The Pruning Methods for feed-forward network and Convolutional Neural Network

Abstract. Pruning neural network is a series of techniques that remove some of the model's parameters out of the training. It is a way to decrease neural network's structure on topology aspect, and also a way to increase training efficiency. In this work, the pruning methods is applied on a deep neural network, for the convolutional layer, a ranking method is applied to find filters that not contribute much to the performance, and for the linear layer, the pruning method is based on the average value the hidden layer's weights. The average values is the threshold to evaluate the weights' contribution. The model is implied on the Kaggle Cats vs Dogs data set to perform a classification function, and the pruning methods will be discussed rely on the model's performance.

Keywords: Neural Network Pruning, Neural Topology, Feed-Forward Network, Convolutional network

1 Introduction

Pruning neural networks is first introduced in 1990 by LeCun, et al [1]. As they stated, some parameters in the network could be redundant and give less contribute to the output. Therefore, by pruning some parameters, the network could be smaller on topology aspect. Pruning neural networks usually be considered as a regular method to improve training efficiency and generalization of the model. With the improvement of deep neural network techniques in recent years, the convolutional neural network has become a very popular method to solve many computer vision applications and on some natural language processing questions. such as object classification and localization, entity detection, and sentiment detection both on images and captions. However, many deep neural networks have complex topology structure, because usually deep models are more possible to get good performance than simply models. Hence, it is more possible the deep models contain the parameters which are redundant. Besides, the design of deep neural network is kind of empiric work, it is difficult to keep the deep models without redundant weights. Therefore, it is necessary to apply pruning techniques on the deep neural network models.

In this paper, a feed-forward network is built for a classification problem on a data set including noise data, then the neural network pruning techniques have implemented and discussed on this model. This method [1] based on the calculating the distinctiveness of two units' weights, if the angle between two vectors is range, then one of them can be removed. Another pruning method [2][3] is also implied to be the matched one, this method measures the weights' value with a threshold generated based on all this layer unit's weights, keep the connection bigger than the threshold and remove the connection lower. For the convolutional neural network is applied for a image classification problem. then the neural network pruning techniques based on the evaluating the activation values have implemented on both convolutional layers and the linear layers.

2 Method

In this section, we first introduce the data set we use and the neural network structure designed on the image classification question, then the pruning method will be explained.

2.1 The image classification model

To choose the training data set for feed-forward network, it is better that the data including some noise data. More specific, the noise data usually not relevant to the target the network need to achieve, it might damage the unit's functionality to capture the crucial features. Copy the training data and add them to the training data set influence the data distribution. Although this method creates the redundant data, and could lead some of the hidden unit to perform the fundament functions, the network is more likely to have bias. Therefore, in this work we choose the Waveform Database set [6], because it originally contains noisy attribute for each instance. In detail, Waveform Database is a physical area experiment data about wave 's form, it has 5000 instances, and each of them have 40 attributes, all the

value is the real number. This data set is designed for classification problem, with 3 classes of waves, all 40 attributes include noise in last 19 attributes with mean 0 and variance 1. The Waveform Database is easy to use, it does not need preprocess. In summary, Waveform Database is suitable data for our work.

Then in this paragraph we give the detail of the feed-forward models. The feed forward network's input layer has the same number of the units with the Waveform data attributes, it could be 40 on the data with noise or 21 on the data without noise. The hidden layer has 50 units, and the output units is 3 as the class of Waveform classification task is 3. The activation function is sigmoid for the input and hidden layer, this is mainly because this activation function ‘s output is more convenient for the first pruning method need to calculate the vector angle. During developing, we find that unit with ReLU can achieve more good performance, but it is not suitable for out pruning methods as it is not easy to normalize its output.

Image classification is a very popular topic in computer vision area, in this work, the Kaggle Dogs vs. Cats dataset is used. The images are labelled by the folder’s name, which means that the folder named with “cats” contains all the cat image, and the same for the folder named with “dog”. Because the original data set is huge, and this work’s aim is to apply the pruning methods on deep neural networks, rather than build a well performed model for image classification tasks, therefore we selected 1000 image for both cats and dogs, totally 2000 images as the training set, and 200 images for each class as the testing set.

The training and testing data must be processed into a good format that can be accepted by PyTorch and Numpy. First, the images in Kaggle Dogs vs. Cats dataset are all be resized in 32 * 32 pixels images, as the original images are all in different size. The images are saved using RGB Values because we consider that is could keep more information than greyscale images. The functions from PIL image library are used to read the images from disk, the labelling operation has been done based on the folder names, some of the training images are randomly chosen and the flipped. The flipped image the training set is shuffled and divided into batches, each batch contains 32 images.

Convolutional neural network (CNN) has widely applied and achieved highly performance on computer vision tasks, it is used as the feature extractor in this model. The first temporal convolutional layer with 2 dimensions is applied to get the input with 3 channels, each channel gets the value of Red, Green and Blue. The kernel size is 5 * 5 based on some other researchers’ experience. There are 16 output channels for the first convolutional layer. The ReLU is used as the activation function. It is method that thresholds any number less than 0 is changed to zero, while others are kept the same.

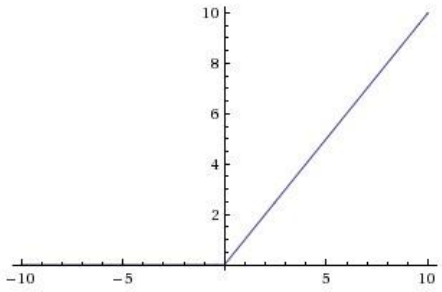

Figure 1, the ReLU Function

Max-pooling is a technique used to reduce the dimensions of an image. The max pooling layer taking the maximum pixel value within a filter. This also helps reduce overfitting and makes the model more generic. To capture the strongest local features, the max pooling layer is then applied to the output of the first Convolutional laye. the filter is defined as f, and the filter size is defined as *d*max. The max operation calculates the local feature value within each channel, the output of max pooling layer is defined as:

$$\max(z_{ij}) = \sum_{a=0}^{dmax-1} \sum_{b=0}^{dmax-1} f_{ab}\, z_{(i+a)(j+b)}$$

Next, another convolutional layer with ReLU activation is applied to extract the more abstract feature based on the first layer’s output. Then, the fully connected layers consist with a input layer, a hidden layers and a output layer are added on. For these linear layers, we will also have applied the pruning methods we used in the feed-forward model.

2.2 The neural network pruning

In this work, some pruning methods be applied on the feed-forward model. The pruning methods is not considered on the first linear layer and the output layer, because both of these layer’s unit lost will highly influence their function. Therefore, we applied the pruning method only on the hidden layer. The hidden layer captures the abstract feature or

data distribution features from input layer or the previous hidden layer, with the more hidden layer and unit in a neural network model, the more difficult to measure the contribution of them, and how they co-operate with other units or layers. We also applied the two pruning methods mentioned in the previous work. the first method is simply based by calculating the distinctiveness of units, if some two units have too close distance or they are in different direction, is means that they functionality is similar or totally contradicted.

This method is specifically get the hidden units' output vector, as the units in the same layer have the same presentation and dimensionality of the vector. Each units' vector can be considered as its functionality representation. In this model, these is no clone units, and the units are all in similar presentation, the dropout method is also not applied. Therefore, the unit's similarity is only on the vector value level. This setting keeps the method clear and straightforward. The method to measure the distinctiveness between hidden units is calculate the angle of their vectors in pattern space. As all the hidden units' activation function is sigmoid, the interval of the activations is [0,1], corresponding the vector angle is representing in the angular range of 0-180 °, not 0-90 ° or 0-360 °. The angle around 90 ° is considered that these two units' functionality is different without any redundant. On the contrary, when the angle between two hidden units vector is between 0 ~ 15° or 165° ~ 180°, in the pattern space, these angle value means the two vector is close or complementary one, we can sufficiently consider that these two units have high function similarity, one of them need to be removed.

In this work, the removed units' weight vector is not being added to the unit which been kept in the network, even this two units' distinctiveness is small, but the inner value of their vector cannot simply have considered with high similar. Add the two vectors up could double the remain units' vector, or counteract some values. The distinctiveness method measure units' functionality similarity, the vector's value similar or not is proposed to be not so important for pruning the network. Therefore, the units are removed only from the back-propagation operation. The remain unit's vector value is not being changed.

The pruning techniques on our neural networks should not be applied at the beginning of the training. In that early stage, the weights and bias of the network is just initiated (usually with small random real numbers), the parameters in each neuron is not be trained to perform their specific functions. Therefore, in this work, our feed forward network is first trained on the training data set, but the number of training epoch is not too big, as there is no need to train the model too well fitted (sometimes maybe overfitted) on the training data set, the model sufficient trained until the rate of its performance increasing start reducing. This model is denoted as the pre-trained model. The pre-trained model is saved both its structure summary and parameters. For the next stage, the pruning method is applied on the pre-trained model, to remove the redundant units away from the back-propagation operation, and set their weights into zero. Conceptually, this operation can be considered that this unit is removed from the network, as it has zero output, and its weight will not be update after pruning operation. In the last stage, the pruned model is loaded and the whole model is fine-tuned on the training data set. Specifically, after applied the pruning methods, the model lost some unit, the network might be damaged from it, it is necessary to train the neurons to 'recover' from the data. The pruned model also be tested on the test set. the performance of the model pruned will be compared at the discussion section. The pruning method could be summarized as an Iterative circle of operations: training, pruning, fine-tune until the model achieve good performance or no more pruning operation could be applied on. This circle is shown in Figure 2.

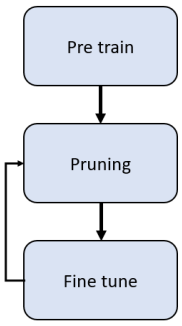

Figure 2, The circle of pre-training, pruning, fine-tune

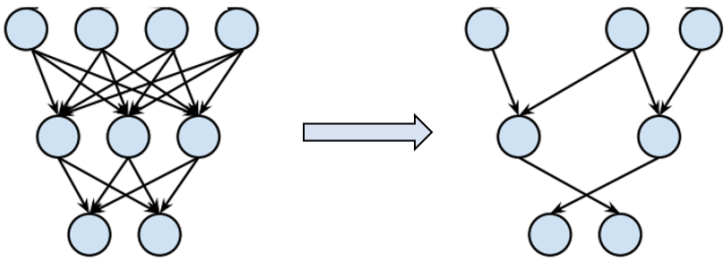

Figure 3, The visulization of pruning feed forward network

By research about the related work, many other method of pruning techniques can be found. In this work, one of the methods is picked to be the matched methods. This method is introduced by Han et al in 2015 [2]. The briefly methods is to remove the connection between units. More specific, in each hidden layer, the mean average over of weight is calculated, and the result is set as a threshold. Then after training the model for some epoch (not too well trained as mentioned at last paragraph), All the units' weights is compared with the threshold, if the weight is lower, then it is "removed" on topology concept, this operation is visualized in Figure 2. the remove operation is similar as the operation mentioned before, set the weight to zero, and keep them out of the back-propagation operation. The model need to be retrained, to tuning the remain units weight.

The pruning method for convolutional filters could be done by reducing the weights in each filter, or by deleting a specific dimension of a single kernel. By this kind of methods, the filters will be sparse at the end of pruning, but it's

not trivial the get a computational speed up. The recent paper also stated when entire filters are pruned, the model suffer from the "Structured sparsity" [4]. Another kind of pruning methods focus on pruning entire filters in convolutional layers. In this work, we choice to pruning the convolutional layer by remove the filters that not in a good active state. The pruning techniques we used on convolutional layers is based on the ReLU activation functions to evaluate the filters. One of the reasons why ReLU activation is very popular is the sparsity in activation that is induced, this feature allows convolutional layers to act as feature extractors. Therefore, it is considerable that to evaluate the filters by its activation function's output value. This means if an activation value is small than a particular value, this particular value can be also considered as the threshold, so in this work we considered the pruning method on convolutional filters could be similar with the first method for the linear layer we mentioned before. this feature extractors can be considered as not important enough for the task. In this work, we calculate each filter's weights value, sum them up to consider it as the kind of "score" of the filter's activation values. The filter with lowest value then be deleted from the model. Obviously, the model need to fine-tuned, to trained the remain filters to keep the model's function.

In this work the two pruning methods are applied on feed-forward network, two stated different way to measure the units 'importance, pruning the fully connected network to a sparse one. In addition, the same pruning method is applied for the linear layer in the convolutional model, and a pruning method conceptual similar is applied on the convolutional layers. In next section, the result will be shown and discussed.

3 Results and Discussion

The Table 1 show the accuracy of the two-model introduced in section 2, the "Dist. Model" is the model remove units based on the distinctiveness of units, the "Threshold Model" is the model remove the units' connection by compare the weight with the threshold value. The top two row shows the performance trained on the training data with noise, the button two row is the result on the data without noise data. There is not too much difference about model performance on the data with or without noise, but the inner difference of pruning operation will discuss based on Table 2 in next paragraph. On the right-hand side is the performance of two baseline model delivered before 2000. The classifier based on first order logic gave the most bad performance [7], today we could state that model based on logic theory is less flexible and powerful that the neural network techniques the model based on decision tree and Naive-Bayes method [8] is very popular at that time, it achieve acceptable performance on the Waveform data. However, both baseline models have no result on the Waveform data without noise, hence we do not discuss about that part. In summary, it can have been seen that the both pruned model get the performance increase about 3% on both the data with and without noise data, and about 6% improvement when compare with the baseline methods. With more detail, during both training stage after the pruning the pre-trained model, the training accuracy has drop a little bit at beginning, about 1.5%. this observation proved that the pruning operation on the model really damage it, but after the fine-tune, the both model's performance have reach a higher point before the pruning. For this work, the pruning operation help the model on performance.

		Dist. Model		Threshold Model		Decision Tree		Logic Classifiers	
Attr.	stage	Acc.	Error	Acc.	Error	Acc.	Error	Acc.	Error
40	Pre-train	82.70	17.30	81.33	18.67	80.5	19.5	78.93	21.07
40	fine-tune	85.83	14.17	85.00	15.00	\	\	\	\
21	Pre-train	82.10	17.90	83.02	16.98	\	\	\	\
21	fine-tune	84.26	15.74	85.98	14.02	\	\	\	\

Table 1, The result of the two methods from this paper in column 3 & 4, and two baseline models in column 5 & 6

	Dist. Model	Threshold Model
Attribute Num	Removed prec.	Removed prec.
40	16.00%	45.00%
21	13.41%	49.43%

Table 2, The percentage of units removed from the network

The Table 2 shows the percentage of how many unit have "removed" after pruning, it is obvious that the pruning method based on calculate units' weight vector's angle much less than the method based on remove units which weights lower than the threshold. Combine with the result from the Table 1. it can be seen that Threshold Model is more good at locate the redundant units, as nearly half of the hidden units is not so functional in the model. On interesting point is that, even distinctiveness model removed less units, but the performance between two models have not shown a gap. The explanation could be that even some hidden units is redundant, but they have no damage for the model's performance, However, it cannot state that these units do not reduce the model's training efficiency, as the training data set is not big enough, and modern computer is already get the compute ability to train some small neural networks.

Therefore, in this work the model based on threshold method can find more removeable units, but the influence of those units could not be shown in this work both in performance aspect and efficiency aspect.

	Dist. Model	Threshold Model	VGG16
Test Accuracy	65.00%	61.07%	85.76%
Removed prec. For linear	23.51%	49.43%	\
Removed prec. For Conv	43.12%	46.57%	\

Table 3, The result of convolutional neural network

Table 3 show the result for the convolutional model on Kaggle Cats vs Dogs' sub data set, it also shows that threshold methods is still work good on the linear layers within the convolutional models. In the third line "Removed prec. For Conv" shows the percentage of the filter removed in the convolutional layers, we can see that even the model is not too deep, but nearly half of the filters are not so contributive. This result proved that pruning techniques is necessary and applicable for the convolutional models. The VGG16 is a popular pre-trained model for image classification problems, we applied VGG16 on our sub Kaggle Cats vs Dogs data set, It performed good. But too be honest, the PC we used to trained models is not some capable for training VGG 16, we only applied the pre-trained model on testing set to get a baseline. Obviously, our simple convolutional model is not powerful like VGG16 model. However, we could see that the potential of pruning convolutional models, maybe the VGG16 model could also be pruned.

In this work, the feed forward network is simple enough to measure the distinctiveness between them within a hidden layer, as the units not strongly functionally combined, and we can also find that some of them are canceling each other. They could be removed from the network without too much damage to the performance. When discuss about the modern deep learning neural networks, the structure is more complex, the inner function of each unit is also powerful and complex, such as the LSTM cell with input gate, forget gate, etc. It is more difficult to develop an accurate method to measure these unit's contribution, badness, and distinctiveness. We have tried to apply the pruning method on convolutional models, the performance is not remarkable, but is proved that is methods have some potential. Besides, the procedure of developing the pruning methods could be even more expensive than develop the model itself. The pruning operation might also reduce the training efficiency several folds.

4 Conclusion and Future Work

In this work, a feed forward network is built for a classification task on the Waveform dataset, two pruning methods have been applied on this network to locate the redundant hidden layer's units. The first method based on calculate the vector angle between two units' weight to measures the distinctiveness of them, the unit will be remove if its functionality is similar to another one. The second method based on the whole ability of the hidden layer, it calculates the average value of all the hidden units' weight within that layer, take that value as the threshold to choose which units' weight have less contribution and should be removed. The model based on these two methods have achieve similar performance on the Waveform dataset's classification tasks, but the pruning method based on the threshold show more ability to locate the hidden units in this work.

There are many pruning techniques for neural networks during past 30 years, but no a common pruning methods has appear, currently the deep learning and deep neural network developing quickly, the structure and topology of these network is becoming more complex, we could say that the pruning problem have become a more model or network specific question, the pruning techniques for deep network have shown the good research potential, there are some works already tried to pruning Constitutional Networks [9][10], in this work we also applied a simply pruning method on CNN, but the pruning techniques for recurrent neural network and LSTM is still a topic can be discussed more, we could focus on this gap. Besides, the pruning methods' efficiency is also a difficult research question, particularly for those big networks work on huge data set.

References

[1] T.D. Gedeon & D. Harris, Network Reduction Techniques

[2] 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).

[3] See, A., Luong, M. T., & Manning, C. D. (2016). Compression of neural machine translation models via pruning. *arXiv preprint arXiv:1606.09274*.

[4] Rauschecker, J. P. (1984). Neuronal mechanisms of developmental plasticity in the cat's visual system. *Human neurobiology*, *3*(2), 109-114.

[5] Walsh, C. A. (2013). Peter Huttenlocher (1931–2013).

[6] L. Breiman, J.H. Friedman, A. Olshen, and C.J. Stone. (1984). Classification and Regression Trees. Chapman & Hall, New York, 1993. Previously published by Wadsworth & Brooks/Cole.

[7] Rodrıguez, J. J., Alonso, C. J., & Boström, H. (2000). Learning first order logic time series classifiers. In *Proceedings of the 10th International Workshop on Inductive Logic Programming*(pp. 260-275). Springer.

[8] Kohavi, R. (1996, August). Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In *KDD* (Vol. 96, pp. 202-207).

[9] Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference.

[10] Anwar, S., Hwang, K., & Sung, W. (2017). Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, *13*(3), 32.

[11] LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems* (pp. 598-605).

[12] Yan, J., & El Ahmad, A. S. (2008, October). A Low-cost Attack on a Microsoft CAPTCHA. In Proceedings of the 15th ACM conference on Computer and communications security (pp. 543-554). ACM.

[13] Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning structured sparsity in deep neural networks. In Advances in Neural Information Processing Systems (pp. 2074-2082).