Facial Expression Recognition and Distinctiveness Angular Measures for Pruning

Namitha John

Research School of Computer Science,
The Australian National University
u6387841@anu.edu.au

Abstract. This paper proposes two different methods to improve facial expression recognition on two versions of the same dataset. The first method uses a simple feed-forward neural network consisting of one hidden layer and the second method extends the first method by applying a more complex convolutional neural network. The models are initially improved through the fine-tuning of hyper-parameters and then a pruning technique is applied to the model which eliminates similar and opposite hidden neurons using the distinctiveness angular measure heuristic. Both versions of the datasets are extracted from the static facial expression database Static Facial Expressions in the Wild (SFEW). The results from this paper when applying the pruning technique in Method A indicate that the pruning technique is satisfactory and doesn't significantly deteriorate model performance. Furthermore, the results from Method B support the pruning technique as there aren't any observable relationships which show a deterioration of testing accuracy as more neurons are pruned.

1 Introduction

Facial expressions are the facial changes that occur in response to a person's internal emotional state, intentions, or social communications. Facial expression recognition and the analysis of facial expressions have been made possible through advances in the field of computer vision and machine learning over the years. The main applications of facial expression analysis are human computer interaction, human behaviour analysis and lie detection. Additionally, it can be used in the medical field to pain monitor patients and for the analysis of stress, anxiety and depression. In this paper, we use a static facial expression database, Static Facial Expressions in the Wild (SFEW) [2]. The SFEW dataset is extracted from another temporal dataset, Acted Facial Expressions in the Wild (AFEW) [2]. The dataset used for this paper contains 675 images of individuals and each individual's facial expression has been labelled as one of seven basic expressions - angry, disgust, fear, happy, sad, surprise or neutral [2].

In this paper, we apply machine learning models on different versions of this dataset and fine tune hyper-parameters to improve the performance of the model. Additionally, we investigate the effects of applying a pruning technique to that model. The first version of the dataset includes the first five principal components of the Local Phase Quantisation and the Pyramid of Histogram of Oriented Gradients with the label of the image. To this dataset we apply a simple three layer neural network. The second version of the dataset includes the original images themselves with the labels and a more complex convolutional neural network is utilised.

Pruning is a method that removes redundant hidden neurons from a network. Deciding the number of hidden neurons is a difficult task when training networks and if the number of hidden neurons is too high then back-propagation can be extremely slow and in turn cause the training of such networks to be slow. These extra neurons duplicate the functionality of other neurons and thus aren't important and can be omitted from the network. Pruning can be done using brute force methods such as by inspection, however, a more automated process is preferred which is why various pruning techniques have been developed.

2 Method

This section discusses the steps that were taken to perform the experiment and the methods that were used to evaluate the performance of the method.

2.1 Method A

The first version of the dataset includes the first five principal components of the LPQ and PHOG descriptors which were extracted from the original images and used as the the ten input features for our simple three layer neural network.

We compute the testing accuracies for each of the different methods to measure performance. If the model performs well on unseen data, it suggests that our model is acceptable. We will take a higher test accuracy as an indication of a better model. Additionally, we also observe training accuracy and training loss to ensure that they are increasing and decreasing, respectively with each epoch.

Baseline Method

The first aim of this paper is to try to improve the base model before applying any pruning techniques on the SFEW dataset given the principal components of LPQ and PHOG descriptors. Local Phase Quantisation (LPQ) is a blur insensitive texture classification method which is based on a quantized phase of the discrete Fourier transform (DFT) computed in local image windows [5]. It is an extension to Local Binary Pattern (LBP) and has been shown to perform better [2]. The Pyramid of Histogram of Oriented Gradients (PHOG) is an extension to Histogram of Oriented Gradients (HOG) and calculates the gradient orientations at localised regions of an image [1].

The initial stage of this two part process of developing a satisfactory model for the facial expression dataset involved setting up the three layer neural network with mini-batch gradient descent, determining the hyper-parameters that would produce optimal results and establishing an acceptable activation function, loss function and optimizer. The hyper-parameters that required fine-tuning were the number of neurons in the hidden layer, the learning rate, the batch size, and finally the number of epochs the neural network would train for. These hyper-parameters were confirmed by calculating the test accuracies for a set of various potential values using 2-fold cross validation and averaging over 10 iterations. For our final model the number of neurons in the hidden layer is set to 20, the number of epochs is set to 500, the batch size to 20, and the learning rate is set to 0.01.

We use k-fold cross validation throughout this method instead of a simple train-test split to reduce bias in our results. The 2-fold cross validation method divides the dataset equally into two subsets. In the first iteration, training is performed on the first subset and the model is tested on the second subset and in the second iteration, the second subset is used for training. The test accuracy is computed as the average of the two results.

The model was trained five times on each of the potential values and the average testing accuracy was determined. Finally, the ReLu activation, cross entropy loss, and SGD optimizer were identified as the most appropriate choices after comparing the performance of the models against other activation functions, loss functions and optimizers such as Tanh and Sigmoid activations, and the Adam optimizer.

5-fold cross validation was also applied to attain more definitive results and is a similar method to that described earlier for 2-fold cross validation, the only difference being that it splits the data into five subsets instead of two. In this method, the model is trained on four out of the five subsets of the data and then tested on final subset. This is repeated until all five subsets have been a test set at least once.

Pruning Method

Once a satisfactory model was generated, the latter stage of the process determining the effectiveness of the pruning technique. The aim of this stage of the paper is to explore different methods of applying the pruning technique such as using activation vectors or weight vectors and whether or not retraining of the model is required after the removal of neurons. We compared various methods outlined earlier to improve the performance of the model after pruning was applied such as k-fold cross validation. All of our pruning technique comparisons mentioned are based on models that have been trained using 2-fold cross validation and averaged over ten iterations. Our results are finally confirmed on the 5-fold cross validation method.

The reduction of the size of the hidden layer is performed using the distinctiveness angular measure. The distinctiveness of hidden neurons is determined by computing the output activation vector of those neurons for each of the inputs in the dataset [3]. Thus, the activation vector will be of the same dimensionality as the number of inputs where each component of the vector corresponds to the activation for a particular neuron [3]. This represents the functionality of the hidden neurons in the input space [3]. Similarly, the distinctiveness angular measure was also applied to the weights of the network to determine redundant hidden neurons and the two pruning methods were compared.

Before computing the angular separation, the activation vectors or the weight vectors are required to be normalised to ensure that the angles remain within the desired range [3]. Usually, angular separations less than the threshold of 15°are considered to be similar and one of the hidden neurons can be removed and the weight of the neuron that is to be removed is added to the weight of the remaining neuron [3]. However, for low angular separations, this averaging effect is insignificant and will not deteriorate the error measure [3]. This produces a network consisting of one less hidden neuron and one that requires no further training [3]. Angular separations of neurons that are over 165°are complementary and both of them can be removed [3]. If more significant neurons are removed at each stage that have angular separations greater than 15°, the model is required to be retrained after the removal of each neuron [3].

In attempts to determine which distinctiveness measure yields better performance we use the 'hidden layer to output layer' weight vector when calculating the distinctiveness measure and compare this to the results obtained when the activation function is used. We also compare the effect on test accuracy when the model is and isn't retrained once a neuron has been removed. This is done by retraining the model for another 500 epochs each time a single hidden neuron has been pruned.

2.2 Method B

The second version of the dataset includes the original images of the faces as the input and a label indicating the facial expression shown in the image.

We compute the test accuracies for each of the different methods to measure performance. If the model performs well on unseen data, it suggests that our model is acceptable. We will take a higher test accuracy as an indication of a better model. Additionally, we also observe training accuracy and training loss to ensure that they are increasing and decreasing, respectively with each epoch.

Baseline Method

The initial goal of the second method before any pruning method is applied is to ensure that the base model is satisfactory. Since the dataset consists of the original images of the various facial expressions, a deep convolutional neural network is applied to extend the previous method. The basic structure of the model can be described as two deep convolutional layers followed by two fully connected layers.

A convolutional neural network is a deep learning algorithm that takes images as inputs and assigns importance (learnable weights and biases) to various aspects in the image and be able to differentiate one from the other [6]. When utilising a dataset of images convolutional neural networks are preferred over standard neural networks because it is able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters [6].

The neural network consists of two convolutional layers, with max pooling applied with a kernel size of 2 to down-sample the input and reduce the computational cost [6]. Additionally, for each convolution the output of the max pooling layer is fed into the ReLu activation function. From previous results obtained in Method A, the ReLu activation function generally produces optimal results for our dataset. Finally, the output from the second convolutional layer is passed into a simple neural network with one hidden layer which then outputs one of 7 values indicating the classification of the image. The size of the hidden layer between the two fully connected layers is set to 50, the learning rate is 0.01, and the batch size is 20 to be consistent with Method A. A diagram of the deep neural network is displayed in Figure 1.

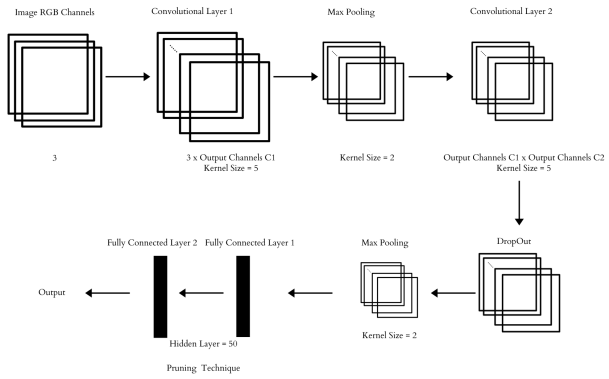


Fig. 1: Structure of Deep Neural Network

The number of output channels indicates the number of features that can be learned by the model. To determine an optimal number of output channels for each of our convolutional layers, we train the model on a range of potential values and decide on 10 for the first convolutional layer and 20 for the second convolutional layer. The kernel size of both convolutions are specified as 5 to maintain a good compromise between a very small filter size and a bigger one that may leave out smaller details. Finally, we train the model for 10 epochs to ensure that the the model is trained for an adequate amount of time before measuring test accuracy.

A simple 8:2 train, test split is used to determine the number of output channels to specify for the convolutional layers. We then utilise the 5-fold cross validation training method to reduce any bias in the results and to confirm the performance of the baseline model.

Pruning Method

The same pruning method used in Method A is applied to the extended model to compare if the difference in dataset and a more complex neural network will yield different results to the results generated when a simple neural network is used. An 8:2 train, test split is used initially to train the model to determine trends in the data. Finally, the results are confirmed on the model with a 5-fold cross validation method.

Neurons are pruned based on the distinctiveness angular measure. The activation and weight functions are normalised as before and neurons are removed if their angular separations are less than 15°or greater than 165°. Retraining of the model is required if more significant neurons are removed that have angular separations greater than 15°.

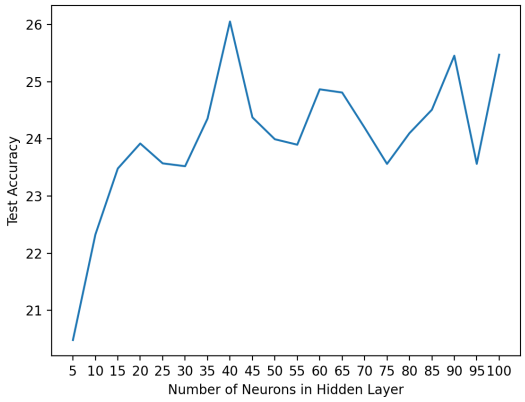
The activation vector is calculated by extracting the hidden outputs from the hidden layer once the ReLu activation function has been applied that exists between the two fully connected layers in our deep neural network. The weight vector is defined as the 'hidden layer to output layer'. The results between using the activation vector and the weight vector are compared. We also compare the effect on test accuracy when the model is and isn't retrained once a neuron has been removed. This is done by retraining the model for another 3 epochs each time a single hidden neuron has been pruned.

3 Results and Discussion

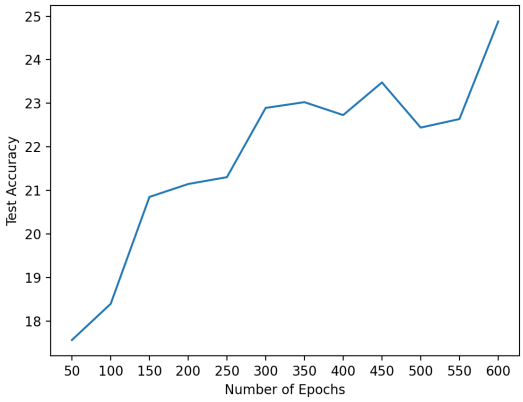
3.1 Method A

Baseline Method

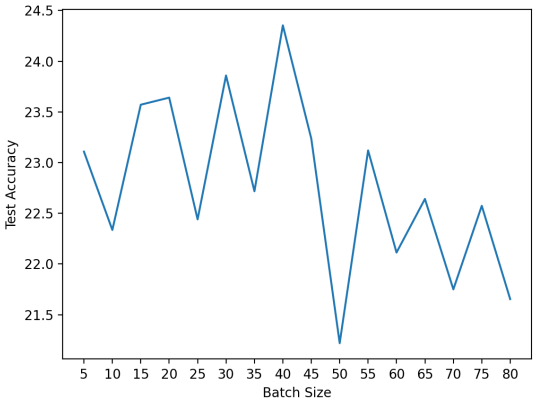
Our initial tests consisted of determining the hyper-parameters that would generate good results before applying any pruning techniques to the training process. All the initial testing is done with the 2-fold cross validation training method and finally results are confirmed using 5-fold cross validation to reduce any bias. The average test accuracies for different potential values of the hyper-parameters are displayed in Figure 2. In Figure 2a, it can be observed that generally a hidden layer that is greater that 35 seems to output higher test accuracies. In Figure 2b, there is a strong positive correlation between the number of epochs and the test accuracy. In Figure 2c and 2d, there aren't any obvious relationships however, batch sizes greater than 50 have very slightly lower test accuracies with batch sizes less than 50 and a learning rate of 0.01 generates the best performance. Using these results and our own intuition, we use the hyper-parameters specified in the Method section.



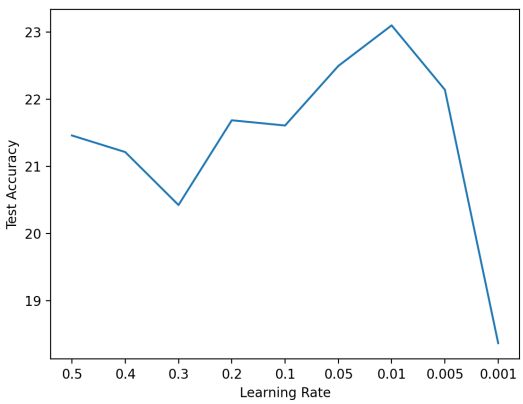
(a) Different Sizes of the Hidden Layer



(b) Different Numbers of Epochs



(c) Different Batch Sizes



(d) Different Learning Rates

Fig. 2: Test Accuracies (%) for Various Hyper-Parameters

We achieve a test accuracy of 26% using 2-fold cross validation and 31% using 5-fold cross validation. Comparing all of our test accuracies computed to the results calculated in the paper, we notice that the classification accuracies presented in the paper are higher at than what we were able to achieve [2]. The paper presents accuracies of 43.71%

for LPQ descriptors and 46.28% for PHOG descriptors [2]. A potential reason for is that the dataset paper uses a support vector machine with a radial basis function kernel for classification, whereas we implemented a simple neural network [2]. Another reason could be due to the inputs that were fed into our simple network as we only used the first five principal components for each of our descriptors and using the original images may give better performance.

Pruning Method

Initially, the minimum angles between hidden neurons are calculated for each neuron that is removed. Figure 3 displays the minimum angles between neurons when each hidden neuron is removed for the first 15 hidden neurons. We can see the angles for the activation vectors are generally much higher than the angles for the weight vectors. The minimum angles for the first two subsequent hidden neurons have a large jump and then the angles are closer together after that for each hidden neuron removed. The weight vector starts of at smaller minimum angles and the angles increase by small amounts each time a neuron is removed.

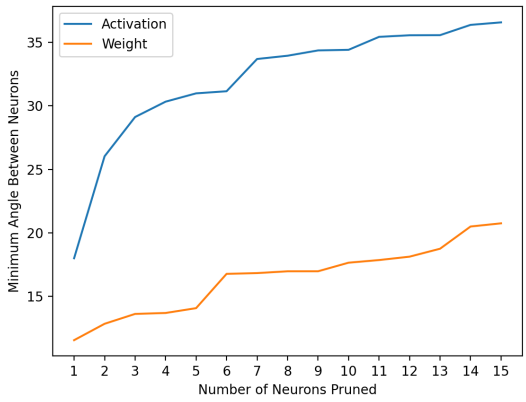
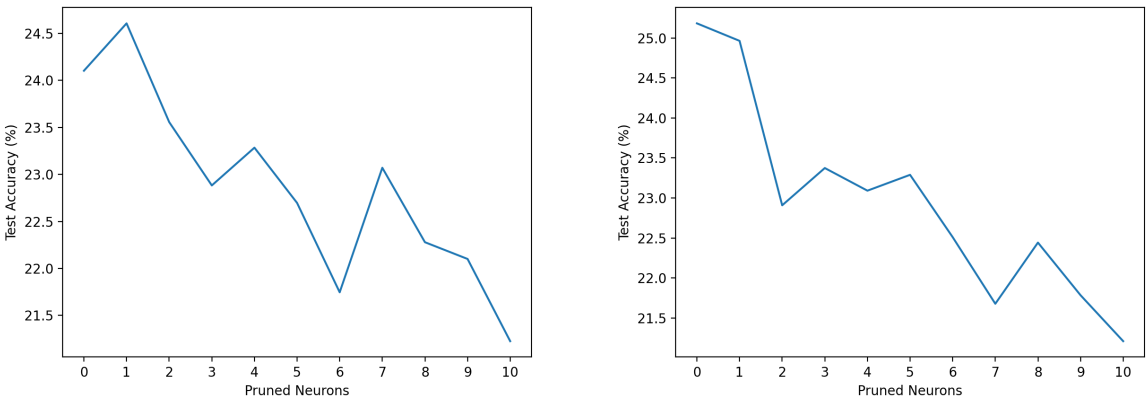


Fig. 3: Minimum Angle Between Neurons

We apply the neuron pruning technique with our 2-fold cross validation method which removes redundant neurons based on the distinctiveness heuristic from the neural network based on the activation vector. Figure 4 displays the performance of the model on the test dataset at each stage of the removal with and without retraining. Across both Figure 4a and 4b, we see that generally for each neuron removed the test accuracy decreases. However, we can observe the accuracy does not decrease by a significant amount. Comparing the graphs when no retraining is performed once a neuron is removed and when the model is retrained, we can see that there aren't any notable differences between the two which may suggest that retraining is not required.

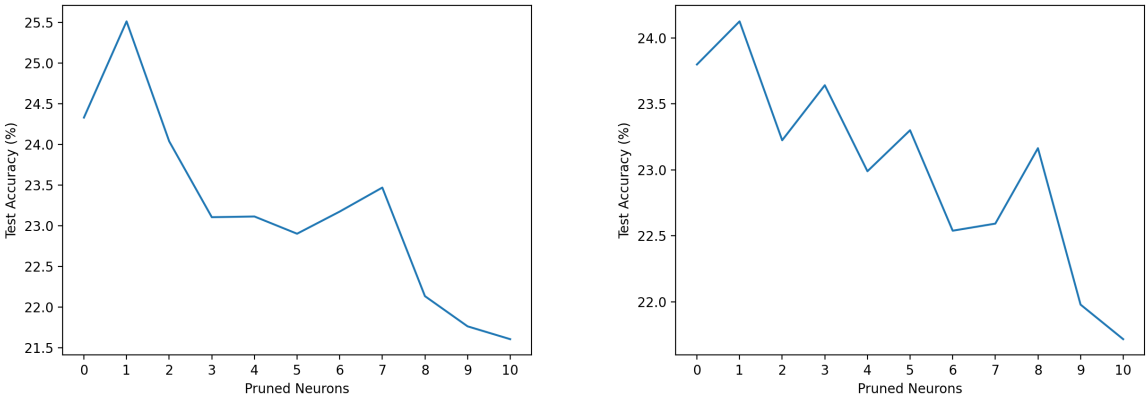


(a) No Retraining After Neuron Removal

(b) Retraining After Neuron Removal

Fig. 4: Pruning with Activation Vector Using Two-Fold Cross Validation

There are other methods to calculate the distinctiveness measure when pruning neurons such as using a weight vector instead of the activation vector. The results are shown in Figure 5 below for when the model is and is not retrained after the removal of neurons using weight vectors. The weight vector generates similar graphs to that for the activation vector with test accuracy generally decreasing for each neuron that is removed. However, for each neuron that is pruned the decrease in accuracy is not significant. Comparing Figure 5a and Figure 5b, without and with retraining of neurons respectively, we can see that the there doesn't seem to be any striking differences in the test accuracies although it seems that test accuracy alternates between increasing and decreasing with every neuron that has been pruned.



(a) No Retraining After Neuron Removal (b) Retraining After Neuron Removal

Fig. 5: Pruning with Weight Vector Using Two-Fold Cross Validation

Using the weight vector produces similar results to the results generated when using the activation vector so we can't make any deductions on whether one is better than the other. The total sum of squares computed for the first few neurons removed barely increases in the technique paper [3]. This is different to our results as we do see slight decreases in accuracy for each neuron removed. However, since the decreases aren't significant we can say that there is only a little drop in performance and this supports the results obtained in the technique paper that the pruning method is based on.

3.2 Method B

Baseline Method

We attempt to keep our model consistent between the two methods by setting most of the hyper-parameters to the same values. The only hyper-parameters that needed to be determined in order to maximise the model performance are the number of output channels from the first and second convolutional layers. We investigate this by computing the test accuracies of 6 potential numbers of output channels for convolutional layer 1 and 2. This is displayed in Figure 6. We decide to the set the number of output channels for the first convolutional layer as 10 and the number of output channels for the second convolutional layer as 20. This is because the accuracy is relatively higher than the test accuracies for other combinations and the neural network is not extremely deep which allows for results to be generated in a reasonable amount of time.

Number of Output Channels for Conv2

Number of Output Channels for Conv1	10	15	20	25	30	35
5	22.962963104248047	38.51852035522461	37.03703689575195	41.48147964477539	39.25925827026367	40.74074172973633
10	43.703704833984375	31.851852416992188	42.22222137451172	41.48147964477539	36.296295166015625	35.55555725097656
15	36.296295166015625	37.03703689575195	38.51852035522461	36.296295166015625	40.74074172973633	44.44444274902344
20	31.11111068725586	37.03703689575195	37.77777862548828	40.0	36.296295166015625	40.74074172973633
25	30.370370864868164	37.77777862548828	37.03703689575195	37.77777862548828	38.51852035522461	34.814815521240234
30	39.25925827026367	33.33333206176758	37.03703689575195	38.51852035522461	36.296295166015625	40.0

Fig. 6: Test Accuracies for Different Output Channel Combinations Between The Convolutional Layers

Thus, we are able to achieve a test accuracy of 42% using a simple 8:2 train, test split and 43.26% using 5-fold cross validation for our baseline model. Comparing this result to results from Method A we can observe that our results obtained from our deep neural network are significantly better than the results obtained from our simple neural network. This could be due to the complexity of the neural network improving the model or that we use the original images instead of the principal components as inputs to this neural network. These results are also closer to the results obtained in the dataset paper [2].

Pruning Method

We use the same pruning method that is outlined in Method B which uses the distinctiveness angular measure to remove similar and opposite neurons and apply it to the hidden layer between the two fully connected layers in our deep neural network. Figure 7 shows the minimum angle between hidden neurons for activation and weight vectors when each neuron is removed. Similar to the results from Method A, the minimum angles when using activation vector are generally greater than the angles from the weight vector and the first two neurons removed have larger differences in angles between them compared to others. The increase in minimum angles for subsequent neurons are smaller after the first two are removed. The activation vector has a much smoother line which indicates that the angles are increasing by a consistent amount, whereas the weight vector seems to have a coarser line. These results align with the pruning technique paper and this may suggest that using the activation vector is more effective than the weight vector.

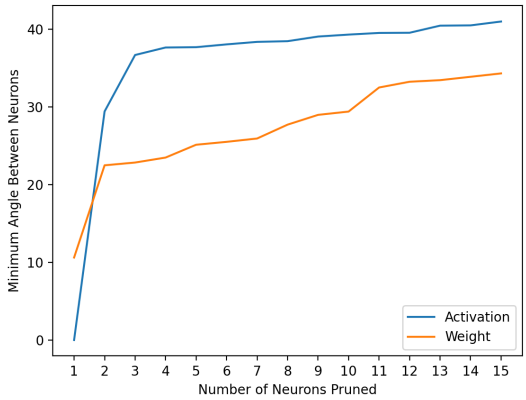


Fig. 7: Minimum Angle Between Neurons

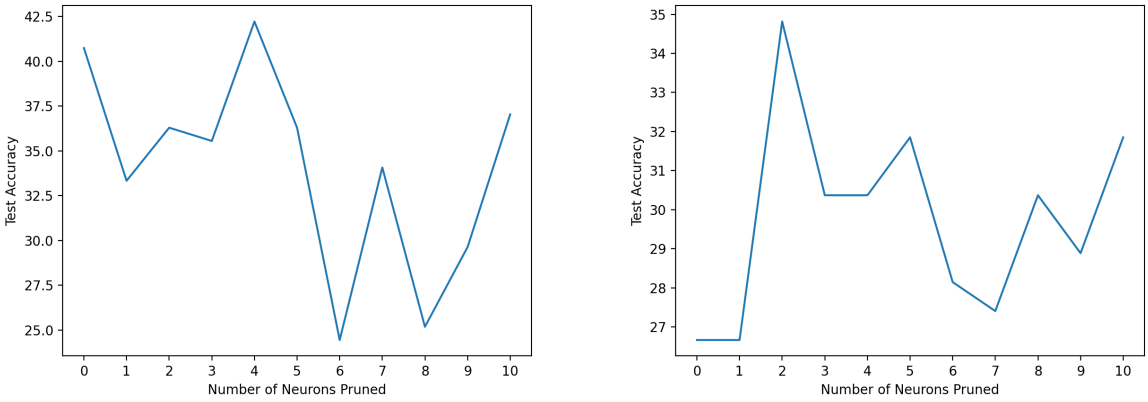
We begin by observing the effects on the test accuracy for each of the first 10 neurons that are pruned using the activation vector. The results for this are shown in Figure 8. We can see that there is no clear relationship between the test accuracies and the number of neurons removed. On the other hand, the results from Method A had a clear negative correlation. Since there isn't any notable drop in performance this suggests that the pruning technique can be applied to this model. However, for models with greater than 5 pruned neurons the accuracies seem to be generally lower than the accuracies for less than 5 pruned neurons. This pattern is consistent across Figure 8a and 8b. Comparing how the accuracies are effected without and with retraining after a neuron is removed we can see they remain within a closer range when the model is retrained in Figure 8b, whereas in Figure 8a the accuracies seem to be more scattered. This suggests that retraining of the model once neurons are pruned may be beneficial for the extended method. Method A showed no improvements when the model was retrained.

Figure 9 shows the effect on test accuracy when pruning is done by calculating the angles between the weight vectors. As with pruning using the activation vector, we cannot observe any sort of relationship between the number of hidden neurons that are pruned and the test accuracies. Similar to Figure 8, the graphs are scattered. However, there is a sudden drop in both Figure 9a and 9b when 8 hidden neurons are pruned.

Since both Figure 8 and Figure 9 show that at least when between 1 and 5 hidden neurons are pruned, there isn't a significant drop in test accuracy. This indicates that the pruning method can be accepted.

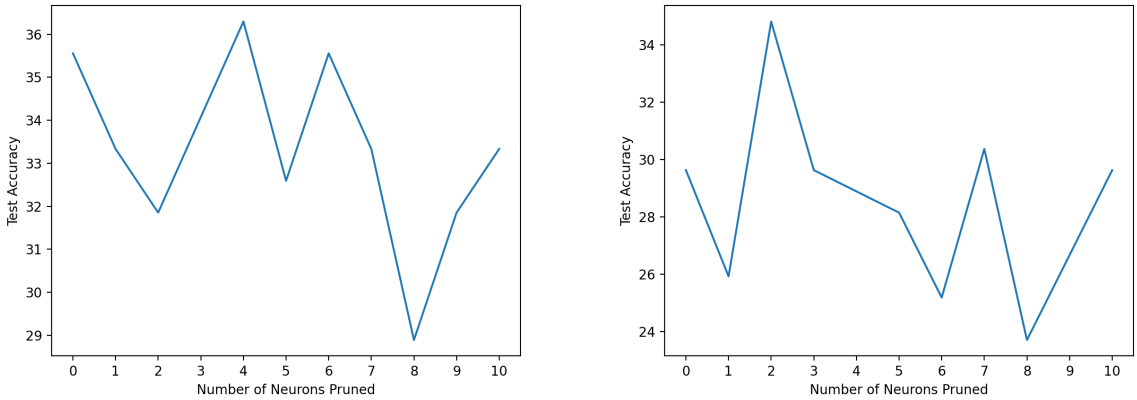
4 Conclusion and Future Work

To conclude, this paper explores the Static Facial Expressions in the Wild (SFEW) dataset for facial expression recognition. The first version of the dataset is explored in the first method which consists of the first five principal components of 2 descriptors, Local Phase Quantisation and Pyramid of Histogram of Oriented Gradients. The second version of the dataset is investigated in the second method which consists of the original images of the facial



(a) No Retraining After Neuron Removal (b) Retraining After Neuron Removal

Fig. 8: Pruning with Activation Vector



(a) No Retraining After Neuron Removal (b) Retraining After Neuron Removal

Fig. 9: Pruning with Weight Vector

expressions. Various techniques are used to improve the baseline models such as fine-tuning of hyper-parameters and K-fold cross validation. Furthermore, a pruning technique was applied to the models to investigate the effects of distinctiveness angular measures for pruning on the SFEW dataset. A comparison of utilising the activation vector and using the weight vector to prune hidden neurons is provided is this paper as well as the effects of retraining of the neural network once a neuron has been pruned.

The experiments conducted show that applying the pruning technique to the model to remove redundant hidden neurons did not significantly deteriorate test accuracy in Model A which used the first version of the dataset. We also observed that there were no effects of retraining on the performance. On the other hand, there wasn't any visible relationship between the number of neurons pruned and the testing accuracy compelling us to conclude that the results from the Method B also support the pruning technique. We do see a slight benefit of retraining the model once a neuron is removed as well.

To improve the baseline model, genetic algorithms can be applied to fine tune hyper-parameters. Additionally, this paper can be extended by applying the pruning technique to a different neural network and observing the effects it has. If this pruning technique can be used across a range of different datasets and neural networks then it can be concluded that the pruning technique is acceptable.

References

1. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). vol. 1, pp. 886–893. IEEE (2005)
2. Dhall, A., Goecke, R., Lucey, S., Gedeon, T.: Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). pp. 2106–2112. IEEE (2011)

3. Gedeon, T.D.: Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. In: Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems. pp. 26–29. IEEE (1995)
4. Mozer, M.C., Smolensky, P.: Using relevance to reduce network size automatically. Connection Science **1**(1), 3–16 (1989)
5. Ojansivu, V., Heikkilä, J.: Blur insensitive texture classification using local phase quantization. In: International conference on image and signal processing. pp. 236–243. Springer (2008)
6. Saha, S.: A comprehensive guide to convolutional neural networks — the eli5 way, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53