Reconstruction net pruning using ADMM

Hao Zhang and Zixuan Chen
Shanghai Jiao Tong University

1. INTRODUCTION

In this project, we apply several neural network compression algorithms to the inverting image network to test the robustness of these algorithms. Specifically, we use the reconstruct net from the paper *Inverting Visual Representations with Convolutional Networks* and compress this network. We extract features of these images from the fourth and fifth hidden layer from the AlexNet and use the Cifar-10 dataset as a replacement of the ImageNet. We then apply the ADMM algorithm from the paper *A Systematic DNN Weight Pruning Framework using Alternating Direction Method of Multipliers* to compress and retrain the pretrained network. At last, we compare the performance(including the value of loss function and the quality of the inverted images) of the nets before and after prunning.

2. BACKGROUND AND RELATED WORKS

2.1 AlexNet

AlexNet is firstly proposed by A.Krizhevsky, I.Sutskever and G.E.Hinton in 2012. And this model got the champion in the ImageNet competition in 2012. This model successfully used some tricks such as ReLU, Dropout and LRN. AlexNet is considered one of the nost influential papers published in computer vision, having spurred many more papers published employing CNNs and GPUs to accelerate deep learning.

2.2 Inverting image

Feature representations are always hard to analyze, and we can learn the representations by inverting an image from a reconstruct network. There are many approaches to invert an image. According to the work of A.Dosovitskiy and T.Brox, we can extract features from the hidden layer of a CNN and inverting an image with several convolutional layers and up-convolutional layers. In A.Dosovitskiy's work, they used AlexNet and get a good result to invevrt images.

2.3 introduction to ADMM algorithm

ADMM was first introduced in the 1970s, and theoretical results in the following decades have been collected in [1]. It is a powerful method for solving regularized convex optimization problems, especially for problems in applied statistics and machine learning.

For some problems which are difficult to solve directly, we can use variable splitting first, and then employ ADMM to decompose the problem into two subproblems that can be solved separately and efficiently. For example, the op-timization problem.

$$minimize_x \ f(x) + g(x) \tag{1}$$

assumes that f() is differentiable and g() is non-differentiable but has exploitable structure properties. To make it suitable for the application of ADMM, we use variable splitting to rewrite the problem as

$$minimize_x \ f(x) + g(x)$$
$$\text{subject to } x = z$$

more details of ADMM algorithm will be illustrated in section (5)

3. MOTIVATION

Nowadays, deep neural networks play an important role in many fileds, such as image classification, natural language processing, image semantic segmentation and so on. But it's getting more difficult to train a deep networks since the parameters become more and more. Although there are several effective methods to decrease the number of parameters, such as using convolutional layer, using smaller convolutional kernel, sometimes it's still necessary to design compress algorithm to simplify the network to get a better training speed.

There are already some methods to decrease the complexity of the network. However, as far as we know, these algorithms are mainly applied on the classifiers and use the accuracy as an index to judge whether this algorithm will affect the performance. And most of these algorithms will adjust the loss function, which will lead to a slightly increasement of original loss. Since the accuracy only focus on the largest or the five largest channels, we think it can not totally reflect the affaction of the compression algorithm. So we decide use another model and choose a compress algorithm to test its robustness.

We choose the reconstruct net and ADMM algorithm because this net can make the result more visible and clear to human.

4. PROBLEM FORMULATION

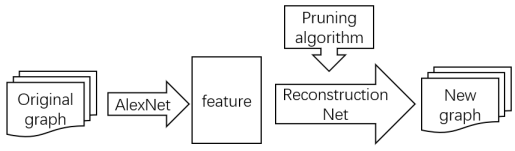

Fig. 1. problem formulation

Our problem is to reconstruct graph from features generated from AlexNet and then generate proper algorithm to prune the reconstruction net. The origin AlexNet is trained with ImageNet, ,which is too large for us to train,

4.1 Inverting image

The origin AlexNet is trained with ImageNet, ,which is too large for us to train, so we firstly adjusted and trained our own AlexNet with a smaller dataset Cifar-10. Then we extract images' features from

	reconstruct v1	reconstruct v2
from layer	4th	5th
feature size	4*4*128	8*8*96
conv num	3	3
conv K/S	3/1	3/1
convt num	3	2
convt K/S	5/2	5/2

Table I.

the fourth and fifth layer respectively and build two reconstruct nets to invert images. We used the parameters from A.Dosovitskiy's work. The construct of these two models are shown in the Table1.
In the Table1, K represents the kernel size and S represents the strides of the kernel. When these two models are trained, we can apply our algorithms to compress them.

4.2 Problem formulation of weight pruning

Assume we want to prune a fully connected DNN layers, and suppose we have input **x**, the output **h** of the fc layer will be

$$h = \sigma(Wx + b)$$

where **h** and **b** have t columns. The nonlinear activation function $\sigma()$ acts entrywise on its arguments, and is typically chosen to be the ReLU function. the output of the i-th layer will be

$$h_i = \sigma(W_i x + b_i)$$

Suppose we input a batch of images, the output of the DNN will be

$$y = W_N h_{N-1} + b_N$$

in this case s is a $k \times t$ matrix. The loss function of this DNN is illustrated as

$$f(\{\mathbf{W}_1, \ldots, \mathbf{W}_N\}, \{\mathbf{b}_1, \ldots, \mathbf{b}_N\}) = -\frac{1}{t}\sum_{j=1}^{t} \log \frac{e^{\mathbf{s}_{y_j j}}}{\sum_{i=1}^{k} e^{\mathbf{s}_{ij}}} + \lambda \sum_{i=1}^{N} \|\mathbf{W}_i\|_F^2,$$

where $||\cdot||_F^2$ denote the Frobenius norm, the first term is the cross entropy loss, the second term is the l2 loss, and the second term is the l2 regularization term. The training of DNN is a process of minimizing the loss by updating weight W and bias b

$$W_i = W_i - \alpha \frac{\partial f(W_i b_i)}{\partial W_i}$$
$$b_i = b_i - \alpha \frac{\partial f(W_i b_i)}{\partial b_i}$$

for i = 1, ..., N and α is the learning rate.
Our objective is to prune the weights of the DNN, and therefore we minimize the loss function subject to constraints on the cardinality of weights in each layer.More specifically, our training process solves

$$\begin{aligned} \underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad & f\big(\{\mathbf{W}_i\}, \{\mathbf{b}_i\}\big), \\ \text{subject to} \quad & \text{card}(\mathbf{W}_i) \le l_i,\ i = 1, \ldots, N, \end{aligned}$$

where card return the number of nonzero element of the given matrix and l_i is the desired number of each layer.

5. PROPOSED METHODS-ADMM

In the above section we discussed the problem formulation of weight pruning as

$$\begin{aligned} \underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad & f\big(\{\mathbf{W}_i\}, \{\mathbf{b}_i\}\big), \\ \text{subject to} \quad & \mathbf{W}_i \in \mathbf{S}_i,\ i = 1, \ldots, N, \end{aligned}$$

where $S_i = W_i | card(W_i) \le l_i, i = 1, ..., N$. It is clear that $S_1, ..., S_N$ are nonconvex sets, and it is general difficult to solve optimization problem with nonconvex constraints. the problem can be equivalently rewrittern in a form without constraints, which is

$$\underset{\{W_i\},\{b_i\}}{minimize}\, f(\{W_i\}\{b_i\}) + \sum_{i=1}^{N} g_i(W_i) \tag{2}$$

where $g_i()$ is the indicator function of S_i, i.e.,

$$g_i(W_i) = \begin{cases} 0 \text{ if } card(W_i) \le l_i \\ +\infty \text{ otherwise} \end{cases}$$

The first term of problem 2 is the loss function of a DNN, while the second term is non-differentiable. This problem cannot be solved analytically or by stochastic gradient descent. We begin by equivalently rewriting the above problem in ADMM form as

$$\underset{\{W_i\},\{b_i\}}{minimize}\, f(\{W_i\}\{b_i\}) + \sum_{i=1}^{N} g_i(Z_i) \text{subject to } W_i = Z_i, i = 1, ..., N$$

The augmented Lagrangian of the above optimization problem is given by

$$\begin{aligned} L_\rho(\{\mathbf{W}_i\}, \{\mathbf{b}_i\}, \{\mathbf{Z}_i\}, \{\mathbf{\Lambda}_i\}) = f(\{\mathbf{W}_i\}, \{\mathbf{b}_i\}) + \sum_{i=1}^{N} g_i(\mathbf{Z}_i) \\ + \sum_{i=1}^{N} \text{tr}[\mathbf{\Lambda}_i^T(\mathbf{W}_i - \mathbf{Z}_i)] + \sum_{i=1}^{N} \frac{\rho_i}{2} \|\mathbf{W}_i - \mathbf{Z}_i\|_F^2, \end{aligned}$$

If we define the scaled dual variable $U_i = (1/\rho_i)\Lambda_i$, the augmented lagrangian can be equivalently expressed as

$$\begin{aligned} L_\rho\big(\{\mathbf{W}_i\}, \{\mathbf{b}_i\}, \{\mathbf{Z}_i\}, \{\mathbf{\Lambda}_i\}\big) = f\big(\{\mathbf{W}_i\}, \{\mathbf{b}_i\}\big) + \sum_{i=1}^{N} g_i(\mathbf{Z}_i) \\ + \sum_{i=1}^{N} \frac{\rho_i}{2} \|\mathbf{W}_i - \mathbf{Z}_i + \mathbf{U}_i\|_F^2 - \sum_{i=1}^{N} \frac{\rho_i}{2} \|\mathbf{U}_i\|_F^2. \end{aligned}$$

The ADMM Algorithm proceeds by repeating, for k = 0,1,..., the following steps:

$$\{W_i^{k+1}\}\{b_i^{k+1}\} := \underset{\{W_i\},\{b_i\}}{argmin}\, L_\rho(\{W_i\}, \{b_i\}, \{Z_i^k\}, \{U_i^k\}) \tag{3}$$

$$Z_i^{k+1} := \underset{\{Z_i\}}{argmin}\, L_\rho(\{W_i^{k+1}\}, \{b_i^{k+1}\}, \{Z_i\}, \{U_i^k\}) \tag{4}$$

$$U_i^{k+1} := U_i^k + W_i^{k+1} - Z_i^{k+1} \tag{5}$$

until both of the following conditions are satisfied.

$$\|W_i^{k+1}, b_i^{k+1}\|_F^2 \leq \epsilon_i, \|Z_i^{k+1} - Z_i^k\|_F^2 \leq \epsilon_i \tag{6}$$

problem (3) can be formulated as

$$\underset{\{W_i\},\{b_i\}}{minimize} f(\{W_i\}\{b_i\}) + \sum_{i=1}^{N} \frac{\rho_i}{2} \|W_i - Z_i^k + U_i^k\|_F^2$$

ans problem (4) can be formulated as

$$\underset{\{Z_i\}}{minimize} \sum_{i=1}^{N} g_i(Z_i) + \sum_{i=1}^{N} \frac{\rho_i}{2} \|W_i^{k+1} - Z_i + U_i^k\|_F^2$$

6. EXPERIMENTS

6.1 Partly compress

We decided to finish our experiment step by step, so we decide to firstly compress the convolution layers only and check whether the algorithm can compress the network effectively. Then we will compress all the layers to test its robustness.

We have chose 3 layers of reconstruction net, that is, the first 3 convolutional layers of reconstruction net to apply our ADMM algorithm. we compare the original graph, the recontructed graph by the original reconstruction net as well as the reconstructed graph by the pruned Reconstruction net. We calculate the nonzero weight values and the weight distribution of each Reconstruction net. And the result is as follows. From figure2 we can see that the recontructed graphs generated from the original reconstruction net are quite different from the original graph due to the higher fuzzy degree. What is amazing is there is small difference between he recontructed graph by the original reconstruction net as well as the reconstructed graph by the pruned Reconstruction net. We can see that before pruning, each layer have different distribution. After pruning process, most of the weight values of each convolutional layer have become zero. From the figure3 and figure4 we can see that almost 90% of the weight value have been pruned. From the above experiments, we can see that the ADMM algorithm is very practical for image reconstruction. However, there is slightly little challenge in balancing the prune rate to secure the degree of accuracy.

6.2 Complete Compress

Since it's useful to apply this algorithm to the reconstruct net, we moved on and compress all the layers. This step is more challenge for the algorithms because the up-convolutional layers are quite important for the inverting images. We test this algorithm with the features from the fourth hidden layer and the fifth hidden layer of the AlexNet. And we need one more up-convolutional layer to prune when inverting images from the fifth layer. In this step, we fix the compress rate at 0.9 and retrain enough epoches to get a precise number of nonzero weight values.And the inverting results of the two model are as follows.

According to the figure5 and figure6, we can see that figure5 has a better performance than figure6, which is easy to understand because inverting an image from a deeper layer is more difficult. However, when we observe the result of the pruned net, we can easily find that the performance of pruned nets are clearly worse than the origin reconstruct net. Comparing with the result of the part compress, we know that this is because we prune the up-convolutional layers. What's worse, the difference between the two nets of figure6 is larger than the two nets of figure5 and the pruned net in figure6 sometime even make mistakes about the color.

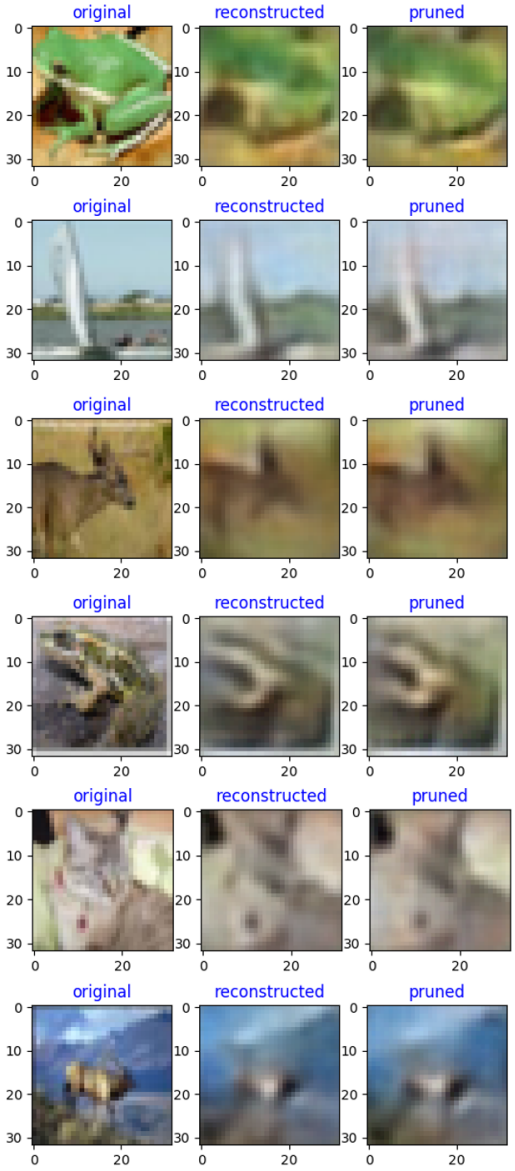

Fig. 2. the original graph, the recontructed graph by the original reconstruction net as well as the reconstructed graph by the partly pruned Reconstruction net

7. CONCLUSION AND FUTURE WORK

In this paper, we apply ADMM algorithm in image reconstruction net. We formulated the process of Inverting an image and pruning a DNN layers. We then introduced the detailed mathematical derivation of ADMM algorithm. We use AlexNet to extract features of the original image and reconstruct the image using the reconstruction net. Then we apply ADMM algorithm to prune the reconstruction net.

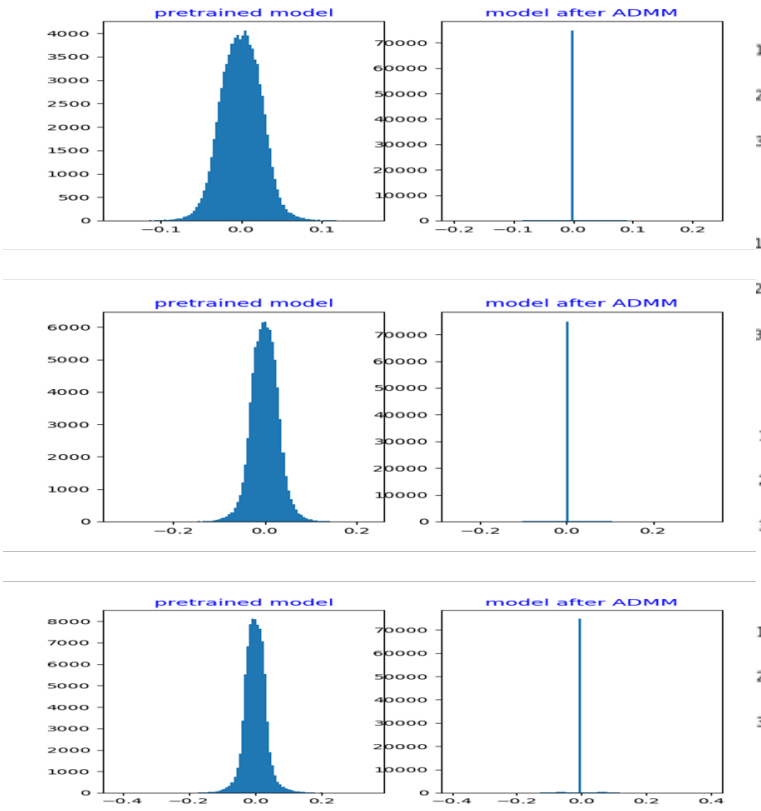

Fig. 3. the weight distribution of each Reconstruction net, the first row is the conv1 layer, the second row is the conv2 layer, the third row is the conv3 layer (partly compress)

Layer	Non-zero weight	None-zero Weight after prune	None-zero Weight after prune %
Conv1	82944	8295	90%
Conv2	82944	8626	89.6%
Conv3	82944	8875	89.3%
total	248k	25.7k	89.7%

Fig. 4. the weight distribution of each Reconstruction net, the first row is the conv1 layer, the second row is the conv2 layer, the third row is the conv3 layer (partly compress)

In the experiment, we find that when we apply this algorithm to all the layers in the net, the affact caused by the algorithms becomes more visible due to the representation of inverting images.
In the future, we plan to compress reconstruction network from layer 5 of Alexnet. In the mean time, we plan to try to combine ADMM algorithm with quantization.

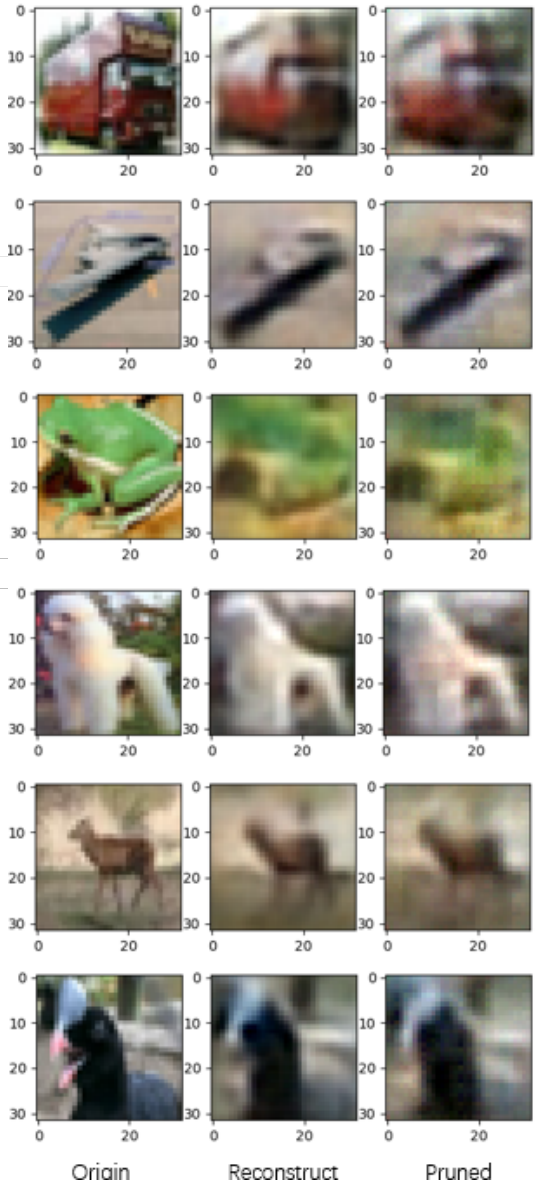

Fig. 5. the original graph, the recontructed graph by the original reconstruction net as well as the reconstructed graph by the pruned Reconstruction net, features from the fourth layer

APPENDIX

A. REFERENCE

REFERENCES

1. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning 3(1) (2011) 1122

2. Zhang T , Ye S , Zhang K , et al. A Systematic DNN Weight Pruning Framework using Alternating Direction Method of Multipliers[J]. 2018.

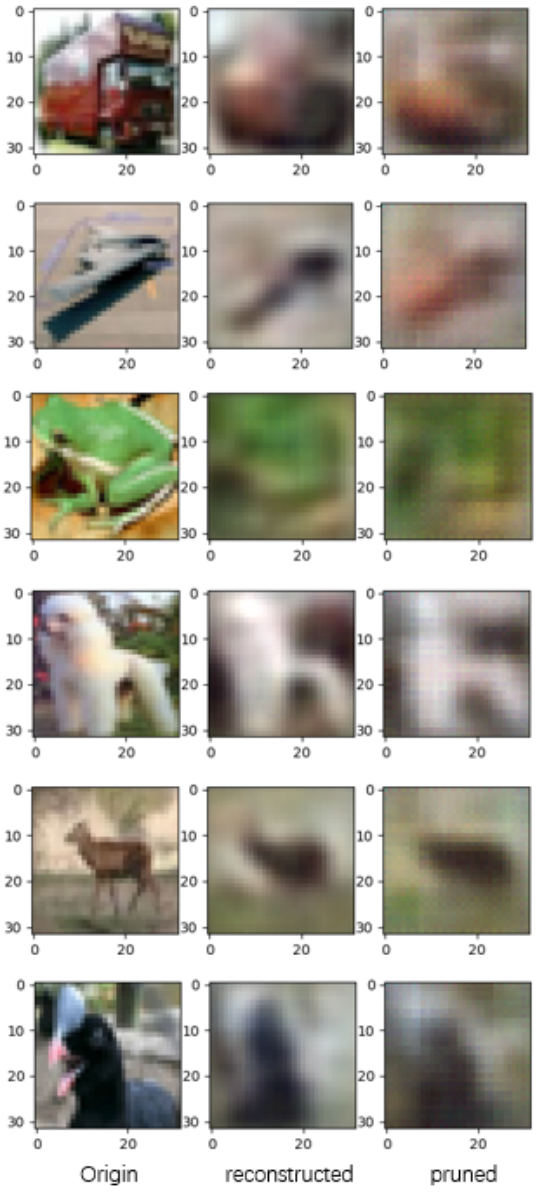

Fig. 6. the original graph, the recontructed graph by the original reconstruction net as well as the reconstructed graph by the pruned Reconstruction net, feaetures from the fifth layer

3. Dosovitskiy A, Brox T. Inverting Visual Representations with Convolutional Networks[C]// IEEE Conference on Computer Vision & Pattern Recognition. 2016.
4. Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. 2012.