Comparing the Efficiency of Pruning Neurons of a Hidden Layer in a Feed-Forward Neural Network between Fuzzy C-means Clustering Based and Distinctiveness Based Pruning techniques

Sree Venkatesh Yelamolu, u7095134@anu.edu.au

Research School of Computer Science, Australian National University

Abstract. In this paper, we introduce a novel method of Fuzzy C-means Clustering-based pruning of neurons in a hidden layer and compare its performance with Distinctiveness based pruning on the task of fine-grain classification of classifying the extracted image feature vectors by Resnet pre-trained on ImageNet dataset of vehicle-x images dataset into 1,362 classes using a one hidden layer feed-forward neural network. We show that using Fuzzy C-means Clustering-based pruning of neurons outperforms the pruning based on distinctiveness in both achieving better test accuracy of 11.8 percent with just 194 hidden neurons for the hidden layer which is slightly higher test accuracy when compared to 11.5 percent when using 4096 neurons for the hidden layer and also with less time complexity for executing.

Keywords: Feed-forward Neural Network· Cosine Similarity · Pruning · Neuron Activations · Fuzzy C-means Clustering · Fuzziness Value.

1 Introduction

A Feed-forward neural network can be used for both regression(Predicting a target with continuous value) [1] and classification(Predicting a target with discrete values) [2] tasks by just changing the output layer activation function and the function for loss minimization. For any multi-class(More than 2 classes) classification task even with images [3] and text classification [4] finally, after the feature vectors are extracted from the inputs like images or text the feature vectors need to be passed through a Feed-forward fully connected network with the output layer neurons activation using Softmax function to get the class probabilities for the feature vectors. Though, MSE loss function can be used for the optimization in regression Cross-Entropy loss gives better results for the classification tasks as the latter considers how probable the feature vector belongs to a particular class and thus tries to maximize it.

1.1 Importance Of Pruning

The number of neurons in the input layer is just the number of features of the training examples and the output features are just the number of class labels we want to predict. But the number of hidden layers and the number of hidden units per layer to be used within this feed-forward neural network is just a hyperparameter that is chosen by running multiple tries on a different number of hidden units and hidden layers and the model with the best validation accuracy is chosen but there is no theoretical proof that the chosen model is the most optimized in terms of the number of parameters. As a result, the models have millions of parameters and many of them might be redundant and might not contribute anything or a little to the target prediction and these models cannot be deployed in computationally limited hardware devices like mobile phones, etc. which results in using cloud computation where people are skeptical sharing their private information to the cloud. So, choosing the number of hidden units based on some theoretical proof [5] makes the model make the most out of the parameters by also reducing the number of parameters that aren't contributing much to predict the target. According to a survey conducted by neural magic [6], 58 percent of the people working as data scientists didn't optimize their model specifically for deploying in production. When the people in the data science team size is low the pruning done is also low when compared to teams with more people.

1.2 Dataset

The dataset used in this paper consists of 2048 dimensional feature vectors of VehicleX [7] synthetic dataset images which were extracted by Resnet architecture which is pre-trained on the ImageNet dataset. The vehicle-x images dataset consists of 1,362 vehicles of various 3D models with fully editable attributes like the viewpoint of the vehicle,

illumination on the vehicle, etc. The dataset with the image feature vectors is split randomly into 3 parts training set consisting of 45,438 examples, a validation set consisting of 14,936 examples, and a testing set with 15,142 examples. We use the t-distributed stochastic neighbor embedding technique [8] to find any clusters within our 2048 feature vectors where all the training examples act as their dimensions and as shown in figure 1. We can see that there's only 1 huge cluster for all the 2048 feature vectors. So, we cannot prune any features within the training data.

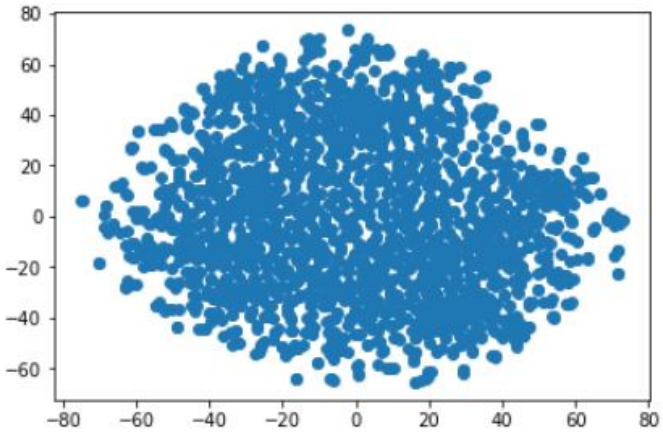

Fig. 1. Training data plot in 2-D

2 Methadology and Model Design

2.1 Preprocessing Of The Dataset

All the example feature vectors of the training set are normalized within each dimension by subtracting the mean of the dimension and by dividing with the standard deviation of each dimension. The training set mean is subtracted from the examples in the validation and testing sets and was divided by the training set's standard deviation.

2.2 Network Design

A Feed-forward neural network as shown in figure 2 is designed initially with a hidden layer. RELU activation function is used for the hidden layer and the Softmax activation function for the output layer. Cross-entropy loss is used for optimizing the parameters(weights and bias) of the network through a Stochastic gradient descent algorithm.

2.3 Evaluation Metric

The evaluation metric used for testing the model is Accuracy. It is defined as the number of examples the model has correctly classified divided by the total number of examples. We used this metric because the test set has approximately an equal number of examples for all the classes.

2.4 Training and Testing the Model

The model is trained on the training set with a batch size of 100 for 100 epochs before the model starts to overfit with the training examples i.e before the validation loss starts to increase and the hyperparameter '4096' neurons in the hidden layer is chosen with respect to the better validation accuracy after testing it on the validation set. Finally, after finalizing the hyperparameters the model is tested on the test set and test accuracy is noted.

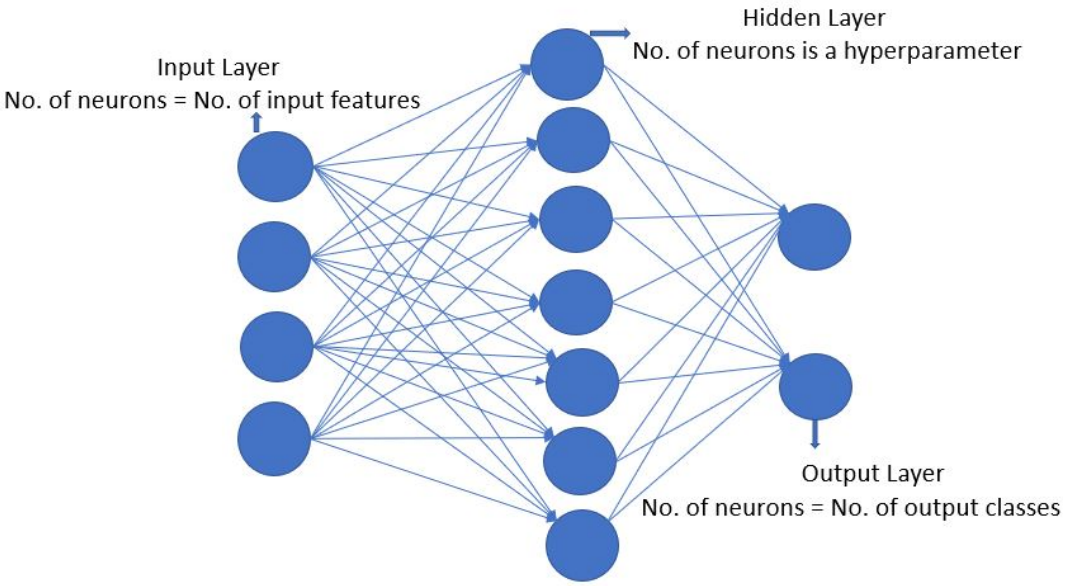

Fig. 2. Feed-forward Neural Network Architecture

2.5 Pruning the input vector features and hidden units of the hidden layer based on Distinctiveness

After starting with the baseline of 2048 input features and 4096 neurons for the hidden layer a series of experiments were conducted to reduce the number of neurons in the hidden layer. Initially, all the 4096 neurons were used for training the model till the validation loss decreases and converges to a specific value. Later the neurons with activation close to 0 for all the training examples are removed [9]. Next, the similarity between each neuron is compared to the other neurons by using the normalized vector of neuron activation values for all the training examples and if the angle between them is less than some threshold that means they are similar one of them is removed from the model and the parameters of the removed neuron are added to the neuron which is similar to it [9]. If the similarity between neurons is is such that they cancel each other(i.e if the angle between them is close to 180 degrees) both of them are removed as they can be viewed as complementary to each other [9]. Later, the model is retrained with the remaining neurons for almost 2000 epochs with the size of the batch size equal to the number of training examples using a faster convergence ADADELTA[10] algorithm which is an adaptive learning rate method for gradient descent where the validation loss stops to decrease. We test the retrained model on the entire test data and note its test accuracy. In this paper, we also check the angles between the features of the input feature vectors and prune some features according to the above criteria and compare the accuracy.

2.6 Pruning the neurons based on Fuzzy-Clustering

We aren't able to prune any features of the training data for this dataset as all the features belong to a single cluster. But we can prune neurons in the hidden layer. Same as in the above section we train our network with 4096 neurons in the hidden layer till the validation loss decreases and converges to a specific value. Later the neurons with activation close to '0' for all the training examples are removed. Next, we apply Fuzzy C-means clustering [11] on the remaining neuron activations by providing half of the number of neurons of our hidden layer as the number of clusters and then once the clustering is done, we only keep a single neuron with the highest fuzzy value for that cluster and prune all other neurons within that cluster. We add the weights of the pruned neurons multiplied by their respective fuzzy values within that cluster to the neuron we are keeping. In this way, we are not wasting the other neuron's weights which are trained but only considering their weights according to how much fuzziness value they have for that cluster. Later the neurons which are kept are retrained for almost 1000 epochs with the size of the batch size equal to the number of training examples using the ADADELTA algorithm as discussed in the previous section where validation loss stops to decrease. We test the retrained model on the entire test data and note its test accuracy.

3 Results and Discussion

A validation and test accuracies of 11.7 percent and 12.5 percent respectively without a hidden layer in the model are obtained which is a little better when compared to adding a hidden layer. But as we wanted to compare the accuracies of different models concerning pruning the neurons we used a hidden layer in our model.

3.1 Base model Accuracy before Pruning and After Pruning the Input Features

As each input is a 2048 dimensional feature vector first the model with a hidden layer of 4096 neurons is trained without pruning any features of the input and later the model is trained after pruning the features according to the criteria mentioned in the section 2.5 and the test accuracies for both the models are compared in the table 1 below. So, we can see that pruning few features of the input data according to how distinct they are can give almost the same accuracy and a little better accuracy compared to the model using all the input features.

Table 1. Table showing the accuracy vs number of input features pruned.

Minimum Angle Threshold (Similarity)	Maximum Angle Threshold (Complimentary)	Number of Features before Pruning	Accuracy before Pruning	Number of Features After Pruning	Accuracy after Pruning
15	165	2048	11.5308	2048	11.5308
30	150	2048	11.5308	2048	11.5308
45	135	2048	11.5308	2042	11.8079

3.2 Distinctiveness based pruning the neurons of the Hidden Layer

After training the model till the validation loss stops to decrease, a three-stage pruning process is done as mentioned in the section 2.5. Table 2 below shows activations of 5 neurons for a sample of 3 training examples. As we can see

Table 2. Table showing 5 Neuron Activations for 3 Training Examples.

Training Example	Neuron-1	Neuron-2	Neuron-3	Neuron-4	Neuron-5
1	0	0	0.3795	0	0.4385
2	0.4773	1.0037	0	0	0
3	0	0.9520	0.2230	0	0.5684

from Table 2 that neuron-4 can be removed if it has the same activations for the remaining training examples as well. We calculate the angle between all pairs of neuron activations as shown in Table 3. We can see that the angle between the 3rd neuron and 5th neuron is very less so we can remove one of them and add the parameters of the removed neuron to the available one. We didn't show the complementary pair of neurons in Table3 as we considered only 3 training examples. Table 4 below shows the accuracy of the models to the number of neurons pruned. As

Table 3. Table showing Angle between different Pair of Neurons.

Neuron Pair	Vector Angle(Degrees)
1,2	43.48
1,3	90
1,5	90
2,3	69.59
2,5	56.98
3,5	21.92

we can see from the table below pruning neurons according to how distinct they are doesn't contribute a lot to the drop in accuracy but still decreases the number of parameters as each neuron has the number of weights equal to the number of input features and a bias.

Table 4. Table showing the Accuracy vs Number of Neurons Pruned.

Minimum Angle Threshold (Similarity)	Maximum Angle Threshold (Complimentary)	Number of Neurons before Pruning	Accuracy before Pruning	Number of Neurons After Pruning	Accuracy after Pruning
15	165	4096	11.5308	4096	11.5308
30	150	4096	11.5308	4095	11.5509
45	135	4096	11.5308	3737	11.5308
60	120	4096	11.5308	283	5.9436

3.3 Fuzzy C-means clustering based pruning of neurons

We did few experiments by setting different values for the initial number of clusters required for doing Fuzzy C-means clustering on the activations of the neurons. As we can see from the table 5 we get test accuracy as good as or even better than using 4096 neurons for the hidden layer when we just use 194 neurons as well. This gives us an idea that we can actually remove the hidden layer and also get test accuracy as good as using the hidden layer or else we can just use those 194 neurons for our hidden layer which is less than 5 percent of what we were using before if the test accuracy is more when compared to the network without a hidden layer but that's not the case as we know the neural network without a hidden layer is performing better for our dataset as mentioned before. So, by comparing Table 4 and Table 5 we can see that using the method of Fuzzy C-means clustering effectively prunes neurons and also improves the performance of our neural network by giving better test accuracy compared to distinctiveness based pruning. But few discrepancies can be seen from Table 5 like the test accuracy of using 93 neurons is more than the test accuracy of using 94 neurons. We don't have solid evidence for what's the initial number for the clusters to choose. We consider half the hidden layer neurons number as the initial number of clusters as a good start.

Table 5. Table showing the Accuracy vs Number of Neurons Pruned.

Number of Neurons before Pruning	Number of clusters mentioned initially	Number of clusters obtained after Fuzzy C-means Clustering	Number of Neurons after Pruning	Test Accuracy
4096	2048	194	194	11.849
4096	1024	94	94	4.732
4096	512	93	93	7.821

3.4 Comparison of time complexity between Fuzzy C-means Clustering-based pruning and Distinctiveness based pruning

As we know for Distinctiveness based pruning we need to calculate the angle between each neuron activation with all other neuron activations. So, the time complexity of Distinctiveness based pruning is quadratic time complex which is high when compared to Fuzzy C-means Clustering-based pruning (The time complexity changes with the number of clusters). We also have a predefined function for Fuzzy C-means which makes it more efficient in terms of coding and time taken to run the algorithm in python. In our experiments, we also found that retraining a model with neurons pruned by Fuzzy C-means clustering-based pruning decreases more neurons compared to Distinctiveness based pruning which in turn makes the retraining much faster.

4 Conclusion and Future work

In this paper, we compared the performance of Distinctiveness based pruning with the novel method of Fuzzy C-means clustering-based pruning. Fuzzy C-means clustering-based pruning not only effectively prunes the number of neurons but also gives better test accuracy when compared to the Distinctiveness based pruning and also gives an idea of whether to use the hidden layer or not.

Future work involves comparing the performance of soft clustering techniques like Fuzzy C-means clustering-based pruning technique, Gaussian Mixture Model(GMM) [12] clustering-based pruning technique, Fuzzy Gaussian

Mixture Model(FGMM) [13] clustering-based pruning technique with Distinctiveness based pruning on also other neural network architectures like Convolutional Neural Networks where pruning can be applied to the convolution filters [14] and also trying different other types of pruning techniques like Badness [15] where neurons are pruned by checking the Badness factor. It's also worth solving the problem of finding a good initial number of clusters for any of the clustering-based pruning techniques. Using these techniques for pruning both the input features and neurons on different other datasets can also be considered in future research.

References

1. Dave, VS., Dutta, K.: Application of feed-forward neural network in estimation of software effort. IJCA Proceedings on International Symposium on Devices MEMS, Intelligent Systems & Communication (ISDMISC)(5). pp. 5–9 (2011).
2. Zhou, W., Dong, L., Bic, L., Zhou, M., Chen, L.: Internet traffic classification using feed-forward neural network. 2011 International Conference on Computational Problem-Solving (ICCP). pp. 641-646. IEEE, (2011).
3. Muthukrishnan, R., M, V.A., Shanmugasundaram, H.: Image classification using convolutional neural networks. 2011 International Conference on Computational Problem-Solving (ICCP). vol. 119, 17, pp. 1307–1319 (2018).
4. Nowak, J., Taspinar, A., Scherer, R.: LSTM recurrent neural networks for short text and sentiment classification. International Conference on Artificial Intelligence and Soft Computing. pp. 553–562. Springer, (2011).
5. Gedeon, T.D., Harris, D.: Progressive image compression. [Proceedings 1992] IJCNN International Joint Conference on Neural Networks. vol. 4, pp. 403-407. IEEE, (1992).
6. Pruning Deep Learning Models for Success in Production. Neural Magic, https://www.youtube.com (2020).
7. Yao, Y., Zheng, L., Yang, X., Naphade, M., Gedeon, T.D.: Simulating Content Consistent Vehicle Datasets with Attribute Descent. arXiv preprint arXiv:1912.08855, (2019).
8. Maaten, L.V.D., Hinton, G.: Visualizing data using t-SNE. Journal of machine learning research. vol. 9, num. 11, (2008).
9. Caelli, J.: Neural Network Reduction: Practical Application of Neural Network Topology Pruning Techniques.
10. Zeiler, M.D.: Adadelta: an adaptive learning rate method.arXiv preprint arXiv:1212.5701. (2012).
11. Bezdek, J.C., Ehrlich, R., Full, W.: FCM: The fuzzy c-means clustering algorithm. Computers & geosciences, Elsevier, vol. 10, num. 2-3, pp. 191-203 (1984).
12. Bouman, C.A., Shapiro, M., Cook, G.W., Atkins, C.B., Cheng, H., Dy, J.G., Borman, S.: Cluster: An unsupervised algorithm for modeling Gaussian mixtures. (1997).
13. Ju, Z., Liu, H.: Fuzzy Gaussian mixture models. Pattern Recognition, Elsevier, vol. 45, num. 3, pp. 1146-1158 (2012).
14. Huang, Q., Zhou, K., You, You, S., Neumann, U.: Learning to prune filters in convolutional neural networks. 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 709–718 (2018).
15. Gedeon, T.D., Harris, D., Network Reduction Techniques, In: Proc. Int. Conf. on Neural Networks Methodologies and Applications, AMSE, San Diego, vol. 2, pp. 25-34 (1991).