Reducing Network Size by Pruning Multiple Filters per Iteration in Deep Convolutional Neural Networks

Jop Heuvelmans (1258456)
Jheronimus Academy of Data Science
j.w.m.heuvelmans@tilburguniversity.edu

Abstract

Transfer learning is an important topic within the field of Convolutional Neural Networks. Deep, complex networks are trained and subsequently pruned to reduce the size and complexity of the network. Traditional pruning methods mostly reduce the number of parameters, which does not reduce inference cost much. Molchanov et al. (2017) propose a different strategy; to prune entire convolutional filters, which does indeed significantly reduce inference cost. However, their approach prunes only one filter per iteration and is therefore very computationally expensive. In this study, the effect of pruning multiple filters per iteration on model accuracy and running time is investigated. In four experiments, the number of filters pruned per iteration is gradually increased using a pre-trained VGG16 network on a simple image classification problem. The results show that one can safely prune up to 25% of the network's size per iteration without significantly effecting model accuracy. It is concluded that pruning multiple filters per iteration is a computationally effective approach to reduce model size and inference cost.

1 Introduction

Convolutional neural networks (CNN) are extensively used in several diverse classification problems including speech and image recognition (Simonyan and Zisserman, 2014; Krizhevsky et. al, 2012; Hinton et al., 2012). The general trend of the past few years has been to build deeper and more complex networks, which have the capacity to tackle increasingly difficult problems. For less complex problems, transfer learning is used. Deep, complex networks trained on a very large labeled vision dataset, such as images from ImageNet (Russakovsky *et al.*, 2015). Subsequently, these complex networks are pruned: redundant and duplicate connections are removed to achieve state-of-the-art accuracy.

Much research has been done in the area of network pruning, but the majority of this research has focused on iteratively eliminating weights with small magnitudes and then retraining up until a predefined accuracy loss is established (Han et al., 2015; 2016). However, for CNNs, pruning parameters does not necessarily reduce inference cost. In CNNs, most parameters reside in the fully connected layers. Pruning techniques such as described by Han et al. (2015) focus on eliminating individual parameters and hence prune much of the fully connected layers. Yet inference cost is affected most by the evaluation of convolutional layers, which are pruned to a lesser extent using the aforementioned techniques. High inference costs pose a problem for embedded sensors and mobile devices, which require computational efficiency (Szegedy et al. 2015).

Molchanov et al. (2017) have proposed a different pruning strategy. For each convolutional filter, they compute the change in the loss function if this filter were omitted. They subsequently rank these changes in the network cost and prune whichever filter results in the lowest change. Next, they fine-tune the network and repeat the process, removing one filter at a time until some stoppage criterion is reached. Although this approach yields promising results (more than 10× theoretical reduction in adapted 3D-convolutional filters with a small drop in accuracy in a recurrent gesture classifier), pruning one filter per iteration is very computationally expensive and infeasible for lesser equipped users.

In this paper, the research of Molchanov et al. (2017) will be extended by pruning multiple filters per iteration and observing the effect on accuracy and running time. This research proposes to find a trade-off between pruning size, accuracy and running time to suit the needs of lesser equipped users. The following research question will be addressed: *"What is the effect of pruning multiple filters per iteration on accuracy and running time?"*

To answer this question, multiple experiments will be performed using a pre-trained VGG16 network and the Cats vs. Dogs dataset published on Kaggle (2019). The combination of the model and dataset is suitable to capture the benefits of transfer learning, because the model is trained to classify images, and this is exactly the problem at hand in the dataset. Moreover, this dataset is comparable to the ImageNet image classification problem for which VGG16 was originally designed. Throughout the

experiments, the number of filters pruned per iteration will be increased and subsequently the effect on accuracy and running time are investigated.

The remainder of this paper is structured as follows. In section 2, the method is outlined in five steps. In section 3, the experimental set-up is expanded by providing information on the dataset, the hyper parameter settings and the evaluation criteria. In section 4 the results of the experiments are discussed. Discussions are provided in section 5. Finally, Section 6 introduces the future work and concludes.

2 Method

This study examines the effect of pruning multiple filters per iterations using a pre-trained VGG16 network. The problem investigated is the Kaggle Dogs vs. Cats classification task (Kaggle, 2019).

A pre-trained VGG16 is chosen because it is a very powerful and deep convolutional network, with many convolutional filters (4224 filters). In this network about 90% of all weights reside within the fully connected layers, but these account for only 1% of the total floating-point operations (FLOPs). Therefore, traditional pruning methods, which focus on pruning weights, will not significantly reduce the number of FLOPs and thus will not reduce inference cost much. In this experiment, convolutional filters are pruned, which should greatly reduce the number of FLOPs (Molchanov et al., 2016).

In this study, four experiments are executed and for each experiment the number of filters pruned per iteration is increased. In step 0, before performing the experiments, a pre-trained VGG16 network is retrained for the problem at hand. Each experiment will subsequently involve four steps, in which step 1, 2 and 3 are repeated until a stoppage criterion is met and step 4 is executed to fine-tune the final model. Figure 1 provides a graphical representation of the method.

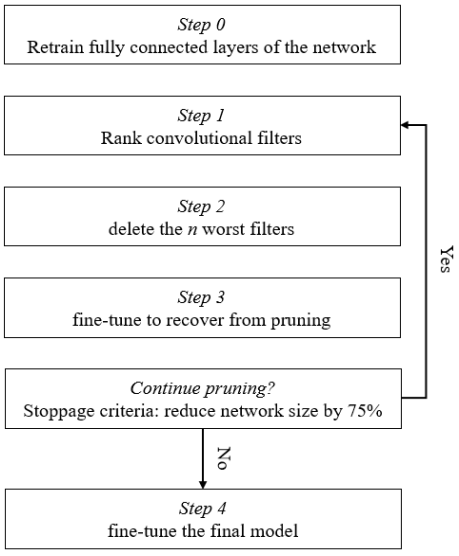

Figure 1: graphical representation of the experimental setup.

2.1 Step 0: retrain fully connected layers of the VGG16 network

A pre-trained VGG16 model is loaded and subsequently the fully connected layers are dropped. The model is then extended with the following layers: dropout (0.5), fully connected layer, ReLU activation, dropout (0.5), fully connected layer, ReLU activation and finally another fully connected layer with two output neurons. A high dropout rate is chosen to prevent overfitting, since VGG16 is a very deep convolutional network which is prone to overfitting.

After adding the new layers, the convolutional layers are frozen, and the model is retrained for ten epochs. Subsequently the accuracy on the test set is computed and stored.

2.2 Step 1: rank the convolutional filters

To determine which filters are redundant and can be safely pruned, a ranking is needed. To compute this ranking, a forward and backward pass is performed on the dataset. This allows to compute the gradient and the activations for the convolutional layers. Next, a filter ranking is computed based on the Taylor criterion proposed by Molchanov et al. (2016). This criterion uses the first-degree Taylor polynomial.

2.3 Step 2: delete the *n* worst filters

In this step the actual pruning takes place. Based on the ranking retrieved from step 1, the worst *n* filters are pruned. Since VGG16 has 4224 convolutional filters, *n* (the number of filters pruned per iteration) will be increased throughout the various experiments from 132, 264, 528 to 1056. These numbers represent 3.125%, 6.25%, 12.5% and 25% of the total number of filters respectively. This allows for comparable model sizes between the experiments.

2.4 Step 3: fine-tune to recover from pruning

In step 3 the network is retrained such that it can recover from the pruning operations. All layers are unfrozen, and the network is fine-tuned for five epochs.

Step 1 through 3 will be repeated until a stoppage criterion is met. In this study, the network will be pruned until the number of filters has been reduced by 75% to 1056 filters. This will take between 3 and 24 iterations, depending on the number of filters pruned per iteration.

2.5 Step 4: fine-tune the final model

After having finished pruning, the model is once more fined tuned for ten epochs to achieve the final model.

3 Experimental setup

In this section, the experimental set-up will be described. First, details about the data and pre-processing is provided. Next, the experiments are further detailed. Subsequently the hyper parameter settings are described and finally the evaluation criteria are introduced.

3.1 Data

The experiments are performed on the Dogs vs. Cats dataset as provided by Kaggle (2019). The original dataset consists of 25000 labeled training examples (12500 pictures of cats, 12500 pictures of dogs) and a test dataset. However, since this dataset originated as a Kaggle competition, labels are not provided for the test set and therefore rendered useless for this experiment.

The training set is reduced to 1000 samples due to computational constraints. Similarly, the test set for this study consists of 1000 subsamples of the original training set. Both the training and test sets are balanced.

Since VGG16 is optimized for handling images of size 224 x 244 pixels, the images of both the train and test set are resized to 256 pixels and subsequently center cropped to 224 pixels. Next, the images are transformed to a tensor and normalized. The train and test set are separately normalized to avoid data-leakage.

3.2 Experiments

The procedure of the experiments is outlined in section 2. In table 1, the number of filters after each iteration are presented for the different experiments. For each experiment, test accuracy and running time will be tracked. Please note that the number of filters pruned per iteration is used as a reference to the corresponding experiments.

Iteration	132	264	528	1056
0	4224	4224	4224	4224
1	4092	3960	3696	3168
2	3960	3696	3168	2112
3	3828	3432	2640	1056
4	3696	3168	2112	
5	3564	2904	1584	
6	3432	2640	1056	
…	…	…		
11	2772	1320		
12	2640	1056		
…	…			
23	1188			
24	1056			

Table 1: number of filters left after each iteration for the different experiments

3.3 Hyper parameter settings

In table 2, the hyper parameter settings for the different steps of the experiment are provided.

Step	0	3	4
Description	Retraining FC layers	Fine-tune after 1 iteration	Fine-tune after all iterations
Learning rate	0.0001	0.001	0.001
Momentum	0.9	0.9	0.9
Epochs	10	5	10
Batch size	16	16	16

Table 2: hyper parameter settings of experiment steps

3.4 Evaluation criterion

To evaluate the performance of the experiments, the accuracy after each iteration and of the final model is tracked. Additionally, for each experiment and for every step the running time is tracked.

3.5 Environment

The experiments were performed on the cloud, in a Floydhub workspace using a Tesla K80 GPU with 12 GB Memory, 61 GB RAM and 100 GB SSD.

4. Results

The main results of the experiments can be found in table 3.

	132	264	528	1056
Accuracy	0.968	0.98	0.972	0.966
Accuracy loss	-1.43%	-0.2%	-1.02%	-1.63%
Running time (seconds)	5063.5	2896.83	1702.35	1139.59

Table 3: result of the experiments. The numbers in the heading indicate the number of filters pruned per iteration.

In figure 2 the accuracy throughout the experiments is plotted.

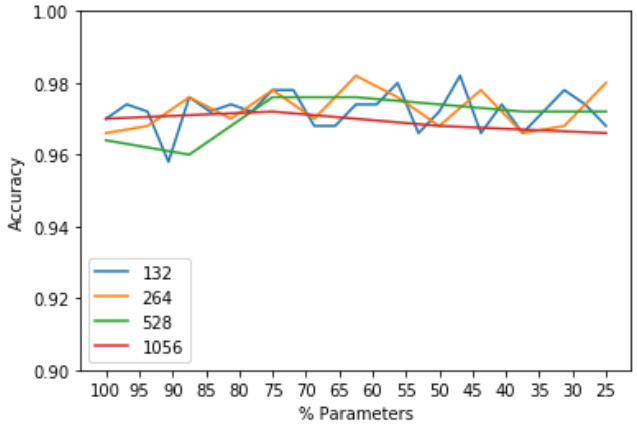

Figure 2: test accuracy throughout experiments

Firstly, it can be observed that the original test accuracy (before pruning) of all models exceeds 96%, which is good considering the low number of training examples and epochs. These results are not surprising considering VGG16 is a very deep neural convolution network optimized for image classification and the problem at hand is relatively easy (binary classification).

Second, the accuracy of the models seems to be somewhat instable. This is probably due to the limited number of training examples and fine-tuning epochs. It is expected that when the network is given more training examples and is allowed to recover longer after each iteration, the accuracy throughout the experiments will become more stable.

Third, it can be observed that the accuracy loss due to pruning is limited. The biggest accuracy loss is 1.63%.

Interestingly, the accuracy loss of pruning 264 and 528 filters per iteration is less than the accuracy loss when pruning 132 filters per iteration. However, this is probably due to the limitations described above.

Below in figure 3, the running times for the various experiments and steps are plotted. It is observed that when fewer filters per iteration are pruned, the total running time increases. This effect is caused by the fact that more iterations are needed to reduce the network to the desired size when pruning fewer filters per iteration. The network has to recover after every pruning iteration, which requires retraining for a predefined number of epochs (5 in this case). Furthermore, the filters have to be ranked in each iteration. Contrastingly, the actual pruning stage and the final fine-tuning stage (once the network has been sufficiently reduced) take roughly equally long for all experiments.

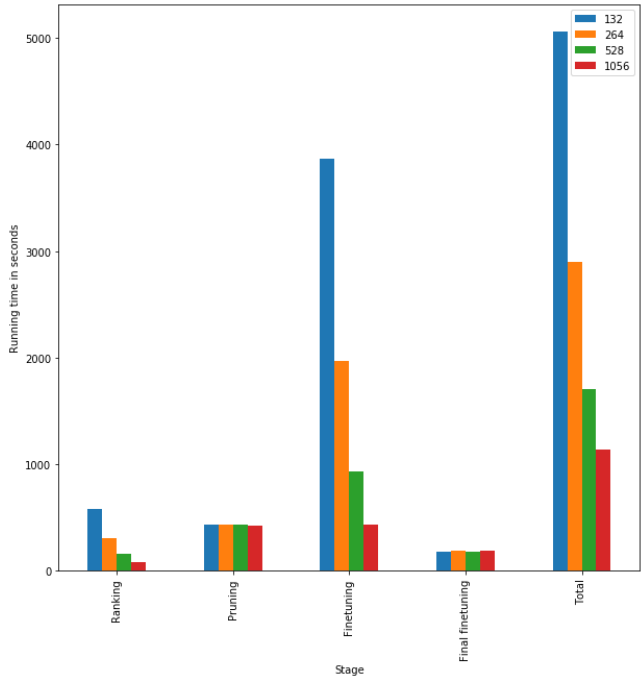

Figure 3: running times of the experiments

Finally, it is observed that the network size is reduced from 524MB to 107MB, 106MB, 102MB and 100MB for the respective experiments.

5. Discussion

The results show that for this problem (Cats vs. Dogs) one can safely prune the network by 75% by pruning multiple convolutional filters per iteration and retraining the network after each iteration to recover from pruning. This insight offers three benefits.

First, by pruning the network, the network's size can be reduced by up to 80%. Second, it is expected that inference cost decreases significantly when using these pruned models. Although this is not explicitly tested in this research, the results from Molchanov et al. (2017) most likely still hold, since the number of convolutional filters in the network is significantly reduced. It is these convolutional filters which heavily increase inference cost.

The two aforementioned benefits were also noted by Molchanov et al. (2017). However, a third additional benefit discovered in this research is that pruning multiple filters per iteration significantly speeds up the pruning process with minimal accuracy loss. For instance, if one prunes 1024 per iteration instead of 132, the running time reduces by 77.5% .

The strengths of this study include its simplicity and relevance. The findings of this study can help data scientists, researchers and students on a day-to-day basis to reduce network size and inference cost in a computationally efficient manner. Especially in products such as mobile devices, where big network size and high inference cost pose a problem, the computational and memory efficiency of pruned models is beneficial.

Some limitations of this study should be addressed. First, due to computational constraints of the researcher, the number of training and test samples was restricted to 500 per category per set. Similarly, the number of epochs is deliberately kept low to reduce computational complexity. This negatively affects the internal validity of the experiments.

Second, the network's size could be reduced even more to gain additional insight. In this research, it was chosen to prune the network to 25% of its original size. However, due to computational and time constraints imposed on the researcher, it was not feasible to experiment with other stoppage criteria.

6. Conclusions and further work

In conclusion, pruning pre-trained convolutional neural networks is an effective approach to reduce network size and inference cost, as demonstrated by Molchanov et al. (2017). Furthermore, pruning multiple filters per pruning iteration significantly speeds up the pruning process, whilst maintaining adequate accuracy. Interestingly, for this problem and model, one can safely prune 25% of the model per iteration and still maintain acceptable accuracy (96.6%).

Multiple areas of future research should be identified. First, to increase the internal validity of this paper, future studies should incorporate more samples and/or training epochs, which increases the robustness of the results. Second, one could experiment with different stoppage criteria to assess the impact on running time and accuracy. Two different kinds of stoppage criterion can be identified, namely (1) prune until the network has been reduced to a predefined size or (2) prune until a predefined accuracy loss is observed. Another area of future research is the generalization of these results to other models and datasets. In this research the Cats vs. Dogs problem is solved using a pre-trained VGG16 network. It would be interesting to see how the results generalize to other models (such as AlexNet or ResNet). Similarly, in future research one could perform similar experiments using other datasets, particularly multiclass classification, and see whether the result of this study generalize beyond this dataset and network.

Acknowledgments

The code for these experiments was based on https://github.com/jacobgil/pytorch-pruning, created by GitHub user jacobgil. The code has been adjusted to run with an updated version of PyTorch and to match the experiments outlined in this paper.

References

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. In Advances in neural information processing systems (pp. 1135-1143).

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A. R., Jaitly, N., ... & Sainath, T. (2012). Deep neural networks for acoustic modeling in speech recognition. IEEE Signal processing magazine, 29.

Kaggle. (2019, April 9). Dogs vs. Cats. Retrieved from Kaggle.com: https://www.kaggle.com/c/dogs-vs-cats/

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).