Pruning Methods based on Hidden Neuron Functionality

Idris Mustafa
Australian National University
Acton, ACT, Australia
u6733671@anu.edu.au

Abstract: Pruning has always been an important part of fine tuning neural networks. The complex architecture of the neural network and various parameters used may cause problems in generalisation performance. Pruning process reduces the size of the neural network, making it more efficient. In this paper we compare the similarity measures, introduced in Weight Matrix versus Neuron Behaviour (T.D. Gedeon), used for the pruning process in a deep learning model trained through transfer learning process. The paper focuses on the results of pruning performed based on two different similarity measures, first using the similarity measure computed from neuron activation output and second, using the similarity measure computed by the weights on the output. The pruning process using neuron activation output results in better generalisation performance than the pruning process using the static weight matrix.

Keywords: Pruning, Neuron Behaviour, Network Reduction

1. Introduction

Neural networks are a popular tool used to solve problems ranging from simple classification, regression to natural language processing and computer vision tasks. Massive amount of data gathered in the last decade has contributed to the increasing popularity of deep neural networks used in numerous fields of study. As the application domain is constantly growing, the architectures of various neural network models have become more complex. Therefore, problems such as generalisation, computation costs, storage and efficiency require more attention.

Pruning is a popular method used to compress the pre trained network, which helps reduce the computation cost and increase generalisation performance. J. Sietsma and RJF.Dow have described how generalisation is improved in a multilayered neural network by taking only a few neurons in the first layer[4]. Rules extraction from pruned networks for breast cancer diagnosis has been described in detail by R. Setiono[6]. In this paper we use pruning of hidden neurons based on the *distinctiveness* property introduced by Gedeon and Harris[2] and compare the results with pruning based on the *distinctiveness* property defined by the weight matrix[1]. Another successful research describes the *sensitivity* property that can be used for the pruning process[5].

In a previous experiment, the pruning process was implemented on a two layer feed forward neural network and the weights were updated using the error back propagation algorithm. The network was trained on the combination of PCA components of Local Phase Quantisation (LPQ) features and the PCA components of Pyramid of Histogram of Gradients (PHOG) features extracted from the SFEW dataset[3]. The model was trained for 1000 epochs with 15 hidden neurons and sigmoid activation on the hidden neurons.The hidden neurons were connected linearly to 7 output neurons which predicted 7 different facial expression classes. The accuracy obtained on the test set was 22%. We extend our research by applying the pruning process on a deep learning model.

In this experiment we use the Static Facial Expression dataset[3]. The images are extracted from the AFEW dataset[11] which contains video clips extracted from movies. The images in the dataset covers facial expressions with varied head poses, large age range, close to real world illumination and different face resolutions. The dataset contains 675 images labelled for 7 expressions *angry, disgust, fear, happy, sad, neutral* and *surprise* and part of our goal is to build a classification model that can classify these 7 expressions.

The images in the dataset contain faces at random positions and random orientations. To effectively classify facial expressions into 7 different emotions, a face detector is run to obtain a box region in which the face is detected. The newly cropped face images are used for the classification purpose. Figure 1. shows the improvement in prediction accuracy after processing the images using a face detector.

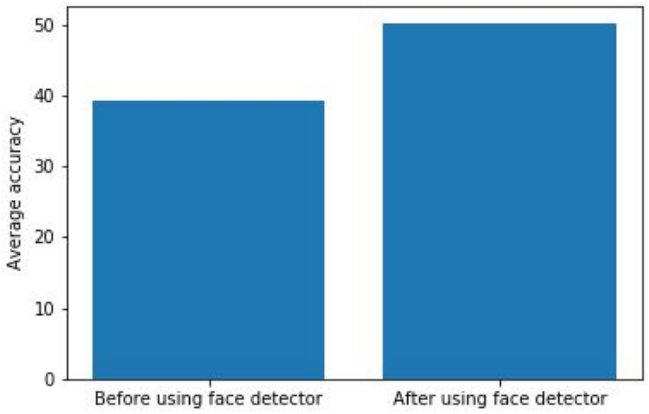

Figure 1. VIsualisation of average accuracy before and after implementing a face detector.

2. Methodology

Convolutional neural networks (CNN) are a class of deep neural networks most commonly applied to computer vision tasks. In Image classification, CNNs are used to reduce the image into a form which is easier to process and does not lose any features which impact the predictions. In our experiment we attempt to build our own classifier model from scratch. The model has 3 layers of convolution with max pooling and relu activation on each layer. The final layer of convolution is linearly connected to a hidden layer of 100 neurons activated using the relu activation function, the output from which is connected to 7 neurons which predict the facial expressions. The model was trained for 70 epochs with a learning rate of 0.001. The model predicts with 37.47% accuracy on the test set.

Another approach used to classify with better accuracy is using pre trained models. The approach of transfer learning not only increases the accuracy from our model but also learns in a lesser number of epochs. Transfer learning is a machine learning method where a model built for a task is reused as the starting point for another task. The models are usually trained on massive amounts of data and then transferred to smaller dataset with some fine tuning. As the number of images in the SFEW dataset[3] is limited to only 675 images, transfer learning approach results in better performance than our model. For fine tuning the pretrained model according to the dataset, we freeze all the layers except the final layer of the pre trained model. The final layer is connected sequentially to 2 layers of hidden neurons and these layers are connected to 7 output neurons. In the training process, only the newly connected layers are trained.

We attempt to pick the best pre trained model for our experiment from a set of models made available in *pytorch models* subpackage. *Resnet18*[7], *VGG16*[9] and *Alexnet*[8] are three CNN models used in this experiment. In terms of training time and prediction accuracy, *Alexnet* takes comparatively less time to train and has better generalisation performance compared to other models. *VGG16*[9] has 13 layers of convolution before the final layer which impacts heavily on the time it takes to train. *Resnet18*[7] is 18 layers deep which takes time to train and also results in poor generalisation performance. It needs more images to perform better as it was trained on the *ImageNet*[10] database with a million images. *Alexnet*[8] architecture has 5 layers of convolution followed by 3 fully connected layers which implies that there are less number of computations during the training process. Hence, we selected *Alexnet* for the next part of our experiment.

We try to investigate the pruning process on the pretrained model picked for our classification problem. The pruning technique is from Gedeon and Harris' Network Reduction Techniques research[2] in which they describe a similarity measure to prune the neurons that are too similar or complementary. The pruning process mentioned in the paper[2] uses the activation output of hidden neurons to calculate the similarity. The paper[2] describes that the *distinctiveness* property can be determined from the neuron activation output vector which represents the functionality of the hidden neurons. The similarity of pairs of vectors is computed by finding the angle between them. The vectors are normalized to [-0.5,0.5] to use the angular range of 0-180^{o}. The pairs with angles less than 15^{o} are considered to have similar functionality and one of the hidden units is removed. The weight vector from the hidden unit which is removed is added to the other hidden neuron. The pairs of vectors that have angular separation greater than 165^{o} are considered to have complementary functionality and both the hidden units are removed.

Pairs		Angles	Similarity
0	1	24.331404861484447	Similar
0	2	132.25963282396518	
0	3	122.56606442934299	
0	4	142.69377130953515	
0	5	131.88621272166222	
0	6	34.664166264298515	
0	7	145.00200641878428	
0	8	27.853400444737986	Similar
0	9	151.02292745517724	Complementary
0	10	146.94683448305372	
0	11	145.25858930514497	
0	12	77.83848868724445	
0	13	141.41475008555855	

Table 1. The angle separation calculated from activation output of the hidden neurons and angular separation threshold set at 30 to determine similar and complementary pairs.

In another research Gedeon[1] describes that the static trained weight matrix can also be used as an indicator to hidden neuron functionality. He uses the weights on the outputs to compute similarity of hidden units. He compares the pruning results using the similarity measure computed from the static weight matrix with the pruning results using the similarity measure computed from the neuron activation output. In our experiment, we compare the results of pruning on our pre trained model using the two different similarity measures.

2.1 Pruning process

The final convolutional layer of *Alexnet*[8] results in 9216 features which are linearly connected to 2 layers of hidden neurons. The first layer contains 1024 neurons. The second layer of hidden neurons are connected to 7 output neurons. Our first pruning process is done by extracting neuron activation output of the second layer. We vary the number of hidden neurons in the second layer from a range of 100 neurons to 1000 neurons, to investigate better the outcome of pruning. The dimension of the hidden neuron activation output vector is (number of input neurons x number of hidden neurons). In our experiment, we have used 70% of the data for training and the remaining data for testing purposes. If we take 100 hidden neurons in the second layer, we obtain a vector of dimensions (472 x100) where 100 columns represent the activation output vectors related to 100 hidden neurons and 472 rows represent the number of training inputs. The output activation vector is normalized to the range -0.5,0.5 and angular separation is computed between pairs of column vectors. We use the angular separation threshold different from the one described by Gedeon and Harris[2]. The pairs of column vectors that have an angular separation less than 30^o are considered similar. Hence, we remove one of the hidden neurons and transfer the weights of the hidden neuron which is removed to the other hidden neuron which remains. The pairs of column vectors that have an angular separation greater than 150^o are considered complementary, hence both the neurons are removed.

Angular Separation Threshold	Hidden Neurons	Accuracy	Neurons Pruned	Accuracy After Pruning
15	100	48.27	4	49.26
	500	52.21	4	52.7
	1000	52.21	21	52.21
20	100	48.27	16	48.76
	500	52.21	63	52.7
	1000	52.21	36	52.21
30	100	48.27	50	48.76
	500	52.21	307	52.21
	1000	52.21	630	53.69

Table 2. For angular separation threshold of 30, accuracy on the test after pruning more than half the number of hidden neurons remains close to the accuracy obtained with no neurons pruned.

The angular separation threshold is set to 30^o for the experiment as for lower thresholds, similar and complementary pairs could not be determined effectively. Table 2. shows the pruning performance with different angular separation thresholds.

A similar pruning is done by taking the static weight vectors to compute similarity. Gedeon[1] describes that the weights from hidden neurons to the output neurons are considered as similar to the use of hidden neuron activation output. Hence, in our experiment, we extract the weight vectors of the final fully connected layer that connects to 7 output neurons. For 100 neurons in the second hidden layer, we obtain a weight matrix of dimension (100 x 7) where 100 rows represent the weight vectors related to 100 neurons. The pruning procedure is repeated as described earlier in section 2.1.

3. Results

3.1 Results of pruning on feed forward neural network

In our previous experiment of pruning on a two layer feed forward neural network, the model was trained for 1000 epochs at a learning rate of 0.01. Table 3. shows the results of pruning using the neuron activation output to compute similarity. Table 4. shows the results of pruning using static weights to compute similarity.

Hidden Neurons	Test accuracy (no pruning)	Neurons pruned	Test accuracy
75	26.11%	21	21.67
100	26.11%	44	18.23
125	24.14%	49	16.26
150	23.65%	68	19.7

Table 3. Pruning results on models with different numbers of hidden neurons using the neuron activation output to compute similarity.

Hidden Neurons	Test accuracy (no pruning)	Neurons pruned	Test accuracy
75	26.11%	24	20.20
100	26.11%	51	15.76
125	24.14%	66	18.23
150	23.65%	89	13.79

Table 4. Pruning results on models with different numbers of hidden neurons using the static weights on output to compute similarity.

Pruning results in Table 3. which uses neuron activation output as distinctiveness of hidden neurons show better generalization performance compared to the pruning results in Table 4. which uses static weights as distinctiveness of hidden neurons. Pruning process using the weight matrix provides a better compression ratio but results in poor prediction accuracy on the test set.

3.2 Results of pruning on the deep learning model

As transfer learning is implemented, the model is likely to overfit on our dataset. We attempt to reduce overfitting by training for a lesser number of epochs and low learning rate. Hence, we train our model for 35 epochs at a learning rate of 0.001 which results in consistent generalisation performance every time we train the final 3 layers of our model.

We compare the results of pruning with 100 to 1000 hidden neurons in the penultimate layer of our model. Table 3. shows the prediction accuracy on the test set and the number of hidden neurons pruned in our pruning process that uses hidden neuron activation output to compute similarity. Table 4. shows the results of the pruning process that uses static weights to compute similarity.

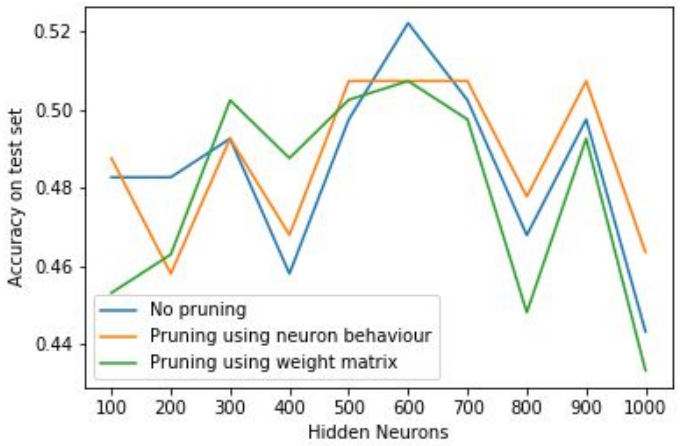

Figure 2. Accuracy on test set obtained from pruning using 2 different similarity measures with different number of hidden neurons in the penultimate layer.

The model generalises better when pruning is done by neuron behaviour distinctiveness. From the results in Table 5, the majority of the cases show an improved prediction accuracy on the test set after pruning away more than half the number of hidden neurons.

Neurons	Accuracy Before Pruning	Neurons Pruned Ratio	Accuracy After Pruning
100	48.27	44%	48.76
200	48.27	61.5%	45.81
300	49.26	61.3%	49.26
400	45.81	63%	46.8
500	49.75	65.4%	50.73
600	52.21	67%	50.73
700	50.24	68.1%	50.73
800	46.79	65.7%	47.78
900	49.75	68%	50.73
1000	44.33	63.9%	46.36

Table 5. Prediction accuracy on test set before and after the pruning process that uses neuron activation output to compute similarity.

Pruning using static weights as neuron distinctiveness results in higher compression ratio than the pruning using the neuron behaviour. 81.5% pruning ratio is achieved from pruning of a model with 1000 hidden neurons. In most of the cases in Table 7, the generalisation performance reduces after pruning.

Apart from the experiment of pruning, we also attempt to improve the prediction accuracy of our classification model. We obtained a prediction accuracy of 53.69% by keeping 512 hidden neurons in the penultimate layer and training the model for 65 epochs at a learning rate of 0.0001.

Emotions	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Precision	0.65	0.29	0.56	0.56	0.35	0.52	0.52
Recall	0.50	0.40	0.41	0.66	0.48	0.48	0.48
F1-score	0.57	0.33	0.47	0.60	0.41	0.50	0.50

Table 6. Precision, Recall, F1-score of model trained with 512 hidden neurons for 65 epochs at a learning rate of 0.0001.

Neurons	Accuracy	Neurons Pruned ratio	Accuracy after pruning
100	48.27	48%	45.32
200	48.27	65%	46.3
300	49.26	65.6%	50.24
400	45.81	69%	48.76
500	49.75	71.2%	50.25
600	52.21	72.8%	50.73
700	50.24	77.3%	49.75
800	46.79	78%	44.82
900	49.75	78.2%	49.26
1000	44.33	81.5%	43.34

Table 7. Prediction accuracy on the test set before and after the pruning process that uses the static weight matrix to compute similarity.

4. Discussion and Conclusions

We have shown the comparison of pruning processes using the two similarity measures as described by Gedeon[1]. The results from pruning of the two layer feed forward network are similar to the results obtained in the pruning process of the deep learning model. The generalisation performance of pruned models is better when neuron activation output is used to determine the distinctiveness of neurons. Better compression ratio is achieved in the pruning process that uses static weights matrix to determine the distinctiveness of neurons. Models with better compression ratio results in poor generalisation performance which implies that some of the necessary neurons are also getting pruned. It also implies that the static weights are not reliable information to determine the functionality of neurons. The neuron activation output resembles closely the behaviour of the hidden neurons.

5. Future Work

Our experiment was focused on the results of pruning from two similarity measures, first using neuron activation output to compute similarity of neurons and second using static weight on the output to compute similarity. The experiment was limited to 675 images for training and testing purposes. Further research can be done to test the pruning methods on models trained on other datasets. The pretrained model, *Alexnet*[8], used in our experiment was trained on the ImageNet database[10]. Research can be extended to investigate if there is a significant improvement in generalisation performance after pruning of various pre trained CNN architectures that achieved benchmark performances on the ImageNet database.

6. Reference

[1] T. D. Gedeon, "Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour," Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, Dunedin, New Zealand, 1995, pp. 26-29.

[2] Gedeon, T. D., and D. Harris. "Network reduction techniques." *Proceedings International Conference on Neural Networks Methodologies and Applications*. Vol. 1. 1991.

[3] Dhall, Abhinav, et al. "Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark." *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 2011.

[4] Sietsma, Jocelyn, and Robert JF Dow. "Creating artificial neural networks that generalize." *Neural networks* 4.1 (1991): 67-79.

[5] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," in *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 239-242, June 1990.

[6] Setiono, Rudy. "Extracting rules from pruned neural networks for breast cancer diagnosis." *Artificial intelligence in medicine* 8.1 (1996): 37-51.

[7] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[8] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

[9] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[10] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009.

[11] *Abhinav Dhall, Roland Goecke, Simon Lucey, Tom Gedeon*, Collecting Large, Richly Annotated Facial-Expression Databases from Movies, IEEE Multimedia 2012