A pruning based method to learn both weights and connections for LSTM

Shijian Tang
Department of Electrical Engineering
sjtang@stanford.edu

Jiang Han
Department of Electrical Engineering
hanjiang@stanford.edu

Abstract

This project is one of the research topics in Professor William Dally's group. In this project, we developed a pruning based method to learn both weights and connections for Long Short Term Memory (LSTM). In this method, we discard the unimportant connections in a pretrained LSTM, and make the weight matrix sparse. Then, we retrain the remaining model. After we remaining model is converge, we prune this model again and retrain the remaining model iteratively, until we achieve the desired size of model and performance. This method will save the size of the LSTM as well as prevent overfitting. Our results retrained on NeuralTalk shows that we can discard nearly 90% of the weights without hurting the performance too much. Part of the results in this project will be posted in NIPS 2015.

1 Introduction

Deep neural network has been widely used in natural language processing (NLP) tasks [1-5] including machine translation, image captioning, name entity recognition, sentiment analysis, and language models. Besides the fully connected neural network, the most popular models for NLP task is recurrent neural network (RNN) [1]. RNN can be used to model the sequential data such as language. However, the vanila RNN is very difficult to train due to the vanishing gradient problem [6]. Long short term memory (LSTM) [7] is a RNN model which has complex gating functions to capture the long and short term memory to make the model easier to train.

One of the problems of LSTM is that the model typically has large number of learnable parameters due to three gating functions. That makes the model easier to overfit and has a large number of parameters which cost a lot of memory, which reduces the power of LSTM in practice for a small amount of data or mobile usage. This is also a general problem for deep learning. Professor William Dally's group is working on the topic of efficient deep learning to solve this problem. One of the work is pruning the deep neural network and makes the model sparse to save storage as well as sparse matrix operation to speed up computation. The group has made progress on convolutional neural network (CNN) [8]. This project is part of the current research in this group, which is extending the similar idea in RNN models such as LSTM for NLP applications. Th goal of this project is to explore the performance of pruning LSTM, and investigate how much weights we can discarded without hurting the performance of LSTM.

The basic idea of pruning deep neural network is that given a pretrained model, we can ignore the unimportant connections. The unimportant connections is the connections between neurons which have a relative smaller absolute value of weights. The input of the connection with smaller weights has little effect on the output. Therefore, we can discard that connections. The intuition is inspired from the human brain [9]. Some of the neuron connections developed in children are gradually discarded when they grow up. These discarded connections are typically less used or less important.

After we pruned the pretrained model, we should retrain the remaining part to adjust the weights. After the retrained model converge, we can still treat it as a pretrained model and then prune and retrain iteratively. During this procedure, we can learn both weights and connections in the deep neural network, rather than learning weights only by regular method. Also, this method will also prevent overfitting and underfitting. In

the case of a smaller dataset, we first adopt a deep network which typically overfits the data if trained directly, then we prune the pretrained overfitted model, and then retrain it to prevent overfitting and also underfitting.

2 Pipeline of model

In this section, we will describe the pipeline of learning both weights and connections based on pruning. The general pipeline is shown in Fig. 1, where we prune the pretrained model first, and then iteratively retrain and prune the remaining model to some extent.

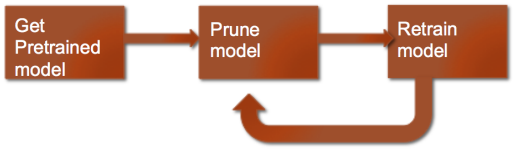

Figure 1: The pipeline of pruning based method to learn both weights and connections.

2.1 Prune the pretrained model.

The key procedure of our method is pruning the pretrained model. As we mentioned about, the goal of pruning weights is to discard the unimportant connections between neurons. Here we use the value of the weights to evaluate the "importance" of connections. The connections with weights relatively smaller will be labeled as unimportant and will be discarded. Fig. 2 shows the example of one layer of pretrained fully connected neural network with certain weights. We can find that the weights of connections with blue color are relatively smaller than the weights of red connections. Therefore, the input of blue connections has less effect on the output neuron, and we can observe that these connections are less important. After we discard the blue connections we achieve a sparse neural networks. In this project, instead

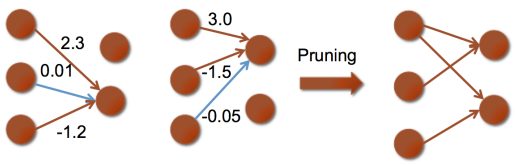

Figure 2: Pruning the unimportant connections based on weights.

of using the raw absolute value to prune, we prune the connections based on the weights distribution of pretrained model. For example, we will prune the connections with the weights within the standard deviation of weights in pretrained model. For more details in the relationship between std and pruned percentage is in the results section.

2.2 Pretrained model and dataset

The first steps of our method is to get a pretrained model in standard LSTM. For fairness consideration, we download a public available pretrained model instead of training from scratch by ourself, because the weights in the public available model are well adjusted and easy to compare.

We choose the Neural Talk [3] as our pretrained model since it is trained on the LSTM and the pretrained model is public available in `http://cs.stanford.edu/people/karpathy/neuraltalk/`. The Neural Talk is used to generate image captioning which is a NLP task. The model feed the image features extracted from CNN to the first time step of LSTM, and use the LSTM as a sequence generator to generate sentence to describe the image. The following equations describe this model.

$$i = W_e CNN(I)$$
$$y(t) = LSTM(i \cdot I(t=0), W_d x(t))$$

where I is the raw image, and $CNN(i)$ is the image feature extracted from CNN, and W_e is the embedding matrix for image, $LSTM()$ represent is the LSTM. It takes $x(t)$ which is the word vector in each time step, W_d is the embedding matrix of words. LSTM only takes the image features i as input at the first time step where the indicator function $I(t=0)$ is 1. $y(t)$ is the predicted word.

For implementation of LSTM, we stack the weights of input gates, output gates and forgot gates together as a large matrix denoted as W_{LSTM}, so in our project, we investigate pruning W_e, W_d and W_{LSTM}.

We use the model trained on Flickr 8K dataset as our pretrained model. The Flickr 8K dataset has 8000 images with each image has 5 sentences to describe it. We choose each image and sentence pair as a sample, and also discard the rare words in the ground truth. After preprocessing, there are 40000 training samples and 2538 distinct words in vocabulary.

The size of a fully connected LSTM on Flickr 8K dataset is W_e is 4096 by 512, W_d is 512 by 2538 and W_{LSTM} is 1025 by 2048. The total learnable parameters are 5495808.

2.3 Retrain the model

After we prune the pretrained model, the next step is retraining the remaining model. We have two options to retrain: one is prune directly to the desired percentage, and retrain the model only once. Another option is iteratively prune the model, and retrain multiple times. For example, if we plan to prune 90% of the weights, we can directly prune 90% of the weights, and retrain the model. Otherwise, we can prune 50% first, and then retrain the remaining model until it converges, then we can further prune the retrained model to 60% and retrain, we can prune and retrain repeatedly until we reach the model with only 10% of the weights remains.

The advantages of iteratively pruning is that we can learn both weights and connections gradually, instead of directly prune too much connections once. The step between each pruning is a hyper-parameter and requires tuning in practice.

To retrain the model, we should adjust the dropout ratio, because after pruning, we have less number of neurons, so we should dropout less number of the neurons. Also, we need to use a relatively smaller learning rate and relatively less epochs to avoid destroy the pretrained weights too aggressively. We find the 10^{-4} learning rate with 10 training epochs will make the model converge and achieves better results.

2.4 Evaluation

We use the BLEU score to evaluate the performance of image captioning. BLEU score [9] has been widely used in machine translation and image captioning. We compute the BLEU score of BLEU1, BLEU2, BLEU3, and BLEU4 for each retrained model, and compare the BLEU score of the pretrained model.

2.5 Implementation

The implementation of pruning and retraining is using the mask matrix for W_e, W_d and W_{LSTM}. The reason is that for the discarded connections, we set its value as zero. Therefore the mask matrix can be constructed as follows, we set the pruned connections as 0 and remaining connections as 1. For forward propagation we elementwised multiply the weights matrix by the mask matrix before each forward propagation. And for backpropagagtion, we elementwisely multiply the gradients of weights by the mask before we update the weights.

3 Experiments

Our experiments consist of four parts: first, we study the relationship between standard deviation of weights distribution and the pruned percentage. Second we study the performance of prune model without retraining, and investigate the sensitivity of W_e, W_{LSTM} and W_d. Third, we experiment with directly pruning, which is pruning only once. Last, we study the performance of iteratively retrain.

3.1 Pruning percentage

We prune the model based on the standard deviation of the weights distribution of pretrained model rather than the raw absolute value. To quantify the pruning rate, we define a term of threshold th. The pruned threshold is th means we discard the weights between $[-th * std, th * std]$ where std is the standard deviation of pretrained model. So, $th = 0$ means we do not prune any connections. The larger value of th, the more connections are discarded and we have a higher pruning rate. Table 1 shows the pruning rate and threshold of W_e, W_d and W_{LSTM}. The pruning rate represents the percentage of connections are discarded, the 90% means only 10% of weights remains. We can find that the weights distribution is slightly different among different matrix due to the threshold is slightly different for given prune rate.

Prune rate	W_e	W_d	W_{LSTM}
0%	0	0	0
50%	0.75	0.68	0.7
60%	0.91	0.85	0.85
70%	1.1	1.06	1.03
80%	1.3	1.3	1.26
90%	1.61	1.63	1.61

Table 1: Relationship between threshold and pruning weights for W_e, W_{LSTM} and W_d.

3.2 Performance without retrain

We will analysis the performance of directly pruning without retraining. Table 2 is the BLEU score performance without retraining. We can find that

as the increase of pruning rate, the BLEU score decreases dramatically if we do not retrain. This is because that the orignal weights can not be adopted directly for a sparse model. Note that here we investigate the overall pruning rate, which is we prune all the weight matrix together.

Prune rate	BLEU1	BLEU2	BLEU3	BLEU4
0%	55.7	37.3	24	15.7
50%	54	35.2	22	14.1
60%	51.5	32.8	19.9	12.3
70%	45.6	27.5	15.4	8.9
80%	35.3	29.2	7.6	2.6
90%	27.3	0	0	0

Table 2: The BLEU score with various overall pruning rate.

WLSTM pruning rate	Threshold	BLEU-1	BLEU-2	BLEU-3	BLEU-4
0%	0	55.7	37.3	24	15.7
30%	0.4	55.1	36.8	23.6	15.4
40%	0.53	55.4	36.8	23.6	15.4
50%	0.7	54.6	36.3	23.1	15.1
60%	0.85	51.8	34.1	21.4	13.7
70%	1.03	48	30.3	18.1	11.1
80%	1.26	40.8	24.7	13.4	7.4
90%	1.61	33.5	24.5	13.1	7.3

Figure 3: The BLEU score with various W_{LSTM} pruning rate.

We pruning rate	Threshold	BLEU-1	BLEU-2	BLEU-3	BLEU-4
0%	0	55.7	37.3	24	15.7
30%	0.44	55.7	37.4	24.1	15.7
40%	0.59	55.5	37.3	23.9	15.5
50%	0.75	55.5	37.3	24	15.7
60%	0.91	55.3	37.1	23.8	15.5
70%	1.1	55.6	37.3	23.9	15.5
80%	1.3	55.8	37.4	23.8	15.4
90%	1.61	56.2	37.7	24.2	15.7

Figure 4: The BLEU score with various W_e pruning rate.

Then we will prune each matrix of W_e, W_d and W_{LSTM} separately to study which matrix is the most important and more sensitive to the pruning. Fig. 3 is the performance of pruning W_{LSTM} only. Fig. 4 is the performance of pruning W_e only and Fig. 5 is the performance of pruning W_d only.

We can find that the model is more sensitive to W_{LSTM}. The performance degrades quickly as the increase of pruning rate, and W_e is less sensitive to pruning, the performance does not changed too much. This is because the W_{LSTM} dominates the weights in LSTM model, and W_e is only used in the first time step when embedding the image features. Therefore in practice we can prune the W_e more aggressively, and prune W_{LSTM} less.

Wd pruning rate	Threshold	BLEU-1	BLEU-2	BLEU-3	BLEU-4
0%	0	55.7	37.3	24	15.7
30%	0.39	56.2	37.6	24	15.5
40%	0.54	55.2	36.5	23.3	15.1
50%	0.68	55.9	36.9	23.4	15
60%	0.85	54.7	35	21.5	13.4
70%	1.06	54.1	34.3	20.7	12.9
80%	1.3	45.3	37.3	19.8	11.7
90%	1.63	32.1	28	9.3	0

Figure 5: The BLEU score with various W_d pruning rate.

3.3 Performance of retrain model

Here we will investigate the performance of directly prune our model to the desired percentage and retrain only once, as well as iteratively prune. Table 3 and 4 show the performance of directly prune and iteratively prune respectively.

Prune rate	BLEU1	BLEU2	BLEU3	BLEU4
0%	55.7	37.3	24	15.7
50%	55	36.6	23.4	15.1
60%	55	36.7	23.8	15.5
70%	55.5	37	24	15.7
80%	55.9	37.4	24.2	15.8
90%	54.1	35.9	23	14.9

Table 3: The BLEU score with various pruning rate and prune directly retrain only once.

Prune rate	BLEU1	BLEU2	BLEU3	BLEU4
0%	55.7	37.3	24	15.7
50%	55	36.6	23.4	15.1
60%	55	36.7	23.7	15.5
70%	55.6	37	23.9	15.6
80%	55.9	37.3	23.9	15.3
90%	55.4	37.1	23.7	15.3

Table 4: The BLEU score with various pruning rate and prune and retrain iteratively.

Compared with the results from Table 2 and Table 3, we can find the if we retrain the pruned model, the performance increases a lot and is almost the same as the pretrained model (0% pruning rate). Also, we can find that when the pruning rate is 80%, the performance of retrained model is even higher than the pretrained model. This phenomenon indicates that the original model is overfitting the data. When the pruning rate is between 0% and 70%, we are actually overfitting the data. And when the pruning rate is 90%, we are underfitting the data.

Then, we will compared the results between Ta-

ble 3 and Table 4. We can find that the iterative retraining does not improves the results when the pruning rate is low compared with prune directly. However, when the pruning rate is high such as 90%, iterative retrained model is better than the directly pruned and retrained model. This is because the fact that for the high pruning rate, if we pruned the model directly to the desired pruning rate and retrain only once, we will lose some potential useful connections. However, if we iteratively prune our model, we are actually learning both weights and connections gradually. As we prune the model step by step, we will carefully discard the unimportant connections based on the converged models at each step. Therefore, the iterative pruning and retraining is better than the directly pruned model due to the better performance in high pruning rate.

We can conclude that for the LSTM model trained in Neural Talk, nearly 90% of the weights can be discarded without hurting the model performance. So the remaining weight matrix is very sparse. This result will be very useful for the scenario with limited memory and storage such as mobile usage. Storing and processing sparse matrix is more efficient than dense matrix, we will also benefit a lot from it.

3.4 Pruning limit

Our further work is to explore the limit of pruning, Which is the upper bound of the number of connections we can discard. We use the iterative pruning and retraining scheme since it has a better performance in high pruning rate.

We increase the pruning rate from 90% and find that when the pruning rate is 95%, the performance of retrain model has an obvious gap with the pretrained model, where all the weights within 3 standard deviation has been discarded. Fig. 6 shows the performance of iterative retrain, the pretrained model as well as the without retrain.

From Fig. 6, we can find that when the pruning rate is 95%, there is a significant decrease in the performance. This is because we prune too much connections and the model is underfitting the data. Therefore, we can conclude that the pruning limit of LSTM is 95%.

3.5 Pruning as regularization

Here we will discuss the difference between pruning and other regularization techniques. As we

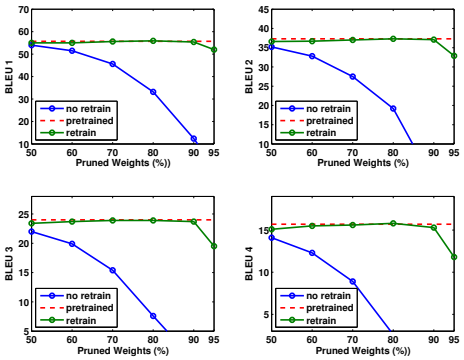

Figure 6: The performance for various pruning rate for without retrain and retrain iteratively.

mentioned above, pruning is a kind of regularization to prevent overfitting. Given a small dataset, we can train it on a large model and then iteratively prune and retrain it.

There are several other techniques which discard connections for regularization for example dropout and L1 regularization. Pruning is different with these methods. First for dropout, it is statistically discard the connections between neurons in a random way. But in pruning, we are discarding the connections in a deterministic way, which connection is pruned depends on the weights. Therefore, we are discarding weights more effectively.

L1 regularization is another regularization which tries to make the weights matrix sparse. But L1 regularization will not iteratively prune the weights matrix and can not make the weights exactly zero.

Therefore, pruning based method is able to learn both weights and connections as well as prevent overfitting.

3.6 Example image captioning

In this part, we will give some example image captioning results to show some interesting results the pruning model given. We choose the pretrained model and the model with 90% weights pruned and retrained iteratively as well as 95% pruned model.

Figs. 7-10 shows some good examples between pretrained model and 90% pruned model. We can find that the generated captioning is exactly or almost the same with only some synonyms replaced.

However, in some other cases, the 90% pruned model fails to capture some detailed information

Figure 7: Generate sentence with pretrained model and 90% pruned model
Pretrained: a brown dog is running through a grassy **field**
Pruned: a brown dog is running through a grassy **area**

Figure 8: Generate sentence with pretrained model and 90% pruned model
Pretrained: a white bird is flying over water
Pruned: a white bird is flying over water

in image compared with the original pretrained model, as Figs. 11 and 12 show. In Fig. 11 the 90% pruned model fails to capture the phrase "a red and white uniform" compared with the pretrained model. While in Fig. 12, the 90% pruned model generates unrelated phrases "on a beach" in addition to the words in original pretrained model.

If we take a look at the 95% pruned model which has a worse performance, most of the sentences has no semantics meaning. Fig. 13 is an example. Here the 95% model is confused with the group of people in the images and fails to identify a single person from a group of people.

4 Conclusion

In this project, we developed a pruning based method to learn both weights and connections in LSTM. Based on the results, we have found that the nearly 90% of the connections can be discarded in the pretrained LSTM, and the weight

Figure 9: Generate sentence with pretrained model and 90% pruned model
Pretrained: a basketball player in a white uniform is playing with a **ball**
Pruned: a basketball player in a white uniform is playing with a **basketball**

Figure 10: Generate sentence with pretrained model and 90% pruned model
Pretrained: a man is standing on a rock overlooking the ocean
Pruned: a man is standing on a rock overlooking the ocean

matrix is very sparse, which will save a large amount of storage and memory. We also find that without retrain the model, pruning will degrade the performance a lot. The iteratively pruning and retrain has a better performance than directly pruning and retrain only once in the high pruning rate. This method also prevents overfitting when the dataset is relatively small.

Our future work includes prune the three gating functions in LSTM separately, to explore which gates is more sensitive and important for LSTM.

Acknowledgments

This project is part of the research topic in Professor William Dally's group, we thank the materials

Figure 11: Generate sentence with pretrained model and 90% pruned model
Pretrained: a motorcycle racer in **a red and white uniform** is riding a motorcycle
Pruned: a motorcycle racer is riding a motorcycle

Figure 12: Generate sentence with pretrained model and 90% pruned model
Pretrained: a man is riding a surfboard on a wave
Pruned: a man in a wetsuit is riding a wave **on a beach**

and discussion from the people in this group.

References

[1] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur. 2010. *Recurrent neural network based language model*, INTERSPEECH.

[2] M. Iyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daume. 2014 *A Neural Network for Factoid Question Answering over Paragraphs*, EMNLP.

[3] A. Karpathy and L. Fei-Fei 2015 *Deep Visual-Semantic Alignments for Generating Image Descriptions*, CVPR.

[4] J. Pennington, R. Socher and C. D. Manning 2014 *Glove: Global Vectors for Word Representation*, EMNLP.

[5] H. Lakkaraju, R. Socher, and C. Manning. 2014 *Aspect Specific Sentiment Analysis using Hierarchical Deep Learning*, NIPS Workshop on Deep Learning and Representation Learning.

Figure 13: Generate sentence with pretrained model and 95% pruned model
Pretrained: a soccer player in red is running in the field
Pruned: a man in a red shirt **and black and white black shirt** is running through a field

[6] Y. Bengio. 2012 *Practical recommendations for gradient-based training of deep architectures*,

[7] S. Hochreiter and J. Schmidhuber. 1997 *Long short-term memory*, vol. 9 Neural Computation.

[8] S. Han, J. Pool, J. Tran and W. J. Dally 2015 *Learning both weights and connections for efficient neural networks*, NIPS

[9] J. Rauschecker. 1983 *Neuronal mechanisms of developmental plasticity in the cats visual system*, vol. 3 Human neurobiology