neural network. Result in Section 4 shows that this VGG-based deep model is much more stronger than a 4-layer CNN even in the task of classifying CIFAR-10.

The model consists of 5 group of convolution layers and 1 group of fully-connected layer , along with 13 convolution layers and 2 fully-connected layers. Every convolution filter has an kernel size of 3x3 with stride of 1 and pooling region of 2x2 without overlap. Note that we shrink the two 4096-dimension fully-connected layer to one 100-dimension fully-connected layer, which significantly reduce the number of parameters.

1.3. Batch Normalization

For very deep networks, small changes in the previous layers will amplify layer by layer and finally cause some problem. The change in the distribution of layers' inputs cause problem since the parameter should adapt to new distribution with iterations, which is called Internal Covariate Shift[6]. To solve this problem, Google's researcher Ioffe proposed Batch Normalization[6], which normalizes every layer's inputs, thus makes the network converge much faster, converge at lower error rate and reduce overfitting to some degree. Ioffe add Batch Normalization layers to GoogleNet and break the record of ImageNet classification task and surpass human ability. However, we believe that Batch Normalization can also optimize very deep model like VGG.

We add Batch Normalization layer before every nonlinearity. Since Ioffe didn't release code, we implement Batch Normalization layer in Caffe by ourself. However, our implementation is a bit different from the standard one. We do the same z-score like operation on the inputs of the Batch Normalization layer:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E(x^{(k)})}{\sqrt{Var(x^{(k)})}} \tag{1}$$

And then do linear operation on $\hat{x}^k$

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)} \tag{2}$$

where k indicates channel, which means all operations are channelwise.

However, we didn't implement Alg2, i.e. the inference stage. We use exponentially-weighted moving average mean and variance instead of the fixed ones, i.e.

$$Mean_{(t)} = \begin{cases} BatchMean_{(1)} & t = 1 \\ \alpha BatchMean_{(t)} + (1-\alpha) Mean_{(t-1)} & t > 1 \end{cases} \quad (3)$$

$$Var_{(t)} = \begin{cases} BatchVar_{(1)} & t = 1 \\ \alpha BatchVar_{(t)} + (1-\alpha) Var_{(t-1)} & t > 1 \end{cases} \quad (4)$$

where the subscript t indicate iterations.

Results proved that EWMA can also stabilize the test phase.

Results show that Batch Normalization significantly improve the converge speed. It can arouse the hidden power of very deep model with the help of dropout, which reduce the error rate from 20% to 8.45%.

1.4. Strong Dropout Setting

When we hope to adopt deep network to small dataset, the very first thing is to think about how to reduce overfitting. The most common method to reduce overfitting is dropout. The standard way of using dropout is to set dropout at the fully-connected layers where most parameters are. However, since the model is very deep and the dataset is relatively small, we use a different way of dropout setting. We set dropout for convolution layers, too. Specifically, we set dropout rate as 0.3 for the first and second group of convolution layers, 0.4 for the fourth group , 0.5 for the fifth group. The dropout rate for the fully-connected layers was set to be 0.5. Results show that the strong dropout setting significantly control the overfitting of the deep model. Along with the use of Batch Normalization, the hidden power of deep model was aroused, which reduce the error rate from 20% to 8.45%.

1.5. Accelerate

Although deep model can provide powerful representations, but training a deep model require a lot of computation, i.e. both forward pass and backward pass are slower than shallow models. We hope to accelerate training without losing too much power. Consider the very first two layers of 3x3 convolution filters with stride of one, which cost too much computation. Although two layers of 3x3 convolution filters can gain more non-linearity than one layer of 5x5 convolution layers but introduce a lot of computation without increase much information. So we replace the first group of 3x3 convolution layers with one layer of 5x5 convolution filters and add padding of 2. The modification raise the error rate from 8.45% to 9.20% and reduce the training time from 19 hour to 17 hour. And when we continue to replace the second group of 3x3 convolution layers with one layer of 5x5 convolution filters, the error rate rise from 9.20% to 11.0% and training time is reduced from 17 hour to 16.5 hour, this shows the use of dense convolution is more important than the previous use of dense convolution.

It's a speed-accuracy trade-off, and we didn't use such accelerating methods in our final model since we want higher accuracy.

1.6. Different Nonlinearity

The most important feature of AlexNet is ReLU nonlinearity, which shows the importance of nonlinearity. We tried three kinds of nonlinearity: ReLU, Leaky ReLU[13], PReLU[5]. For Leaky ReLU, we manually set the negative slope. Specifically, negative slopes were set as 0.3, 0.2 ,0.1 for the first group, the median two group and the last group of convolution layers, respectively. And negative slope was set as 0.05 for the fully-connected layers. Results show no obvious difference using different nonlinearitys. In fact, we might not have used the best negative slope setting for Leaky ReLU. But we believe that there isn't too much difference between these nonlinearitys since they are all non-saturate nonlinearitys.

2. Training

We trained 5 models on CIFAR-10 dataset for comparative experiments.

Baseline: The CIFAR10-full example in Caffe[7] was used as baseline. It's a 4-layer CNN with 3 convolution layers and 1 fully-connected layer. The training setting in the example was used without modification.

Baseline-BN: Add Batch Normalization layers before every nonlinearity in the Baseline model and use the same training setting as Baseline model except for the learning rate policy, which is tuned for better performance.

CIFAR-VGG-BN: Add Batch Normalization layers before every nonlinearity in the model described in Section 2.1. Use mini-batch sgd to solve the net, with batchsize of 100. The momentum, base learning rate and base weight decay rate are set to be 0.9, 0.001, 0.006, respectively. Lower down the learning rate every 20,000 iterations by a factor of 0.7.

CIFAR-VGG-DROPOUT: Use dropout setting described in Section 2.3 on the model in Section 2.1. The training setting is the same as CIFAR-VGG-BN model except for the learning rate, which is tuned to guarantee convergence.

CIFAR-VGG-BN-DROPOUT: Use both Batch Normalization and strong dropout setting on the model in Section 2.1 with the same training setting as CIFAR-VGG-BN.

Note that we do no data augmentation except for horizontal mirroring.

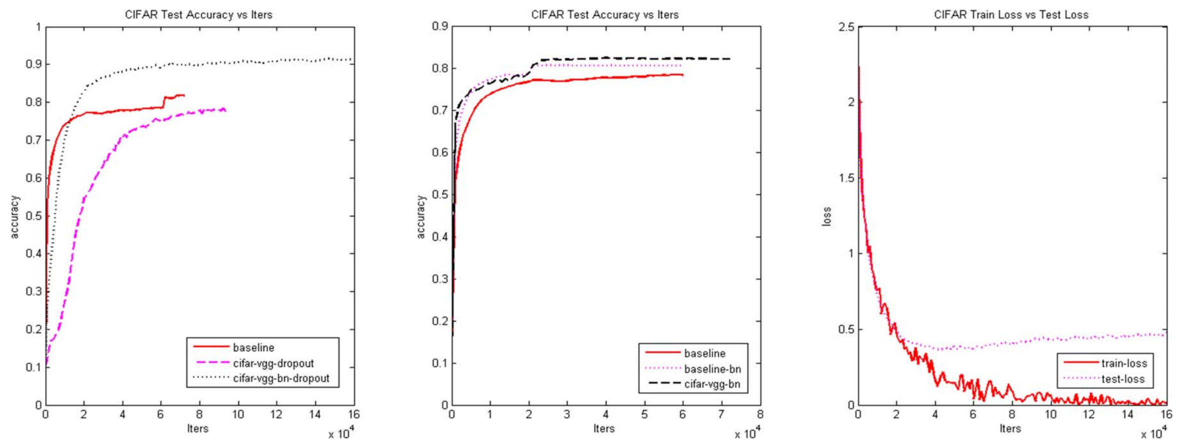


Figure 2. Some results of our comparative experiments. (left) Comparisions on accuracy. (middle) Comparisions on the speed of convergence. (right) The performance curve of the CIFAR-VGG-BN-DROPOUT model.

2.1. Result

We trained models on CIFAR-10 dataset. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Both the number and the size of images are very small, for a very deep CNN.

Figure 2 and Table 1 shows our experiment results.

We found that, when using only Batch Normalization or strong dropout setting on the very deep model, the model gains accuracy that was not very high. But the CIFAR-VGG-BN model converges pretty fast, even faster than the Baseline-BN model, and it gains higher accuracy than the two baseline models. This shows that when we need fast convergence without very high accuracy, very deep models are still better choice than the shallow ones.

When using both BN and strong dropout setting on the very deep model, the model gains very high accuracy and converges fast. In our opinion, depth offers opportunity to gain high accuracy, BN prevent the training procedure from stucking into poor local optima and strong dropout setting significantly controlled overfitting. This shows that small dataset can also utilize the power of depth.

We also found that the CIFAR-VGG-BN-dropout model show slightly overfitting, so we tried to strengthen the regularizer. However, doing so didn't raise accuracy but only reduce overfitting. We think the power of the model on this data has reached its limit and we believe that the potential of the model will be aroused further with data augmentation. However, our goal is to show that a very deep convolutional neural network can be used to fit small dataset like CIFAR-10 with simple but proper modifications, rather than beat the state-of-the-art.

Model	error rate
Baseline	18.20%
Baseline-BN	19%
CIFAR-VGG-BN	17.45%
CIFAR-VGG-DROPOUT	21.71%
CIFAR-VGG-BN-DROPOUT	8.45%
Network in Network	8.81%
DropConnect Network[4]	9.32%
Maxout Network[16]	9.35%
Human rater[8]	6%

Table 1. Some results of the comparative experiments. In *italic* are methods achieved by others.

2.2. Conclusion

As we proved, very deep models can also be used to fit small dataset as long as the input image is big enough so that it won't vanish as the model going deeper.

The key to adopt deep model to small dataset is to use Batch Normalization and strong dropout setting. when we need fast convergence without very high accuracy, we can use Batch Normalization on a very deep model and then the model will gain comparable accuracy with shallow models and converges faster. When we need high accuracy, we can use both Batch Normalization and strong dropout setting on a very deep model, we can gain high accuracy with relatively fast convergence.

However, we don't think Batch Normalization and strong dropout setting are the only way to arouse the power of deep model on small dataset. More methods will require future work.

3. Acknowledgements

This work was partially sponsored by supported by the NSFC (National Natural Science Foundation of China) under Grant No. 61375031, No. 61573068, No. 61471048, and No.61273217, the Fundamental Research Funds for the Central Universities under Grant No. 2014ZD03-01, This work was also supported by the Beijing Higher Education Young Elite Teacher Program, and the Program for New Century Excellent Talents in University.

References

[1] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, , and L. Fei-Fei. Imagenet large scale visual recognition competition 2012. http://www.image-net.org/challenges/LSVRC/2012/. ILSVRC2012. 1

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 1

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014. 1

[4] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *ICML*, 2013. 4

[5] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015. 3

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 1, 2

[7] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014. 3

[8] A. Karpathy. Lessons learned from manually classifying cifar-10. http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/. CIFAR-10. 4

[9] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 1(4):7, 2009. 1

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[12] M. Lin, Q. Chen, S. Yan, M. Lin, Q. Chen, and S. Yan. Network in network. *Network In Network - ResearchGate*, 2013. 1

[13] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013. 3

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 1

[16] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. *Proceedings of International Conference on Machine Learning*, 2013. 4

[17] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014. 1