Evolutionary Algorithm and Distinctiveness-Based Pruning on Depression Predictions using a Feed-forward Network

Clarence Wei Chun Lee

Research School of Computer Science, Australian National University
Canberra, Australia

U6556361@anu.edu.au

Abstract. Distinctiveness-based pruning is a well-known pruning method for neural networks with the goal of reducing network complexity while retaining the accuracy of the initial model. In addition, evolutionary algorithms are widely used for feature selection purposes to reduce the size of neural networks. In this study we compare the performance of cosine similarity and L2-norm-based similarity pruning methods with basic random pruning algorithms, while combining them with different feature selection methods. The pruning methods are found to perform very similarly to each other in this model and dataset, successfully reducing the number of neurons within the network by up to 70% in combination with evolutionary algorithm-based feature selection while retaining accuracy.

Keywords: Neural network, Evolutionary algorithm, feature selection, Distinctiveness-based pruning, Cosine similarity, Euclidean distance, L2 Norm

1 Introduction

Depression is a common mental illness which can adversely affect and limit a person's psychosocial and daily functioning, reducing their quality of life [4]. It manifests in the form of persistent feelings of sadness, lack of motivation in doing typical tasks in daily life, and suicidal thoughts, attempts and behaviors in more severe cases [1].

There are two main classificatory diagnostic systems (International Classification of Diseases [ICD], and Diagnostic and Statistical Manual of Mental Disorders [DSM]) [4], but these can be subjective and time-consuming, largely relying on questionnaires that are reported by the patients themselves, sometimes with assistance from a clinician [1]. Being reliant on the patient's own honestly and willingness to report the truth, these diagnoses can be highly biased and inaccurate.

Research by a team from the Research School of Computer Science in the Australian National University of Canberra have studied the possibility of utilizing neural networks and its recent advancements to come up with a more objective method to detect and classify different levels of depression, by utilizing data collected using several sensors and cameras observing the physical response of a human observing a potentially depressed person reading lines of text through video [1]. The physical response data recorded include pupil dilation, skin temperature, and galvanic skin response, and it is found that there is a highly significant correlation between these data and the severity of depression of the person observed. Using a combination of genetic algorithm and neural networks, the team achieved overall accuracies of up to 92%, significantly better than subjective predictions made by the test subjects at 27%.

In this study, I utilize the data collected from the previously mentioned research to study the effects and usefulness of applying distinctiveness-based pruning techniques to neural networks used to perform classification. Pruning neural networks is the act of systematically removing parameters from a network with the goal of creating smaller networks which perform similarly to the original system. The need for pruning stems from the enormous amounts of computation and memory required by many neural networks, which makes the amount of available resources and time a bottleneck for many people and organizations attempting to work with neural networks [5].

In addition to that, an evolutionary algorithm is also used to select the 20 most important features out of the 85 that are found after combining all the data collected. Evolutionary algorithms are heavily inspired by biological evolution, such as reproduction, mutation, recombination, and selection. This should help to further improve the efficiency of the neural network by reducing the number of parameters in the network.

2 Properties

There are many different pruning methods that have been used in the past, like random pruning and pruning by inspection [2]. This research specifically investigates distinctiveness-based pruning, which works by inspecting the neural network and selectively removing parameters based on how distinct they are from the overall set.

Distinctiveness of a parameter in this context means how different the parameter is compared to other parameters within the same network layer. This degree of difference is calculated by constructing a vector for the weights of each neuron in a layer, and then using a similarity or distance function to calculate how different they are compared to each other.

The use of an evolutionary algorithm to select the 20 most important features is also investigated in this study. In this context, the *importance* of a feature refers to how well a feature can contribute towards the overall effectiveness of a model. In essence, the use of the most important features to train a model will lead to a more accurate model when compared to a model trained with less important features.

3 Method

The main objective of this research is to find out how effective distinctiveness-based pruning and feature selection using evolutionary algorithms are at reducing network size. That is, what percentage of parameters are we able to remove, and whether we can discard a significant proportion of features without affecting the accuracy of the unmodified model too significantly. Some studies, like the one done by Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews from John Hopkins University have found that weight pruning in the low levels (30-40%) do not increase pre-training loss or affect transfer to downstream tasks in a transfer learning model at all, while medium to high levels of pruning start to show increasingly observable loss of the ability for the model to transfer pre-training information downstream [6].

The secondary objective is to study the difference in performance among distance/similarity functions, specifically between cosine similarity, L2 norm, also known as Euclidean distance. In a note written by Chris Emmery, it is stated that cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter [7]. A random pruning algorithm is also included to be used as a baseline to study how much of an advantage it is to use more elaborate pruning methods.

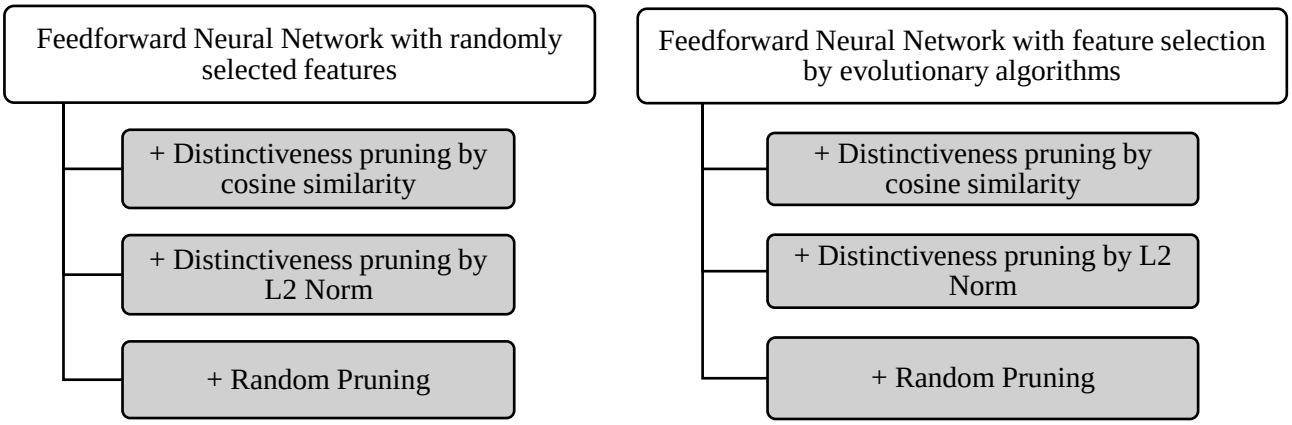

Fig. 1. Study on 3 different pruning methods with and without feature selection by evolutionary algorithms

3.1 Network Structure

The classification problem is first modelled on a fully connected feedforward neural network with backpropagation. Three datasets from all the data provided (pupil dilation, galvanic skin response, and skin temperature) [1] are imported, with every column except the test subject identifier and output used as features for the model. From this dataset, 20 features are picked either randomly (to be used as the baseline) or by using evolutionary algorithms. The data is then split into a train set with 85% of the data and a test set with 15% of the data for final evaluation.

This results to 20 inputs, 2 hidden layers with 13 neurons and 8 neurons respectively, and 4 neurons on the output layer corresponding to the 4 different levels of depression severity classifications. The softmax activation function is used for the final layer, while the other layers utilize leaky relu. Other settings include cross entropy loss as the loss function and stochastic gradient descent for the optimizer.

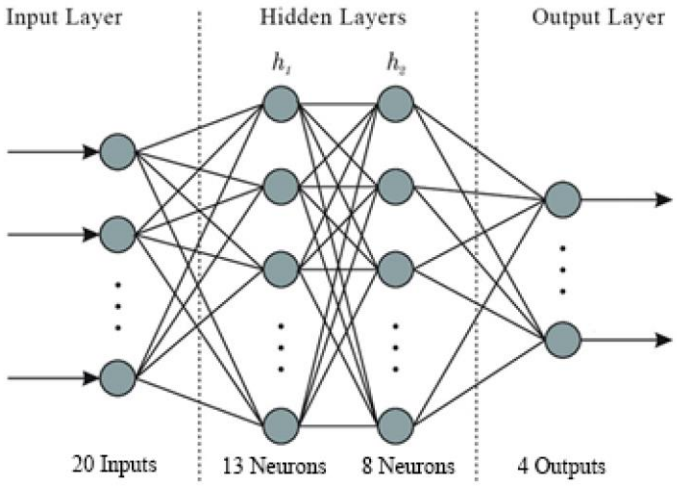

Fig. 2. Feedforward Network Diagram

3.2 Base model fine-tuning and evaluation

The process of fine tuning the base model involves testing the accuracy of the model using several different combinations of epoch counts, momentum settings, and its learning rate. For this purpose, 20 random features were picked out to be used for testing. For each combination, the model is run 10 times and the average accuracy is recorded. Note that there is a focus on minimizing the epoch count as this model will be used as the fitness function during feature selection using evolutionary algorithms.

Table 1. Test accuracy for different settings configurations.

Epoch count	Momentum	Learning rate	Mean test accuracy in % (10 runs)
2000	None	0.15	24.14
2000	0.9	0.15	25.86
2000	0.9	0.1	26.90
2000	0.9	0.05	24.48
2000	0.9	0.07	27.59
3000	0.9	0.06	28.28
3000	0.9	0.06	25.52
3000	None	0.05	25.86
3000	None	0.1	24.48

Despite 3000 epochs with momentum of 0.9 and learning rate of 0.06 returning the best results, ultimately it was found that using 2000 epochs, momentum of 0.9, and a learning rate of 0.07 gave the best balance between performance and accuracy. These settings are used for the base model for our following tests on different pruning methods, as well as for the fitness function for the evolutionary algorithm.

3.3 Feature Selection

In this study, feature selection is the act of selecting a subset of size 20 out of the 85 available features for training and testing the network. Two feature selection methods are used: evolutionary algorithm-based feature selection, where each gene in the chromosome represents one feature, and random feature selection, where features are randomly selected and used throughout the rest of the experiment. Random feature selection is used as a baseline to find out how effective the evolutionary algorithm-based method is.

The steps used for evolutionary algorithm-based feature selection can be summarized as shown below:

1. Each feature in the dataset is given an index number
2. An initial population of 50 individuals is generated, each with a random array of length 20, with each integer in the array representing the index number of a feature
3. The fitness of each individual is evaluated by using their set of features to train and test the base neural network, with their resulting accuracies becoming their fitness scores
4. Random two-point crossover is used for mating
5. Mutation of each child occurs at the probability of 0.1, where if a gene is mutated, it is replaced with a feature that is not present within the chromosome
6. Repeat for 400 generations, and find the best individual

3.4 Distinctiveness Pruning

Distinctiveness pruning can be done in several ways. The technique used in this study is heavily inspired by the following steps as found in Guanjie's research paper [3]:

1. After several epochs of training, retrieve the hidden layers' activation output
2. Normalize all the outputs by -0.5
3. Build vectors for each hidden unit with the dimension as the number of training set length
4. Calculate the vector angle between each two vectors
5. If two vectors' angle smaller is than 15 degrees, they are considered sufficiently similar, and one of them should be removed. The removed neuron's weight should be added into the kept one
6. If two vectors' angle is greater than 165 degrees, these two neurons are effectively complementary. They can be removed simultaneously

In this study, several modifications were made to the above steps to fit the network and program structure. Mainly, normalization is not done, and cosine similarity or L2 norm is used in place of vector angles. A similarity threshold variable is also being used as a hyperparameter to tune how similar two vectors must be in order to be pruned. When neurons are pruned, weights are totally removed and not added into any other neurons. A summary of the steps taken is as follows:

1. After a set number of epochs of training, retrieve the hidden layers' input weights
2. Calculate the similarity or distance between each pair-combination of neurons in a particular layer
3. If two neurons have a similarity value above the set threshold, the corresponding weights of one of those neurons in the next layer will be set to 0
4. Once pruning is completed for all layers, resume training using the pruned network

Cosine similarity and L2 norm (Euclidean distance) can both be used to measure distinctiveness. However, the main difference between the two is that cosine similarity measures the cosine of the angle between two vectors, which range from -1 for totally opposed (180°), 0 for a right angle (90°), and 1 if they point to the same direction (0°), while Euclidean distance measures the distance between two vectors. Hence it is possible for 2 vectors to have a cosine similarity of 1 but have a Euclidean distance of more than 0.

3.5 Random Pruning

For the random pruning model, a set percentage of neurons are randomly selected to be excluded from the network. These neurons will have their weights set to 0, effectively disabling them. The random pruning model will be used as a baseline to help dictate how much more effective distinctiveness-based pruning is.

4 Results and Discussion

A total of 6 models are built using the methods described above, namely the three pruning methods (L2-norm (Euclidean distance) distinctiveness pruning model, cosine similarity distinctiveness pruning model, and the random pruning model), either with randomly selected features, or with features selected using evolutionary algorithms. All results shown are the average over 10 runs, and results are first evaluated based on the test accuracy obtained for the number of neurons pruned from the models. Reasons for variations are discussed on and we will try to understand and explain the results. After results for each model is obtained, they will be compared with each other to study the differences in performance for different pruning and feature selection methods.

4.1 Results for each model

Distinctiveness-based pruning is applied after several epochs of training. For this test, pruning for all methods are applied after 2000 epochs. The similarity threshold is adjusted to control the number of neurons pruned for each run. Results obtained for distinctiveness-based pruning using cosine similarity are shown in the table below (Note: "threshold" refers to the similarity threshold, where neurons with similarities above this threshold are pruned):

Table 2. Randomly selected features with distinctiveness-based pruning using cosine similarity results.

Threshold	Avg no. of neurons pruned	Avg % of neurons pruned	Avg test accuracy in %
0.25	14.0	70.0	24.21
0.4	11.0	55.0	23.87
0.5	4.5	22.5	27.13
0.6	1.3	6.5	28.33

Table 3. Evolutionary algorithm selected features with distinctiveness-based pruning using cosine similarity results.

Threshold	Avg no. of neurons pruned	Avg % of neurons pruned	Avg test accuracy in %
0.25	14.2	71.0	33.24
0.4	10.0	50.0	34.39
0.6	4.3	21.5	34.62
0.7	2.1	10.5	36.20

One important observation from this test is that at a similarity threshold of 0.6 in table 2, only an average of 1.3 neurons were pruned at each run, which means for many runs, no neurons were pruned at all. This implies that it is rarely the case where any combination of neurons are more than about 80% similar (given that cosine similarity ranges from -1 to 1). This is in contrast with the usage of distinctiveness-based pruning for image compression [2], where pruning is set to only occur for pairs of vectors that are more than 91.67% similar. This may be because images are more likely to contain data points that are highly similar, e.g.: neighboring pixels that are the same color.

From the results obtained, it can be seen that the test accuracy is significantly better across the board when the evolutionary algorithm is used. In the model with randomly selected features, when we reach 55% of neurons pruned, test accuracy drops to below 25% which means classifications are totally random (given that we have 4 possible outputs). However with EA, the test accuracy remains quite high even at 71% of neurons pruned.

Next, we perform the same test with distinctiveness-based pruning using L2-norm (Euclidean distance). Note that in this case, pairs of neurons with L2 norms *below* the threshold will be pruned, as lower L2 norms indicate smaller distances and greater similarity. All other settings remain the same.

Table 4. Randomly selected features with distinctiveness-based pruning using L2-norm results.

Threshold	Avg no. of neurons pruned	Avg % of neurons pruned	Avg test accuracy in %
0.55	14.5	72.5	25.11
0.50	10.7	53.5	23.10
0.47	4.1	20.5	28.29
0.45	2.2	11.1	28.57

Table 5. Evolutionary algorithm selected features with distinctiveness-based pruning using using L2-norm results.

Threshold	Avg no. of neurons pruned	Avg % of neurons pruned	Avg test accuracy in %
0.55	14.4	72.0	35.22
0.50	10.7	53.5	34.62
0.45	4.5	22.5	34.11
0.42	2.0	10.0	37.10

To achieve a similar number of neurons pruned per run, the threshold for this pruning method is finely tuned in attempt to keep the percentage of neurons pruned within 3% as the previous method. Similarly to the previously tested pruning method, the results across the board are much better with EA.

Next, we implement a random pruning algorithm to be used as a baseline. In this case, the percentage of neurons to be pruned are set. Random pruning only occurs after 2000 epochs so that results are comparable.

Table 6. Random pruning results.

Avg % of neurons pruned	Avg test accuracy in % (No EA)	Avg test accuracy in % (EA)
70.0	24.10	34.22
50.0	24.37	34.37
20.0	26.29	33.60
10.0	28.33	36.50

From these results we observe that with features chosen through evolutionary algorithms, the proportion of neurons pruned seem to have a minimal effect on the test accuracy of the trained model.

4.2 Comparing Models

All the results from individual tests are put in the same table and each row represents different % of neurons pruned.

Table 7. Accuracies of different models compared.

% of neurons pruned	Avg Test Accuracy in %					
	Cosine S.		L2 Norm		Random	
	No EA	EA	No EA	EA	No EA	EA
70 - 73	24.21	33.24	25.11	35.22	24.10	34.22
50 - 55	23.87	34.39	23.10	34.62	24.37	34.37
20 - 23	27.13	34.62	28.29	34.11	26.29	33.60
5 - 12	28.33	36.20	28.57	37.10	28.33	36.50

Looking at the comparison, all the pruning methods perform similarly through every range of neurons pruned. At all of the percentage groups, average test accuracy is within 3% of each other and is within the margins of error. It appears that the pruning method used does not significantly affect the performance of a neural network for classification problems. Afterall, it seems that most of the studies found on distinctiveness-based pruning with positive results are made in the attempt to increase the performance of convolutional neural networks [8], image classification, and image compression models [2], suggesting that distinctiveness-based pruning is best suited for problems involving images.

From another study by Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag [5], it is also observed that while it is shown that many pruning methods outperform random pruning, this is only true for large amounts of pruning on a very large network, and interestingly does not always hold for small amounts of pruning. Pruning all layers uniformly also tend to perform worse than allocating parameters to distinct layers intelligently or pruning globally.

However when comparing the methods used for feature selection (EA and No EA), the improvements are significant and undeniable. The test accuracies of the models trained with features selected using evolutionary algorithms perform consistently better than models that are trained using random features, thus concluding that evolutionary algorithms are highly effective solutions for selecting the best features to train a model, resulting in a much smaller neural network than one that is trained using every feature within the dataset.

However it is important to note that given the relatively small dataset and poor base model accuracy, the margin of error is quite large and it can be very difficult to detect smaller differences in performance. These are some of the many factors that may have affected the results obtained in this study.

5 Conclusion and Future Work

Overall, I have demonstrated that a significant proportion of neurons can be pruned from a neural network without affecting its accuracy. However, the use of different pruning methods seems to have minimal difference in the resulting performance when used for this classification problem. Besides that, feature selection using evolutionary algorithms is a very effective method of reducing the size of a neural network.

Some deeper research into different factors that may affect pruning performance have to be done in future studies, which may include different network sizes, network structures, uniform pruning versus global pruning and so on. The use of a more accurate base network will also help us detect more precise changes in performance between methods. Besides that, more time and resources have to be invested to gain more precise results, as evolutionary algorithms require a lot of time and computing power to complete.

References

1. Xuanying Zhu, Tom Gedeon, Sabrina Caldwell, Richard Jones: Detecting emotional reactions to videos of depression. Research School of Computer Science, Australian National University Canberra, Australia
2. T.D. Gedeon, D. Harris: Progressive Image Compression. School of Computer Science and Engineering, The University of New South Wales, Australia, Centre for Neural Networks, King's College London, U.K.
3. Guanjie Huang: Distinctiveness Pruning on 'fight or flight' Response Prediction LSTM Network. Research School of Computer Science, Australian National University Canberra, Australia
4. Malhi, Gin S; Mann, J John: Depression. The lancet; London (24 November 2018)
5. Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, John Guttag: What is the State of Neural Network Pruning? (2003)
6. Mitchell A. Gordon, Kevin Duh, Nicholas Andrews: Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning. John Hopkins University (2002)
7. Chris Emmery: Euclidean vs. Cosine Distance (2017) Ref: https://cmry.github.io/notes/euclidean-v-cosine
8. Wenrui Li, Jo Plested: Pruning Convolutional Neural Network with Distinctiveness Approach (2019)