Network Reduction Based on Distinctiveness for Classifying Faces-Emotion

Samyak Jain
Research School of Computer Science & Engineering
Australian National University
Acton, ACT , Australia

Abstract. In this age of deep learning , neural nets with large number neurons and parameters are very common . But , a rule of thumb for obtaining good generalisation is to make sure that we build the smallest system that fits the data. It is essential to prune the network for making it more efficient. In this paper we discuss about one such pruning technique that we learned from a previous paper that is based on the angles between the unit vectors of the hidden layer which are similar or complementary towards the output. Initially we build a simple three layer network to classify different emotions of faces and then compare the results after applying the technique and see how it affects our neural network.

Keywords: Network Pruning , Neural Net , Faces-Emotion,Reducing Network Size.

1 Introduction

With the rapid advancement of machine learning , much of the progress in the last decade has been a result of deep learning. Deep Learning models these days require a significant amount of computing, memory, and power which becomes a bottleneck in the conditions where we need real-time inference or to run models on edge devices and browsers with limited computational resources. It is therefore the need of the hour to make these networks very efficient. Pruning neural networks helps to obtain good generalisation as it aims to remove undesired parameters in the existing network. The goal of pruning is to reduce the size of the initially made neural network without affecting the accuracy of the network. The results obtained from reducing a large network to a smaller one is better than making a smaller network from the start. Gedeon And Harris[1] provided us with a pruning technique that is based on reducing the number of hidden neurons based on the angle between the pattern vector of the activated output of the hidden layer.

The Static Facial Expression Analysis dataset[2] provides us with features such as Local Phase Quantisation(LPQ) and Pyramid of Histogram Of Oriented Gradients(PHOG) as our input features. Using these features we try to classify emotions of static faces by our 3 layered neural network. Our task is to build a simple neural net first and then obtain the accuracy of the network . Then, we try to implement the technique of network pruning and compare the accuracy of our pruned network to that of our original network. In this paper, we will see how the pruning has effected our model based on the number of neurons pruned and observe how this is impacting the accuracy of the model.

2 Method

Since our dataset consists of 5 (LPQ) features and 5 (PHOG) features , we first tested our network model on these features individually . We tried testing our model with hidden neurons ranging from 5 to 25 . We find that the average accuracies of these individual features were not higher than the accuracy we obtained from combining these features and using all 10 features as inputs. So we decided to proceed with our findings and experiments using all 10 features as inputs together.

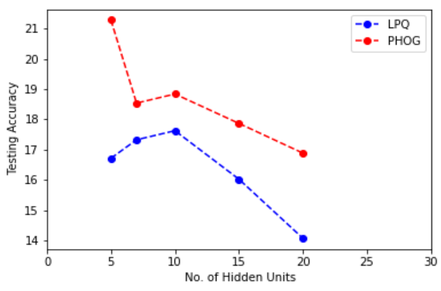

Fig.1 Testing Accuracy of LPQ and PHOG

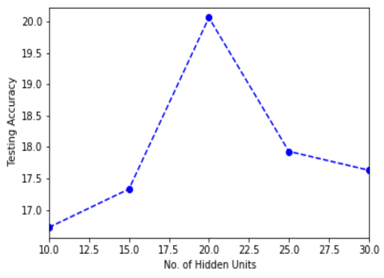

Fig.2 Testing Accuracy of LPQ and PHOG combined

Firstly we will start off with pre-processing our SFEW dataset which would include normalising the values , filling any missing values that might be present in the dataset. Next step would involve separating emotion labels from the rest of the dataset since that what we are trying to classify. Then, construction of our neural network which would involve initialising the weight and shape of our input , hidden and output layers . After running our model, next step would be to find angle between the hidden activation output . After finding the angles , getting the indices of the pair of similar and complementary neurons and storing them.

After this , we modify our network by removing the complementary neurons and adding weights of the similar neurons and then removing one of them and then compare the results of our original model with the new reduced model.

2.1 Pre -Processing

To begin with we see that our data has some missing values , so we filled those missing values with the means of their column. We did not set the missing values to zero, as zero might have some impact later when we begin to build our model. The class labels were ranging from 1 to 7 , but since neural network expects first class to be zero so we represented our labels ranging from 0 to 6. Then we split our data into training and testing sets with 346 rows out of 675 for training and the rest for testing. We scaled our training inputs with a min-max scaler so as to normalise our data from 0-1.

2.2 Simple Neural Net model

We start off by building a simple 3 layer network with input neurons of the size of our input dimension and hidden neurons set to 20 and outputs neurons to be the size of our output classes . According to paper[5],a general rule of thumb when defining the size of hidden neurons are given as follows:

- The number of hidden neurons should be 2/3 size of the input layer , plus the output layer.
- The number of hidden neurons should be less than twice the size of input layer.

Besides this RELU activation function on the hidden layer and Adam optimiser are used. For calculating loss, the Cross Entropy loss is used.[6]

2.3 Finding angle between hidden neurons

From our model, we make use of the hidden layer output which is the relu activation applied on the hidden layer .This hidden layer output is of the size (input_neurons,hidden neurons) where every column is our hidden pattern vector and we aim to find angle between all pairs of columns. According to Geoden and Harris[1], first of all these values should be normalised between -0.5 to 0.5 so that the range of angle is 0-180 degrees instead of 0-90.Then , we find the angles between all pairs of columns by using cosine similarity. The paper[1] suggested that vectors between which the angle of separation is below 15 degrees are considered sufficiently similar and one of them is removed . The weight vector of the the unit which is removed is added to the weight vector of the unit which remains. Similarly, vectors between which angle of separation is greater than 165 degrees are considered to be complementary and both should be removed as they are in opposite direction of the output.

H1 H2 H3 H4 H5 H6

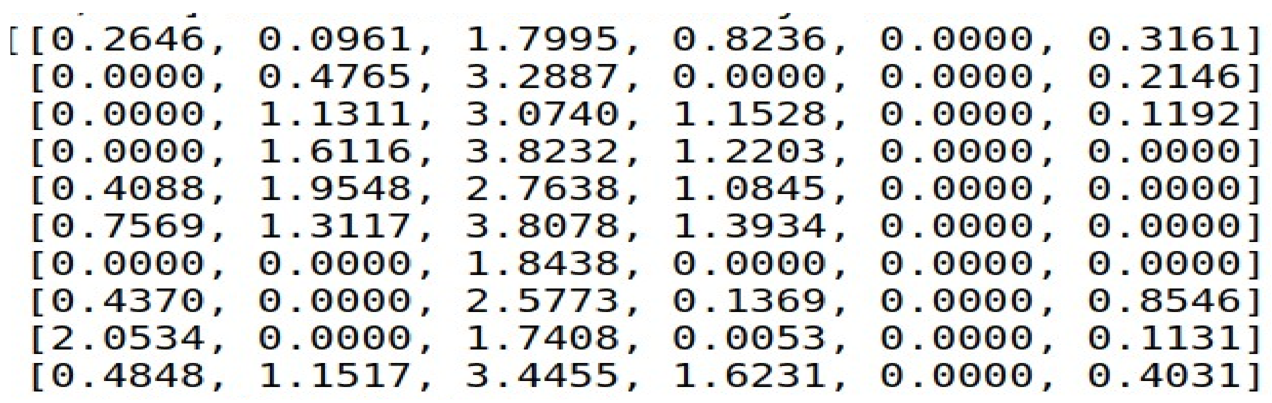
```
[[0.2646, 0.0961, 1.7995, 0.8236, 0.0000, 0.3161]
 [0.0000, 0.4765, 3.2887, 0.0000, 0.0000, 0.2146]
 [0.0000, 1.1311, 3.0740, 1.1528, 0.0000, 0.1192]
 [0.0000, 1.6116, 3.8232, 1.2203, 0.0000, 0.0000]
 [0.4088, 1.9548, 2.7638, 1.0845, 0.0000, 0.0000]
 [0.7569, 1.3117, 3.8078, 1.3934, 0.0000, 0.0000]
 [0.0000, 0.0000, 1.8438, 0.0000, 0.0000, 0.0000]
 [0.4370, 0.0000, 2.5773, 0.1369, 0.0000, 0.8546]
 [2.0534, 0.0000, 1.7408, 0.0053, 0.0000, 0.1131]
 [0.4848, 1.1517, 3.4455, 1.6231, 0.0000, 0.4031]
```

Fig.3 First 6 Pattern vectors before normalising

Table 1. Pair of unit vectors and the corresponding angle between these vectors. Angles below 15 are similar and angles above 165 are considered complementary.

Pair of Units		Degree of Vectors(Angle)
6	7	61.29
6	8	155.12
6	9	50.96
6	10	27.72
6	11	30.18
6	12	27.18
6	13	27.18
6	14	29.35
6	15	43.43
6	16	27.18
6	17	27.18
6	18	30.26
6	19	33.49

2.4 Pruning our model

As we have calculated the angles between the unit vectors we now know which neurons we have to prune. Our first step would be to remove all pairs of neurons that are complementary (greater than 165) to each other . This will make our adding of weight vector calculation for similar neurons easier as complementary nodes would be removed first. To do this , first we set the corresponding row of the weight vector of both complementary neurons to zero. Second we take the corresponding column in the output weight vector and set it to zero. And , at last we set the corresponding row of bias vector of hidden unit to zero.

Now since we have deleted the neuron pairs which are complementary , it will ease the computation and addition of the similar neuron vectors. To do this, we first made a clone of all the weight and bias vectors of our original model. After finding out the indices of the similar pair of neurons , the weight vector of neuron (at first index) is added to the weight vector of another neuron(at second index), and then the weight vector of former neuron is set to zero. This steps would ensure that the size of the network has reduced from the original one.

The last thing to do is to edit the weights and bias of the original model so that now our model works with the new reduced sets of weights and bias. For this we created another model and initialised all the layers of this new model with the layers of our old model. Then we edited the weights and bias of our new model with the new sets of features we got from pruning. Pruning all layers uniformly tends to perform worse than intelligently allocating parameters to different layers[3 , 4]. This would make sure that our model would not need to be retrained and we can directly test our new model on the training set.

3 RESULTS AND DISCUSSION

The classification report that provides average precision and recall for each class (emotion labels) is the most important evaluation measure in this report. Before pruning the training accuracy we achieved was 47.11% and the test accuracy achieved was 17.63 %. Then after performing pruning and creating a new model with our new reduced network we find that our accuracy on the test set decreases to 14.29 %. We ran both the models five times and saw that when the neurons pruned are more than 8 (out of 20 that we stated initially with) our reduced network performs better than the original model. But when less neurons are pruned (6 or 8) the original model performs better on the test set. It is evident that 20 hidden neurons are very large for our network, since half the number of neurons are getting pruned almost very time. So we can observe that the smaller the network prunes to , the reduced network performs better.

Table 2. Average expression class wise Precision, Recall and f1-score results before model pruning.

BEFORE PRUNING							
Emotion	**Angry**	**Disgust**	**Fear**	**Happy**	**Neutral**	**Sad**	**Surprise**
Precision	0.10	0.07	0.20	0.30	0.27	0.29	0.32
Recall	0.20	0.13	0.37	0.36	0.18	0.22	0.19
f1-Score	0.13	0.10	0.26	0.33	0.21	0.25	0.24

Table 3. Average expression class wise Precision, Recall and f1-score results after model pruning.

AFTER PRUNING							
Emotion	**Angry**	**Disgust**	**Fear**	**Happy**	**Neutral**	**Sad**	**Surprise**
Precision	0.23	0.21	0.45	0.02	0.18	0.04	0.12
Recall	0.18	0.19	0.17	0.25	0.13	0.18	0.19
f1-Score	0.20	0.20	0.24	0.04	0.16	0.07	0.15

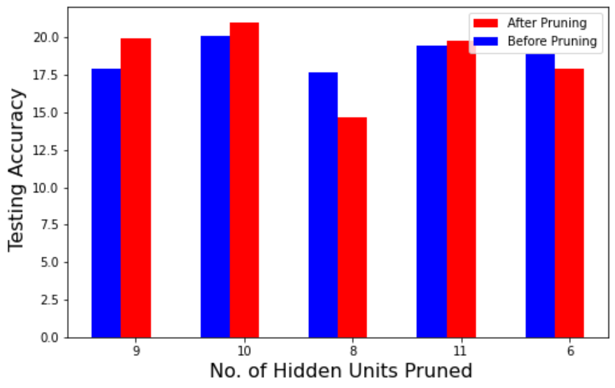

Fig 4. Accuracy before and after pruning with different number of neurons pruned

From Fig.4 we can observe that when large number neurons are getting pruned (greater than 40%) than the model after pruning performs better than the original one, but on the other hand when small number of neurons are getting pruned (less than 40%) the original model performs better.

4 FUTURE WORK AND CONCLUSION

This report provides a basic neural network classification problem with only one hidden layer. We see that almost every time the pruning takes place, it reduces the network size from 40% to 50%. Since our model has only one hidden layer different pruned models does not have significant accuracy or evaluation measure difference between original and reduced model. We can try to build the model with large number of hidden layers and the compare the results which could give us a better idea as to how the pruning technique is affecting our model since neurons in each hidden layer would then be pruned . Another thing we could do is try and test our model with different training and testing set sizes. For our model we were restricted to have 346 data samples as training and rest for training. Giving more data for training and then comparing results might give us better classification accuracy.

This report confirms that 20 hidden neurons that we initially started with is more than what the neural network needs to perform good generalisation. This could be seen by the fact that every time the number of neurons pruned are almost 50%. The more number of neurons pruned the better our model is performing on the test set. More research as to how our dataset performs with different number of hidden neurons and the ratio of hidden neurons to the number of neurons getting pruned could be done.

5 REFERENCES

[1] Gedeon, TD, Harris, D, "Network Reduction Techniques," Proc. Int. Conf. on Neural Networks Methodologies and Applications, AMSE, vol. 1, pp. 119-126, San Diego, 1991a. Proc. 9th Ann. Cog. Sci. Soc. Conf. , Seattle, pp. X-y, 1987.
[2] Dhall, A., Goecke, R., Lucey, S., & Gedeon, T. (2011, November). Static facial expressions in tough conditions: Data, evaluation protocol and benchmark. In 1st IEEE International Workshop on Benchmarking Facial Image Analysis Technologies BeFIT, ICCV2011.
[3] Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks, 2019.

[4]Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In Advances in neural information processing systems, pp. 1135–1143, 2015.
[5] Sheela, K.G. & Deepa, S.N.,: Review on Methods to Fix Number of Hidden Neurons in Neural Networks, Mathematical Problems in Engineering, vol. 2013, pp. 1-11. 2013
[6] Z. Zhang, "Improved Adam Optimizer for Deep Neural Networks," *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, Banff, AB, Canada, 2018, pp. 1-2.