Examining the Effects of Pruning on a Convolution Neural Network

Abinash Khanal

Research School of Computer Science, Australian National University
Canberra, Australia
u4594331@anu.edu.au
27 May, 2018

Abstract. This paper investigates the effects of network pruning on model performance. We draw upon the work of EHUD D. KARNIN [1] and implement the "simple procedure" for pruning back-propagation trained networks. On this occasion, we consider an image classification problem, which involves a ten-class image dataset. We use a convolution neural network, as they are the best suited for problems of this kind. After various experiments and analysis, we find that the procedure outlined by [1] does produce a simpler network and one with comparable accuracy. Furthermore, we learn that if we are willing to sacrifice some accuracy, randomly pruning the network can result in much simpler networks, rather than with our pruning procedure.

Keywords: image classification, network pruning, convolutional neural networks, deep learning, CIFAR-10

1 Introduction

The size and complexity of a neural network plays an important role, as resources as limited. The general understanding is that a smaller, simpler network is better. Simpler networks train faster, generalize better and can result in simpler rules. Almost every time one uses a neural network to solve a particular problem, the question of how many hidden layer neurons is always an issue. One often finds using themselves just guessing or using some rule of thumb, which cannot always lead to the best results. It is worth noting that it is better to start off with a larger network rather than a

network too small. A network too small may not actually solve the problem at hand (such as the XOR function). However a network too large can over-fit, leading to poor generalization results. Thus it is always recommended to commence with a more complex network and use a pruning technique to reduce the irrelevant synapses.

The purpose of this investigation is thus to determine, whether a pruned network, based on the simple pruning procedure proposed by [1] can perform just as well as a larger base-case network. Our model is a simple convolutional network (conv-net), however, it could easily be very large, if we increased certain parameters. Thus an effective pruning method would be handy to reduce network sizes.

This image-classification problem was investigated as more complex problems such as driverless cars are built upon this foundation. Understanding the difficulties and techniques that are utilized in a simpler image classification tasks such as this one, can help us approach more complex problems. The dataset provided is already in an elegant format; with target variables encoded appropriately and contains no missing values. The dataset is one of the most common datasets used in the study of computer vision and convolutional neural networks. The wide usage and popularity of this particular dataset means we are able to compare our performance.

2. Method

The dataset chosen is "CIFAR 10" complied by [3]. It contains 60,000 32X32 colour images with 10 classes and 6,000 images per class. The dataset contains 50,000 training images and 10,000 test images. The labels are encoded from 0 to 9; representing the class to which the image belongs to. The classes contained in this study are plane, car, bird, cat, deer, dog, frog, horse, ship and truck. The dataset contains classes that are mutually exclusive. It can be easily obtained by visiting the CS Toronto website. Below are 10 random samples from the dataset:

Figure 1. Random Samples from the Dataset

Keeping the dataset in mind, we model a convolution neural network with the following structure: 2 convolutional layers (Conv 1 and Conv 2), one pooling layer and three fully connected layers (FC1, FC2 and FC3). A dropout layer has been omitted since we want to study the effects of our pruning method, which is in some sense, regularising our network. The process of dropout is very similar to that of pruning. Thus in order to prevent potential interference, we decided to not include a dropout layer.

2.1 Technique

The usual back-propagation algorithm was used to train the convolutional network. For pruning the network, we use the method proposed by [1], which computes the sensitivity of each synapse as:

$$\hat{S}_{ij} = \sum_{0}^{N-1} \left[\Delta w_{ij}(n)\right]^2 \frac{w_{ij}^f}{\eta(w_{ij}^f - w_{ij}^i)}.$$

The method is appealing, as the above computation is based on values that are known throughout the training process. Furthermore, this method does not interfere with the training process itself and does not affect the cost function, as some other proposed methods do. Thus the simplicity and the low overhead of computation make this an ideal process. However, there are problems associated with these methods, which are not considered by [1]. One obvious issue is what happens if the denominator in question is 0, which was the case for our dataset. Thus we have decided to adapt the method and set the sensitivities to 0 for problem weights. Another issue not dealt with by [1] is that of negative sensitivities; we use thresholds and normalization to overcome this. A brief description of the algorithm is shown below:

Initialize an empty array, S, of size equal to the number of synapses for the layer
Let $W_{initial}$ be the initial weights array for this layer.
For n = 0 to number_epochs:
$W_{current}$ = current weights computed in this epoch
$\Delta w(n)$ = weight change in the n-th epoch.
$S = S + \Delta w(n)^2 + \frac{w_{current}}{\eta(w_{current} - w_{initial})}$
Normalise the sensitivities (using any appropriate measure)
Determine the synapses that are below a specified threshold
Set the weights of these synapses to 0

In this method we train the network and in parallel compute the sensitivities for each of the layers. Once training has stopped we can analyze the sensitivities and then decide to prune the weights. A pruning threshold is then used, once the sensitivities are normalized. We normalize to the range (0,1). The threshold determines which units to prune, and once this information is available we prune those weights.

2.2 Validation

To validate the model we use the usual confusion matrix and predictive accuracy. We also study the accuracy within each of the classes to determine the effectiveness of the model.

2.3 Investigations and Experiments

We want to examine how well a simpler model performs in the classification task. Thus we first look at a larger base model determine its performance, and then compare it to the pruned model. Subsequently we prune the weights in the FC1 and FC2 layers of our network. We then perform randomized pruning, on the same layers, to determine whether our pruning method performs better than a randomly pruned network. Finally, we study the effects of different pruning threshold values on model performance.

3. Results

We first investigate the base model. This is the classifier chosen with the parameters shown below. We then investigate what happens when we prune this network. The network parameters that we will use throughout are shown below.

Table 1. Convolution Layer Parameters

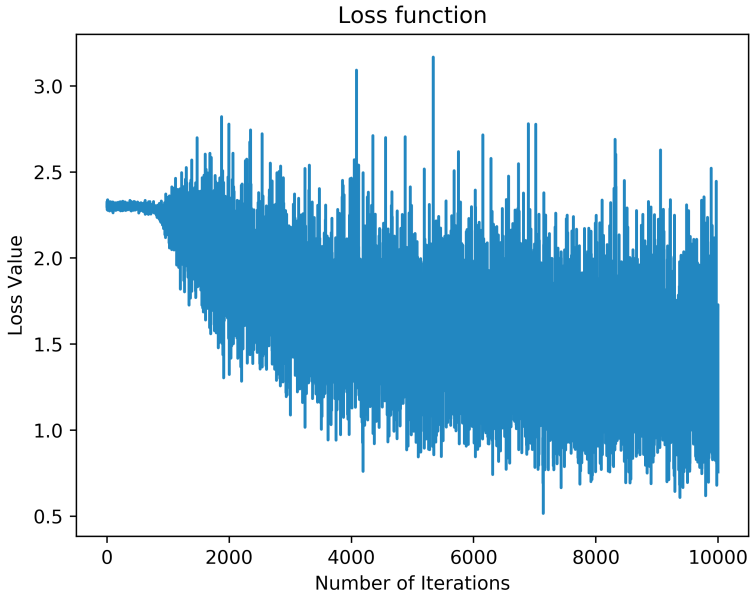
Layer	Input	Output	Kernel Size
Conv 1	3	10	5
Conv 2	10	16	5

Table 2. Fully Connected Layer Parameters

Layer	Input	Output
FC 1	400	120
FC 2	120	84
FC 3	84	10

The learning rate was kept constant to 0.001 throughout the process. This figure was determined after experimenting with various different values. We discern that the highest accuracy obtained was 52%**,** and this occurred with the above learning rate. Thus we will use 52% as our benchmark for our particular study. Current state of the art algorithms have reported accuracies up to 93%, for this dataset. We will not be so ambitious as we are using a very simple convolutional network, for studying pruning.

Figure 2. A plot of the loss function for the base model



The loss function above is fluctuating throughout the training process. However, it does show a decreasing trend and never exceeds the range shown above. Thus, we are obtaining some kind of convergence.

Figure 3. The training confusion matrix for the base model

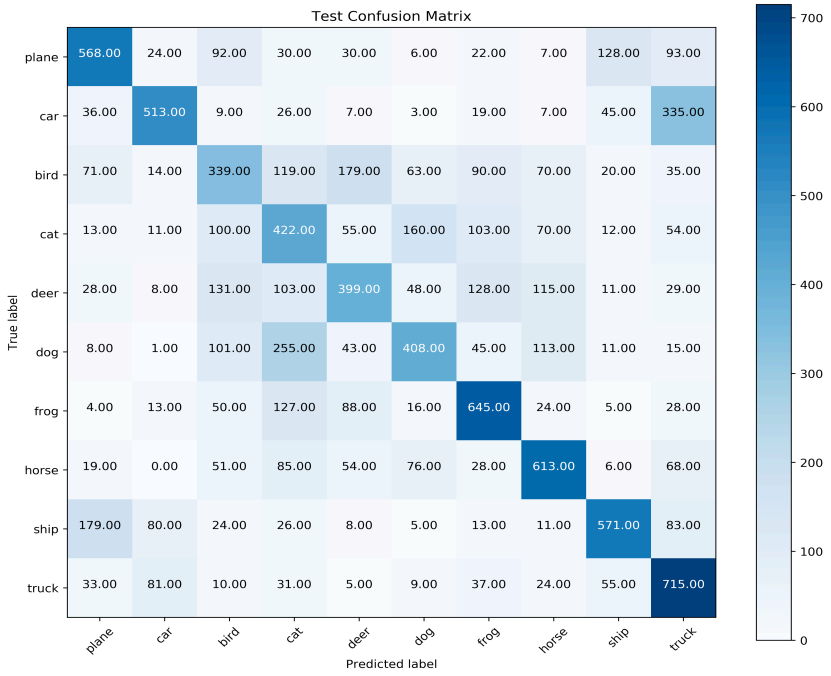
Table 3. The overall Accuracy of the base network on the 10,000 test images is 52 %.

Class	**Plane**	**Car**	**Bird**	**Cat**	**Deer**	**Dog**	**Frog**	**Horse**	**Ship**	**Truck**
Accuracy	56%	51%	33%	42%	39%	40%	64%	61%	57%	71%

Examining the training confusion matrix above, we notice some interesting details. During training, our classifier wrongly classified cats as dogs and dogs as cats. This is anticipated, as the two animals are extremely similar in their appearance. We also

notice that Truck and Frog have the highest accuracy. The test-data confusion matrix, shown below also confirms our observation. Truck is still the most accurately classified class. We notice that the two confusion matrices are very similar.

Figure 4. The test-data confusion matrix for the base model.

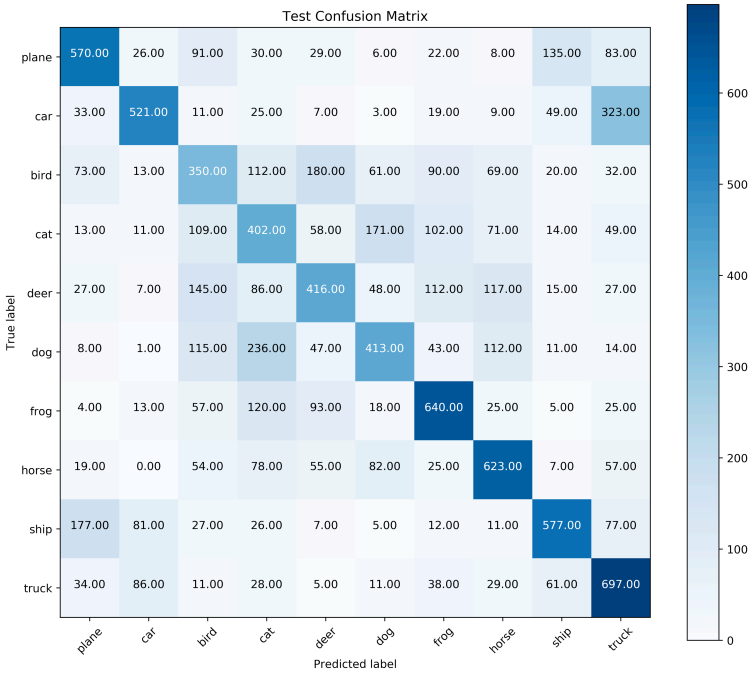


Next, we prune the base case model with a pruning threshold of 0.45. The overall accuracy is still 52%. As a result, we have removed 46 synapses from FC1 and 13 from FC2. While this may not be much, it is still an improvement, particularly because we have not forfeited any accuracy.

Table 4. Comparing the accuracy of the base model and the pruned model.

	Plane	**Car**	**Bird**	**Cat**	**Deer**	**Dog**	**Frog**	**Horse**	**Ship**	**Truck**
Accuracy Base	56%	51%	33%	42%	39%	40%	64%	61%	57%	71%
Accuracy Pruned	57%	52%	35%	40%	41%	401	64%	62%	57%	69%

Figure 4. The test-data confusion matrix for the pruned model.



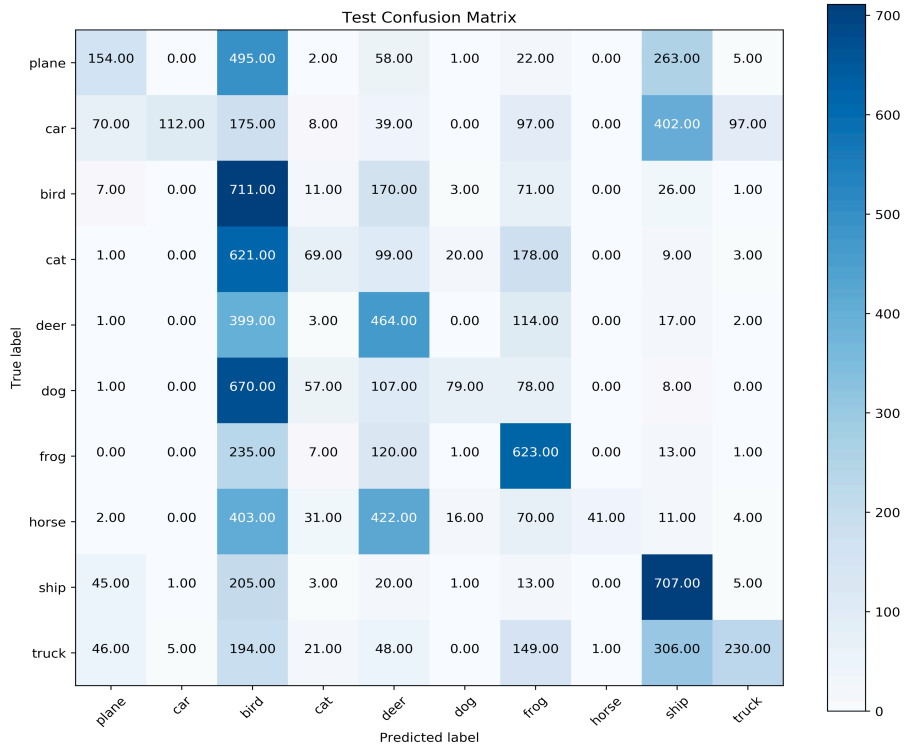
The above is a confusion matrix of the pruned model on the test dataset. Again, we see that we have improved the accuracy across each class as a result of our pruning. This implies that our model is now generalizing better than before.

In order to verify whether our pruning method is correct, we carry out another experiment in which we randomly prune the same model above. To this end, we generate random sensitivities and prune based on these values.

Table 5. Comparing the accuracy of the base, pruned and randomly pruned model.

	Plane	**Car**	**Bird**	**Cat**	**Deer**	**Dog**	**Frog**	**Horse**	**Ship**	**Truck**
Base	56%	51%	33%	42%	39%	40%	64%	61%	57%	71%
Pruned	57%	52%	35%	40%	41%	41%	64%	62%	57%	69%
Random	15%	11%	71%	6%	46%	7%	62%	4%	57%	23%

Figure 5. Test data confusion matrix for the randomly pruned model.



W observe that randomly pruning the network, reduces overall accuracy to 31%. The confusion matrix above also shows that our resulting network is erroneous. We note that although accuracy dropped to 31%, the accuracy is still better than randomly predicting a class (10%). We notice that the class car, plane, cats and dogs are not identified so well anymore. However, ship and frog do not seemed to be affected as much.

Although, random pruning seems to reduce accuracy, we cannot rush to a conclusion just yet. This is because so far, we have used a fixed pruning threshold of 0.45. It may be the case that different threshold values may result in different networks. As with many machine learning problem, certain parameter values tend to work better for certain problems. Thus in order to determine which threshold values work better; we

carry out an experiment where we study the network accuracy for various values of the threshold parameter.

Figure 6. A plot of pruning threshold vs accuracy for the pruned and random pruned models

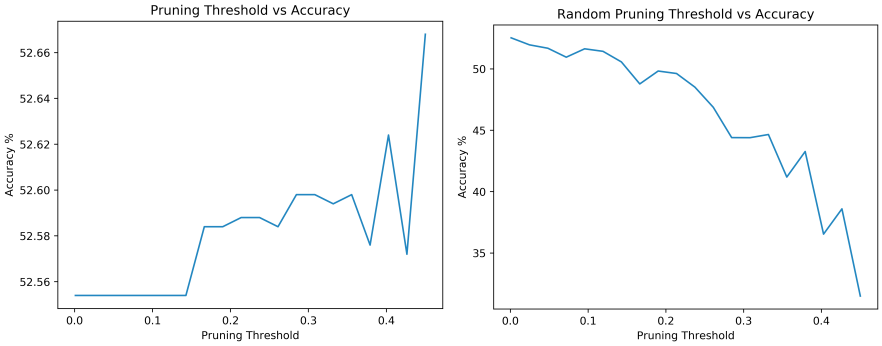


Table 6. An extract of the results for the pruned model.

Pruning Threshold	Test Accuracy	Synapses Removed FC1	Synapses Removed FC2
0.45	52 %	46	13
0.31	52 %	3	3
0.21	52 %	2	2
0.12	52%	1	1
0.01	52%	1	1

Table 7. An extract of the results for the **randomly** pruned model.

Pruning Threshold	Test Accuracy	Synapses Removed FC1	Synapses Removed FC2
0.45	31 %	21,653	4,497
0.31	44 %	14,830	3,139
0.21	49 %	10,326	2,170
0.12	51 %	5,662	1,257
0.01	52%	37	7

We use 20 equally spaced points between 0.45 and 0.001 and generate the above plots. When using our pruning method, the maximum accuracy is 52.66, which occurs at the threshold value of 0.45. In this particular case we remove 46 FC1 synapses and 13 FC2 synapses. Random pruning results suggest that lower threshold values give

better accuracy. The best accuracy occurs when we use a very small threshold, in which we barely remove any synapses.

It is worth perceiving that although random pruning fluctuates more in its accuracy, it does sometimes produce an accuracy of up to 51% percent and removes 2,332 FC1 synapses and 505 FC2 synapses. So it tends to simplify the network significantly on certain instances. After additional verification, we confirm that random pruning does indeed achieve this. In Comparison, our pruning procedure tends to not remove as many synapses, but keeps the accuracy somewhat constant. Furthermore, with random pruning, we obtain 52% accuracy only when we remove very few synapses. Thus both methods agree on the 52 %accuracy if we prune a similar number of synapses.

4. Conclusion

The purpose of this investigation was to determine the effects of pruning a network by using the "simple procedure", proposed by [1]. We conclude by realizing that for this particular dataset, the pruning method produces a slightly a simpler network. It does so by maintaining the 52% accuracy. We further recognize that if we are willing to sacrifice some accuracy, we can obtain much smaller networks by using a random pruning method. The random pruning technique should be tested on other datasets, as it may have performed well in this particular dataset. Amongst our research, we failed to find convincing evidence against the procedure proposed by [1].

There are numerous improvements we can make to our study. The algorithm used, only allowed us to simplify the network after we had already trained it. Even for our current dataset, training and experimentation times were starting to become an issue. It would be ideal if the algorithm told us which weights to prune and when to stop training, simultaneously as the training is happening. This would then result in the optimal network. Another investigation that we did not carry out is pruning the convolutional layer connections. The approach would be identical, however it may be possible to simplify our model even further. Another area worth exploring is the use of genetic and evolutionary algorithms in pruning networks. Perhaps considering advanced techniques such Fractional Max-Pooling, Shakedrop Regularization and Regularized Evolution may help us learn more or dismiss our simple pruning procedure. It is also worth comparing our pruning procedure to the results from dropout layers that come naturally with convolutional networks.

References

1. Karnin, ED: A simple procedure for pruning back-propagation trained neural networks. IEEE Transactions on Neural Networks, vol 1., pp. 239-242, 1990.
2. T.D. Gedeon & D. Harris: Network Reduction Techniques. Department of Computer Science, and Brunel University.
3. Krizhevsky , Alex: Learning Multiple Layers of Features from Tiny Images, 2009.