Final Project: LeNet-5 Pruning

Chaoji Zuo chaoji.zuo@rutgers.edu
CZ296 / 190003416

Abstract

Deep Neural Network compressing is an advanced topic in deep learning. It is also essential for researchers who want to save the energy and deploy the deep neural network on embedded systems with limited hardware resources, because the original deep neural network models is intensive in storage and computation. Specifically, Pruning/Sparsity skills are one of the famous model compression techniques. Under an iterative procedure of pruning, we can generate sparse models which have similar performances with the original models but much less floating point operations and we can reduce the model sizes with the help of quantization or Huffman coding. In this project, I try to prune a LeNet-5 network on MNIST and conduct multiple experiment to analyze the model performance.

1. Introduction

In recent years, deep neural networks have received lots of attention, been applied to different applications. Especially, deep neural networks have evolved to the state-of-the-art technique for computer vision tasks like image classification, object detection, etc. Although these models are very competitive, they still rely on deep networks with millions of parameters, which consumes considerable storage, memory and computing power. For example, the AlexNet Caffemodel is over 200MB, and the VGG-16 Cafffemodel is over 500MB (Han et al. (2016)).

In order to achieving the goal of reducing computational time and parameters storage, the deep learning community had done several works about model compression and acceleration. And in the work from Cheng et al. (2017), they classify those approaches into four categories: parameter pruning and sharing, low-rank factorization, transferred/compact convolutional filters, and knowledge distillation. The parameter pruning and sharing based methods explore the redundancy in the model parameters and try to remove the redundant and uncritical ones.

In this project, I select the 7nd of available Type-B project: **Prune a LeNet-5 convolutional NN on MNIST dataset with high compression ratio and negligible performance loss**. Pruning skill is a common trick to reduce the computation in many transitional machine learning models like decision trees. Han et al. (2015) first introduced pruning skill into deep learning in this work. This work not just include pruning, the authors design a three stage pipeline: pruning, trained quantization and Huffman coding. They give a whole guide of deep learning network compression. Their experiment result showed their method reduced the storage required from 240MB to 6.9MB, with out loss of accuracy.

Theme Name	Description	Applications	Featuress
Parameter pruning and sharing	Reducing redundant parameters which are not sensitive to the performance	Convolution layer and fully connected layer	Robust to various settings, can achieve good performance, can support both train from scratch and pre-trained model
Low-rand factorization	Using matrix/tensor decomposition to estimate the informative parameters	Convolution layer and fully connected layer	Standardized pipeline, easily to be implemented, can support both train from scratch and pre-trained model
Transferred/compact convolutional filters	Designing special structural convolutional filters to save parameters	Convolution layer only	Algorithms are dependent on applications, usually achieve good performances, only support train from scratch
Knowledge distillation	Training a compact neural network with distilled knowledge of a large model	Convolution layer and fully connected layer	Model performances are sensitive to applications and network structure only support train from scratch

Table 1: Summarization of Model Compression Approaches

Considering the limitation of my computing equipment and complexity, I just follow the project topic requirement and try to prune a LeNet-5 network. LeNet-5 is a basic neural network model and it would be introduced more in section 5. I follow the guide from slide lec13 and Han et al. (2016) to achieve the pruning step and try to adjust the hyper-parameters to do comparison and analyses. My prune process achieve a high compression ratio and negligible performance loss model.

2. Related works

The works of deep learning models compression and acceleration can be classified into four categories: parameter pruning and sharing, low-rank factorization, transferred/compact convolutional filters, and knowledge distillation. The parameter pruning and sharing based methods explore the redundancy in the model parameters and try to remove the redundant and uncritical ones. Low-rank factorization based techniques use matrix/tensor decomposition to estimate the informative parameters of the deep CNNs. The approaches based on transferred/compact convolutional filters design special structural convolutional filters to reduce the parameter space and save storage/computation. The knowledge distillation methods learn a distilled model and train a more compact neural network to reproduce the output of a larger network.

3. Data Description

In my project, I try my models on the MNIST database of handwritten digits, which has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. This dataset is released by the founding father of convolution nets: Yann LeCun, Corinna Cortes and Cristopher J.C. Burges. It is a good database for people who want to

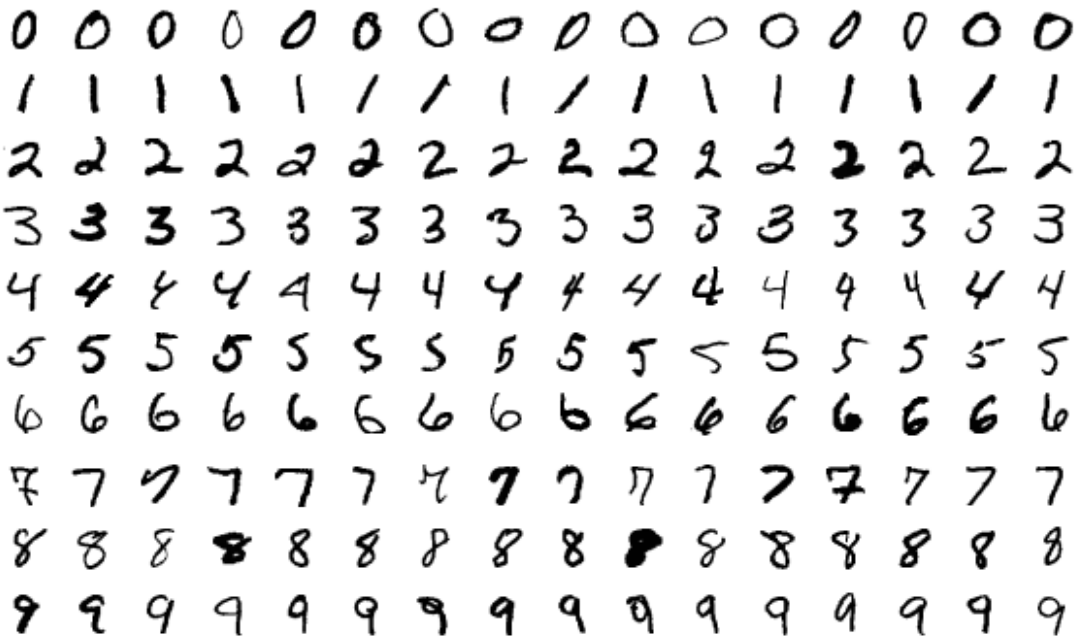

Figure 1: MNIST samples

try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

MNIST contains 70,000 gray-scale images of hand-drawn digits, each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a sigle pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive. Figure 1 shows some example digits in MNIST dataset.

4. Method Description

Network pruning has been used both to reduce network complexity and to reduce over-fitting. An early approach to pruning was biased weight decay. Optimal Brain Damage and Optimal Brain Surgeon prune networks to reduce the number of connections based on the Hessian of the loss function and suggest that such pruning is more accurate than magnitude-based pruning such as weight decay. However, second order derivative needs additional computation.

The phases of pruning and retraining may be repeated iteratively to further reduce network complexity. In effect, this training process learns the network connectivity in addition to the weights

In this project, I follow the process proposed in Han et al. (2015), which have a three-step process, this method starts with a connectivity learning via normal network training, which would learn the importance of connections. The second step is to prune the low-weight

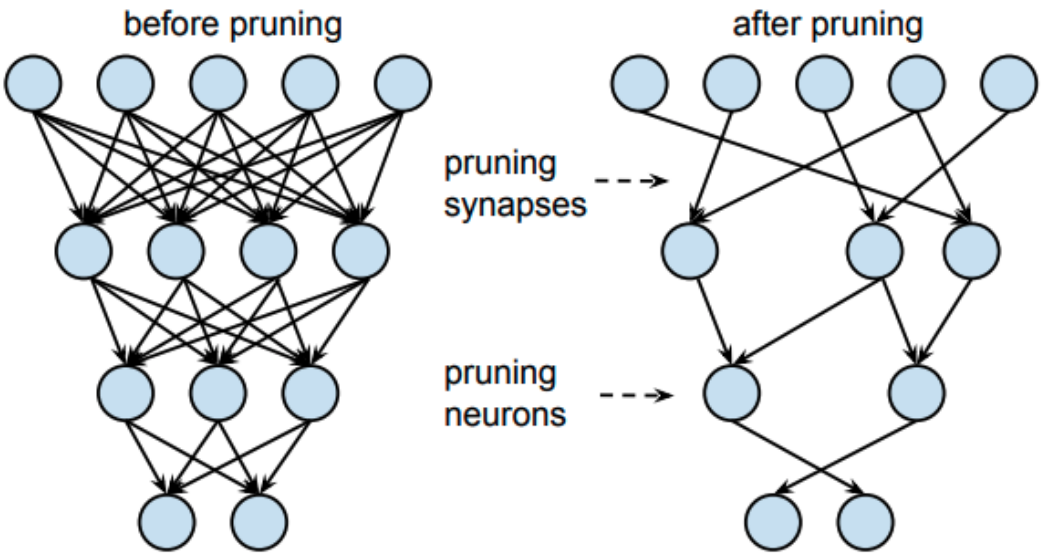

Figure 2: Synapses and neurons before and after pruning.

connections, it would use a mask layer to convert a dense network into a sparse network, as shown in Figure 2. Then in the final step, the network will be retrain to learn the final weights for the remaining sparse connections.

Moreover, I also try the prune method supported by the PyTorch in my project and I compare the performance of these two approaches.

5. Model Description

LeNet is a convolutional neural network structure proposed by Yann LeCun et al. in 1998. In general, LeNet refers to lenet-5 and is a simple convolutional neural network. Convolutional neural networks are a kind of feed-forward neural network whose artificial neurons can respond to a part of the surrounding cells in the coverage range and perform well in large-scale image processing. Figure 3 shows the whole architecutre of LeNet-5.

The input for LeNet-5 is a 32×32 grayscale image which passes through the first convolutional layer with 6 feature maps or filters having size 5×5 and a stride of one. The image dimensions changes from $32 \times 32 \times 1$ to $28 \times 28 \times 6$.

Then the LeNet-5 applies average pooling layer or sub-sampling layer with a filter size 2×2 and a stride of two. The resulting image dimensions will be reduced to $14 \times 14 \times 6$.

Next, there is a second convolutional layer with 16 feature maps having size 5×5 and a stride of 1. In this layer, only 10 out of 16 feature maps are connected to 6 feature maps of the previous layer.

The fourth layer (S4) is again an average pooling layer with filter size 2×2 and a stride of 2. This layer is the same as the second layer (S2) except it has 16 feature maps so the output will be reduced to $5 \times 5 \times 16$.

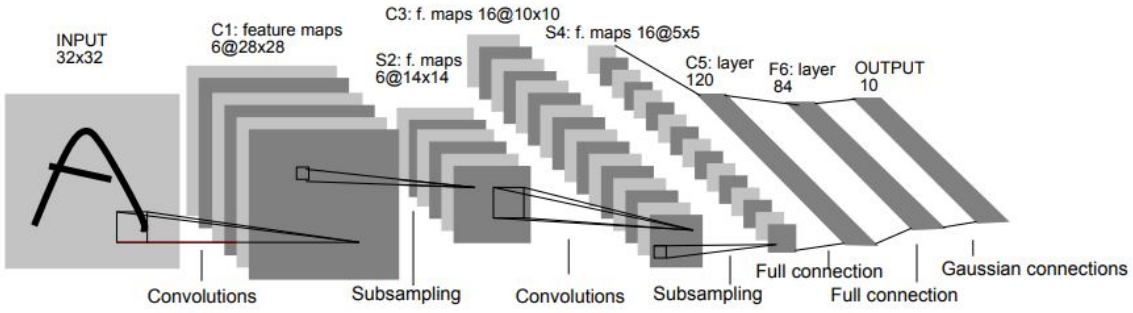

Figure 3: Original LeNet-5 Architecture

The fifth layer (C5) is a fully connected convolutional layer with 120 feature maps each of size 1×1. Each of the 120 units in C5 is connected to all the 400 nodes ($5 \times 5 \times 16$) in the fourth layer S4.

The sixth layer is a fully connected layer (F6) with 84 units.

Finally, there is a fully connected softmax output layer $\hat{y}$ with 10 possible values corresponding to the digits from 0 to 9.

6. Experimental Procedure and Results

In my experiment, I first find threshold value for pruning, which is the sparsity ratio that I require before running. Then I generate a mask tensor to prune the original weights. Finally I would use the updated weight to retrain the model.

In more detail, I calculate the mid-value which can satisfy the sparsity ratio before the training, then make a mask array to store the location information, after each training, use the mask layer to replace the trained weights. Then do another epoch of training. Repeat this process again and again until meet the epoch requirement.

I did several experiments by adjusting the hyper-parameters, e.g., learning rate, sparsity ratio and running epoch. I would make sure the other parameters are consistent when I adjust one specific parameter. My default parameters can be list as follows: batch size(128), learning rate(0.01), SGD momentum(0.5).

Figure 4 shows the change of loss with epoch number under different sparsity ratio, the sparsity ratio means the ratio of weights the model would remove during the pruning step. From the image we can find that the smaller the compression ratio, the better the performance.

Figure 5 shows the number of images which can be classified correctly after 5 epochs, which can prove the relationship between sparsity ratio and loss. Moreover, the loss would converge after enough epochs. Figure 6 shows the the loss of high sparsity ratio situation, we can find that if we increase the number of iterations, we can gain a high quality but sparse model. So I try to enlarge the epochs to 60 to see the performance when model converge. Figure 7 shows the comparison between original model and a model with 0.9 sparsity ratio, their performance difference is very small but always exist.

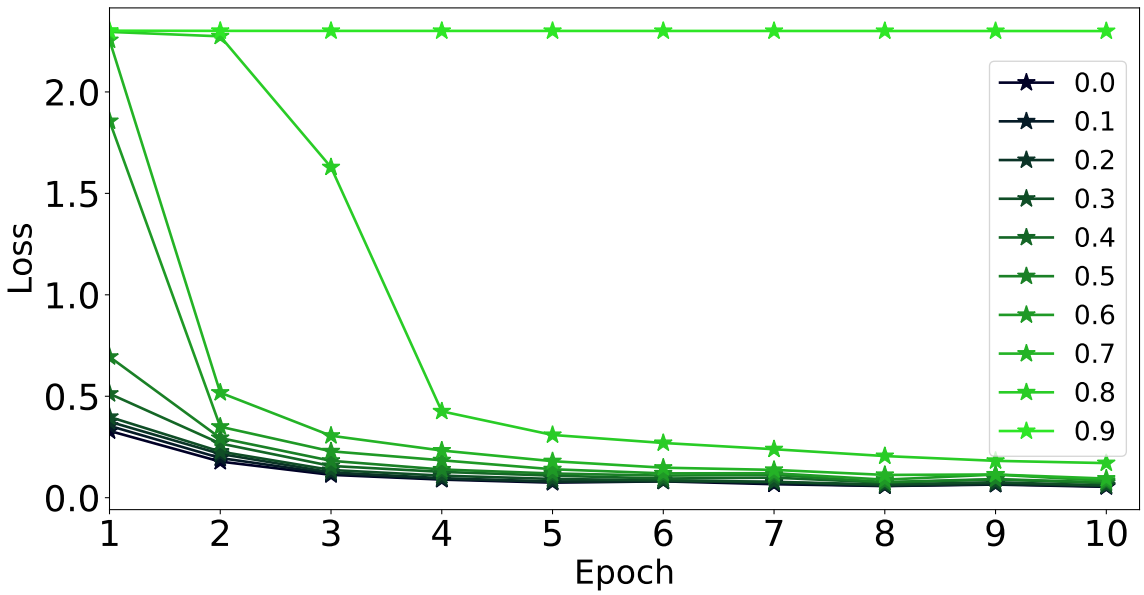

Figure 4: Loss With Different Sparsity Ratio

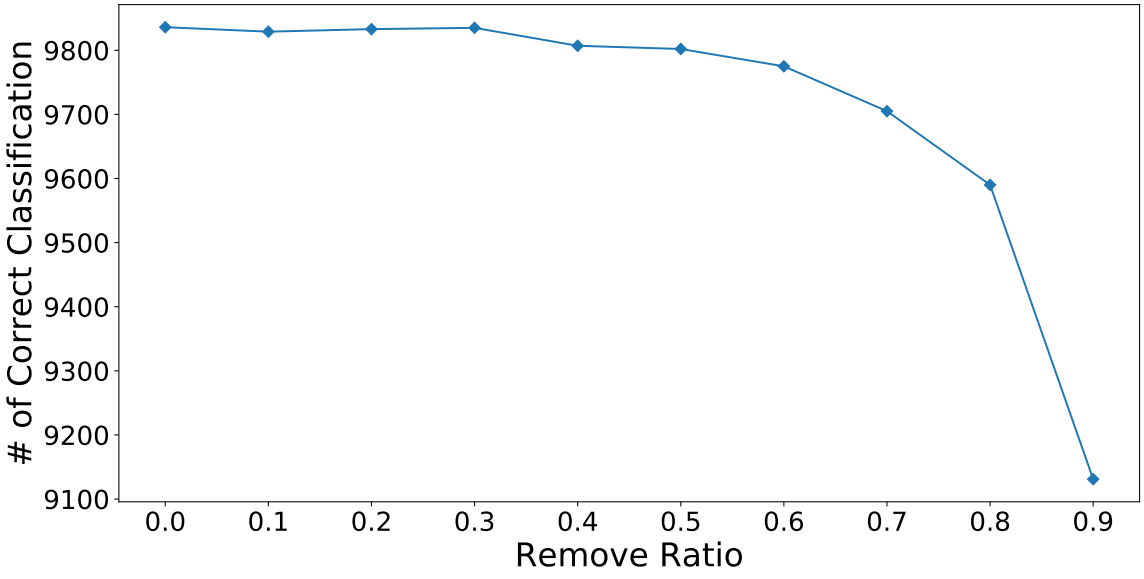

Figure 5: Number of Correction Classification Images After 5 Epochs

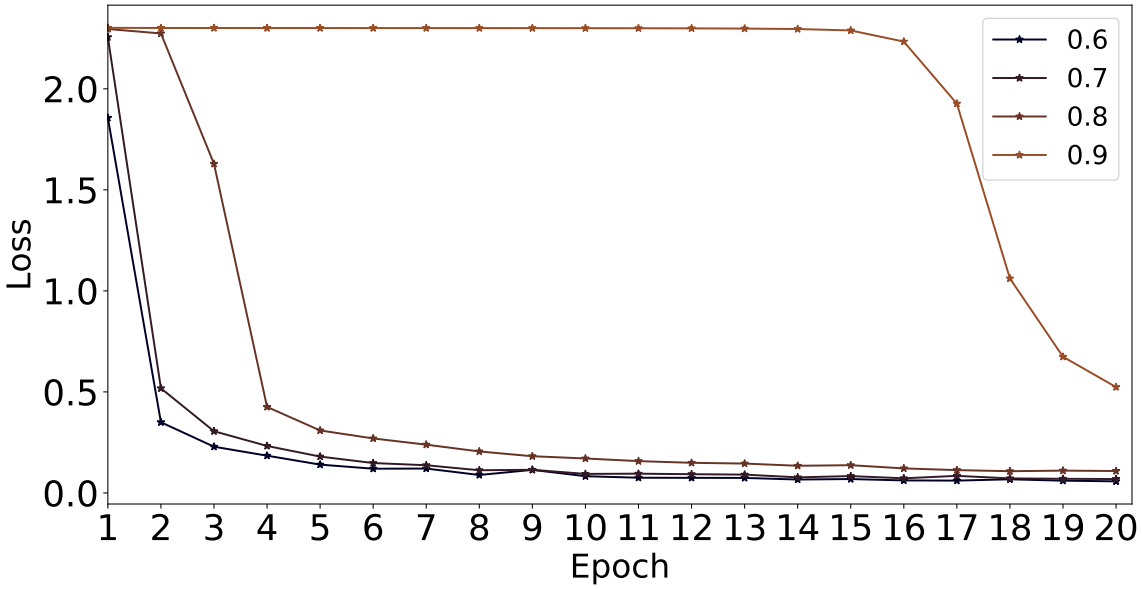

Figure 6: Loss of 20 Epochs

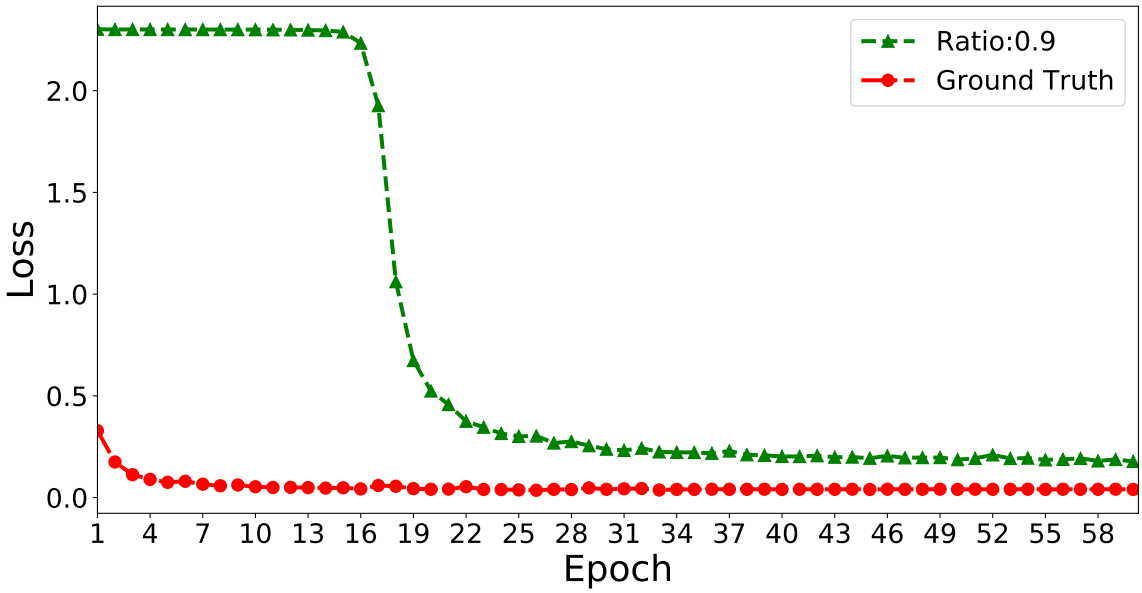

Figure 7: Comparison of 60 Epochs

I also compare other factors like running time, and I found the sparsity ratio would not influence training time a lot but a high sparsity ratio would decrease the test time a little. The difference is not very significant because the LeNet-5 model is very simple, and the average running time is short.

7. Conclusion

In this project, I start from the slide of lecture 13 to have the basic knowledge of network pruning, then I read several papers about modern network pruning skills and model compression and acceleration. Having a basic idea about what is deep learning pruning, I tried to prune a common image classification model, LeNet-5 on the common image classification dataset, MNIST. I conduct several experiments by adjusting the hyper-parameters. Finally, I achieve a high sparsity model which just have a negligible performance loss comparing to the original LeNet-5 model.

References

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *CoRR*, abs/1710.09282, 2017. URL `http://arxiv.org/abs/1710.09282`.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015. URL `http://arxiv.org/abs/1506.02626`.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR 2016 : International Conference on Learning Representations 2016*, 2016.

P. Langley. Crafting papers on machine learning. In Pat Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.