Comparison of distinctiveness pruning and genetic algorithms for the reduction of a fully trained neural network

Sara Tanovic

Research School of Computer Science
Australian National University
Canberra, Australia
u6676643@anu.edu.au

Abstract. A three-layer neural network was implemented on images from the Static Facial Expressions in the Wild (SFEW) database for facial expression classification, with 10 input features extracted using PHOG and LPQ. The baseline network had a final classification accuracy of 22.6% across 7 emotion classes. Two pruning methods for network reduction are investigated and compared: distinctiveness pruning, and genetic algorithm (GA) pruning. Distinctiveness pruning aims to identify similar and complementary pairs of hidden neurons using cosine similarity as a measure of distinctiveness. Pruning with a threshold angle of 15 degrees was found to have little effect on network compression, while setting the angle to 45 degrees was able to consistently reduce the network down to its optimal size (20-30 hidden neurons) with a 1% diminishment in accuracy. GA pruning aims to optimize the final combination of trained hidden neurons to improve both accuracy and efficiency. The extent of network reduction was tuned by the penalty coefficient λ of the fitness function, and it was found that using $\lambda = 50$ compressed the network by more than 60% across a range of initial architectures with a 1% increase in accuracy, while using $\lambda = 0$ had a slightly lower compression effect but increased accuracy by 2%. This technique also shows promise for the purpose of post-training network optimization. The most accurate network was constructed by first training a network with 100 hidden neurons, and then implementing GA pruning with $\lambda = 0$ to prune it down to 54 neurons for a final accuracy of 25.2%.

Keywords: neural networks, network reduction, facial expression recognition, distinctiveness pruning, genetic algorithms

1 Introduction

Facial expression recognition (FER) is a classification problem within the fields of artificial intelligence and computer vision. The task of mapping an image of a face to a category of human expressions can help provide information about the mood, personality, and intentions of the subject. This has applications in related problems such as fatigue driving detection, lie detection, and pain monitoring in patients [1].

Successful FER algorithms and their featurisers must be robust to both human variability (i.e. gender, race, posture) and environmental differences (i.e. background, lighting) in order to be effective with a wide range of subjects and situations. Popular datasets for this problem, including JAFFE [4], MMI [3] and Multi-PIE [2], were created in static and controlled laboratory environments with little orientational and environmental variation, and so the resulting trained models often fail to function in real world scenarios. The Static Facial Expressions in the Wild (SFEW) database has previously been used to evaluate the generality of traditional AI FER techniques in uncontrolled environments [1].

Applications of FER require algorithms which deliver fast and robust classifications in order to be applicable in real-world scenarios. Thus it is essential to investigate methods for increasing computational efficiency while maintaining accuracy. One such candidate is network reduction via the pruning of hidden neurons [6, 8]. Pruning approaches generally involve some quantification of the functionality of hidden neurons in order to remove redundant nodes, thus reducing the network. These approaches have been well explored in the research, and many such algorithms have been proposed [8]. One particular method involves a measure of distinctiveness in order to identify pairs of similar or complementary neurons, and then pruning those redundant neurons [6]. This method should theoretically preserve the performance of the network after pruning while improving efficiency.

Another potential avenue for exploration involves genetic algorithms (GAs). This class of algorithms aims to mimic the Darwinian theory of evolution to create a population and then selectively mate and mutate the fittest members. This has been used previously for network architecture optimization through feature selection [10] and selecting the best hidden layer size [9]. Both of these processes require generating a population of architectures and then individually training each one in order to compare their fitness. This becomes increasingly computationally expensive for large networks, and is thus unfeasible for complicated tasks such as FER. Instead another less computationally expensive task would

involve first training a network with an excess number of hidden neurons, and then using GA to prune the network down to its most compact form while maintaining accuracy [7, 8].

More research into computationally inexpensive architecture optimization is needed to establish the most useful methods for determining hidden layer sizes in neural networks. In this paper the network reduction techniques of distinctiveness pruning and GA pruning are studied and compared. The network used for testing is a simple three-layer neural network for classifying images from the SFEW database using the top 5 principal components of both the LPQ and PHOG descriptors. This dataset has been used in previous studies with a low classification accuracy of 19.0% [1], and so any insights that could be brought about by network reduction would help with further research into the task.

2 Method

2.1 SFEW database

The *Static Facial Expressions in the Wild (SFEW)* database [1] is derived from the *Acted Facial Expressions in the Wild (AFEW)* database [5], which consists of a compilation of videos extracted from various movies. Each short video features a dynamically acted expression of human behaviour in an environment which is very similar to real world scenarios, and every clip is classified based on the subject's facial expression. Static frames are selected from these videos to produce the 675 images in the SFEW database, each with an attached label of *angry, disgust, fear, happy, sad, surprise* or *neutral*. This covers a range of unconstrained facial expressions, postures, ages, resolutions and illuminations, and so this database provides a good basis for more robust FER models and allows for realistic evaluation of existing models [1].

Of the 675 images available, 75 are labelled with *disgust* and there are 100 images in each of the remaining classes. This makes the dataset unbalanced, and so the resulting trained model may be less effective in detecting disgusted facial expressions. Stratified splitting is used to ensure that the labels are evenly spread between the resulting training and test sets to reduce trained biases.

2.2 Preprocessing

Each image in the SFEW dataset is first cropped with a Viola-Jones face detector, and then both the local phase quantisation (LPQ) and pyramid of histogram of oriented gradients (PHOG) descriptors are calculated for each cropped face. These descriptors were chosen due to their invariance to blur and observed use in object recognition [1]. The parameters were empirically chosen to be pyramid level L = 4, bin length = 16, and angle range = [0-360] for PHOG, and block size = 4 × 4 for LPQ.

Principal component analysis is used to reduce the complexity of these features, selecting the top 5 principal components for both descriptors. These two sets of components are then combined to form a set of 10 input floats for each classified image. These features vectors are normalized to improve subsequent network performance. One image from the *fear* class did not have an associated PHOG feature vector, and so this pattern was removed from the dataset.

The remaining 674 patterns were divided into 10 sets for 10-fold cross validation. Stratified splitting was used to ensure that both the training and testing sets in each fold were balanced across the 7 classes. These are both standard techniques used for training and testing small datasets, as they help ensure an even spread of labels and make the most of all the data available. The results from cross validation are averaged to produce the statistics presented in this paper.

2.3 Neural network

A three-layer neural network was constructed to classify each pattern vector. The input and output layers consist of 10 and 7 neurons respectively, corresponding to the size of the input descriptor vector and the number of classes. The number of neurons in the hidden layer is varied between 20 and 1000 in this experiment, and a sigmoid activation function is used to restrict the output range to [0, 1] for use in the network reduction step. The output layer implements a softmax activation function to predict the final class, and this is common across many classification models. Cross entropy loss is used to train the network via an Adam optimizer with a batch size of 32. These hyperparameters are standard for this type of simple architecture across the literature.

2.4 Distinctiveness pruning

The Gedeon-Harris method [6] was used to identify redundant hidden neurons and thus prune the neural network at the end of training. This is done by first training the neural network and then computing the activation of each hidden neuron. The resulting activation vector contains the output of that hidden neuron for each pattern in the training set. These outputs are normalized from their original range of [0,1] to [-0.5,0.5] to allow for the computation of angles in the range of [0,180] degrees. The cosine similarity of each pair of vectors is then calculated to determine the distinctiveness of the outputs of the two neurons.

The threshold angle for similar and complementary neurons was initially set to 15 and 165 degrees respectively. Neurons with less than a 15 degree separation between their activation vectors are considered to have similar functionality, and so their weights and biases can be summed and one neuron can be removed. Neurons with more than a 165 degree separation are considered to have complementary functions within the network, and so both neurons are removed. The input weights and biases of these neurons are removed from the network, as are the output weights connecting to the output neurons.

This network reduction process is implemented at the end of model training, thus removing all similar and complementary neuron pairs in one sweep. The threshold angle is initially set to 15 degrees, but is then varied for experimentation and analysis purposes.

2.5 Genetic algorithm for network reduction

A genetic algorithm approach was used to prune the hidden layer of the trained neural network. The hidden layer was encoded into a chromosome with length N corresponding to the initial number of hidden neurons. Binary representation was used to indicate if the neuron was turned 'on' or 'off' for a particular individual network. This representation has been used successfully for the similar task of feature selection [10].

A randomly initialized population of 100 individuals was used. Each individual was then evaluated by first pruning the network according to the chromosome representation and then measuring the fitness of the reduced model. Different fitness functions are explored and tested in order to determine which would produce the most optimal hidden layer. In general these fitness functions were related to the accuracy of the model on the training set and have a penalty proportional to the final hidden layer size.

Tournament selection with a tournament size of 3 individuals was used to select the next population set, and a 5-member hall of fame was used to ensure that the best individuals remained in the population throughout subsequent generations. The selected individuals were mated using one point crossover with a probability of 80%, and there was a 20% probability of mutation with an independent probability of 10% for each bit to be flipped. This entire process was repeated for 50 generations, as this was found to be the point at which the maximum fitness of the population no longer changed. The best individual from the final population was extracted and used to generate the final pruned network.

3 Results and discussion

3.1 Baseline neural network

First the hyper-parameters for a traditional neural network were selected in order to optimize the overall accuracy of the trained model and subsequent tests. A sample network was run for 2000 training epochs while monitoring the overall accuracy in both the training and test sets. It was determined that 300 steps would be sufficient to train the network while avoiding the effects of overtraining, since this was generally the point at which the training and test accuracies started to significantly diverge. Learning rates of 0.01, 0.001, and 0.0001 were sampled along with a hidden layer of 10, 20, 50, 100, 200, 500, and 1000 neurons, and the final testing accuracy of each trial is shown in Table 1.

Table 1. Compilation of average final testing accuracies of neural networks after 300 training epochs with varied hidden layer sizes and Adam optimiser learning rates. The highest accuracy is bolded.

Learning rate	Number of hidden neurons						
	10	20	50	100	200	500	1000
0.01	21.7	**22.6**	21.2	21.3	21.8	20.7	18.8
0.001	19.1	19.9	20.5	21.7	21.4	21.2	21.6
0.0001	14.2	15.3	15.7	16.7	18.7	19.6	20.1

Of these combinations of learning rates and hidden neurons, the most optimal network was found to have 20 hidden neurons and a learning rate of 0.01, with a final accuracy of 22.6%. A support vector machine (SVM) classifier has been previously implemented on the SFEW database with LPQ and PHOG descriptors with a baseline testing accuracy of 19.0% after 2-fold cross validation [1]. It is worth noting that the database used in [1] had an additional 25 samples in the *disgust* class and a Strictly Person Independent (SPI) protocol, so a true comparison of these results cannot be made. However, this still shows that a simple neural network can perform just as well as a SVM on this task.

3.2 Distinctiveness pruning

Pruning hidden neurons with similar or complementary functions should reduce the network and thus improve efficiency while maintaining accuracy [6]. To determine the extent of this improvement and its effect on classification accuracy, the number of hidden neurons and overall accuracy before and after pruning was plotted for a variety of networks. These models were all trained with a learning rate of 0.01 and 300 training epochs, as determined in the previous section.

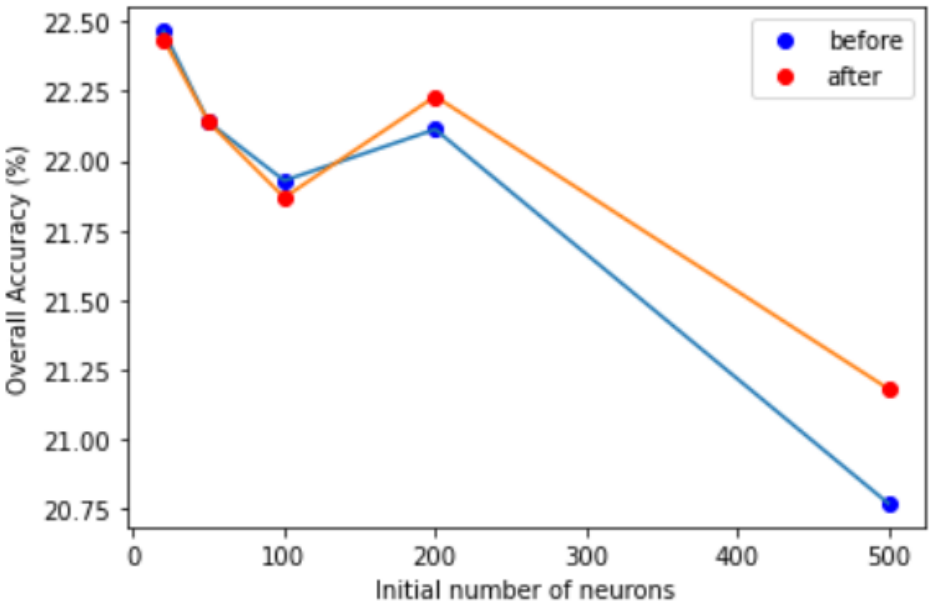

Fig. 1. Overall testing accuracy before and after distinctiveness pruning as a function of the initial size of the hidden layer.

Table 3. Number of hidden neurons before and after distinctiveness pruning.

Initial hidden size	20	50	100	200	500
Final hidden size	19.98	49.96	99.66	198.56	467.68

In the last section it was shown that a hidden layer size of 20 neurons gave the most optimal architecture for this problem. It would then be expected that any pruning that occurs on a larger network would converge to a network of roughly that same size, however this is not the trend observed in Table 3. Instead the final size of the hidden layer remains fairly proportional to the initial size, and the extent of pruning grows with the initial size of the network. In fact, pruning has almost no effect on the size of networks with less than 100 hidden neurons. This indicates that distinctiveness pruning with a threshold angle of 15 degrees as previously proposed [6] is not an effective tool for significant network reduction.

The average testing accuracy showed little difference before and after pruning and showed little dependence on the initial size of the hidden layer. This shows that no functionality is lost during pruning, and as expected there is also no significant gain in accuracy. The difference in accuracy seems to be random, and most likely depends on the total effect of all the removed neuron pairs.

One factor that could be influencing the extent of network reduction is the distinctiveness threshold angle. This angle controls which neurons are deemed to be similar or complementary within the reduction process. The dependence of the final testing accuracy and the proportion of pruned neurons on this distinctiveness threshold was investigated with a neural network. A hidden layer of size 100 was deliberately chosen to visualize the effects of pruning, along with the standard learning rate of 0.01 and 300 training epochs. The threshold angle was varied between [0-90] degrees in 5 degree intervals.

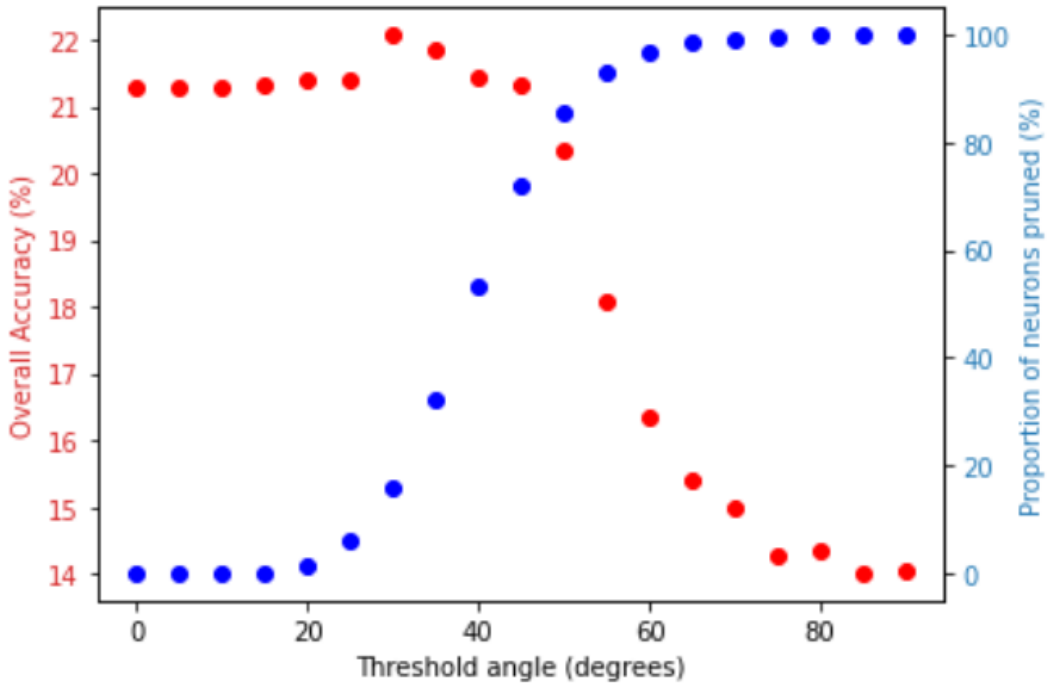

Fig. 1. Network final testing accuracy (red) and hidden layer size sensitivity (blue) as a function of the distinctiveness threshold angle.

As expected, training accuracy suffers as more hidden neurons are removed, despite earlier showing that 20 units resulted in the most accurate network for this task. This is likely due to the large threshold angle required to remove such a significant proportion of the hidden layer. These neurons are likely to have a more significant function in the network, and so their removal causes drastic changes in the subsequent testing accuracy.

It is interesting to note that the overall testing accuracy only begins to decrease after a threshold angle of 45 degrees. At the recommended angle of 15 degrees there is no discernable effect on network size or accuracy, while at 45 degrees the network is reduced by 72% with no significant impact on accuracy. This suggests that the threshold could be increased from 15 to 45 degrees in order to further improve computational efficiency while maintaining accuracy. This also calls into question the validity of the distinctiveness measure itself, since it is expected that two neurons which had activation vectors 45 degrees apart from each other would be significantly different enough to properly disrupt the network when pruned. This indicates that shared functionality between hidden neurons may need a more complex measure than the pairwise distinctiveness calculated here.

3.3 Network reduction via genetic algorithm

Genetic algorithms can also be used to optimize the final hidden architecture of a trained neural network. Two different functions were used to measure the fitness of each generated architecture:

$$Individual\ fitness = training\ accuracy\ \% - \lambda \frac{N_f}{N_i} \tag{5}$$

$$Individual\ fitness = training\ accuracy\ \% - \lambda N_f \tag{6}$$

where N_i and N_f are the initial and final hidden layer sizes respectively, and λ is the penalty coefficient. The hidden size penalty is introduced to avoid overtraining of the model on the training set, while also incentivizing network reduction [10].

These two fitness functions were implemented with varying λ values for the GA pruning of networks with a range of hidden layer sizes. It was found that the variation in λ had no effect on the final hidden layer size of the networks pruned with equation 6, and instead all networks were consistently reduced by about 55% for networks with $N_i > 20$ hidden neurons. This shows that a penalty of λN_f is not sufficient to incentivize network pruning. Conversely, for GA pruning with equation 5 as the fitness function, increasing λ was found to significantly increase the proportion of pruned neurons. This may be because the hidden size penalty is made proportional to the initial hidden size, and so it is able to have a similar effect across different initial architectures. For this reason equation 5 is used as the fitness function for further analysis of the GA pruning process.

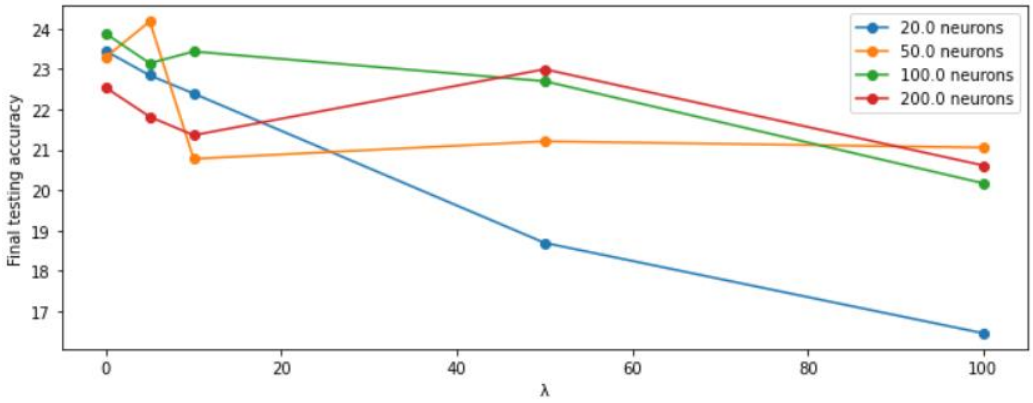

Fig. 3. Network final testing accuracy for four different initial hidden layer sizes, pruned using equation 5 as the fitness function with varying λ values.

There is a weak negative correlation between λ and the classification accuracy of the pruned network on the testing set as shown in figure 3. The extent of this correlation shows that λ has little effect on the generalizability of the pruned network. As expected by the construction of the fitness function, there is a strong correlation between λ and the classification accuracy on the training set. A higher value of λ corresponds to a larger penalty on the network size, and so the individual with the best fitness has both decent accuracy on the training set, as well as significant network reduction. The best accuracy on both sets of data is achieved by setting λ=0, effectively eliminating the hidden layer size penalty and making optimizing the training accuracy the priority of the algorithm.

These results show that GA pruning can either significantly reduce network size by using a high λ value, or improve classification accuracy with a low λ. Thus the λ value must be tuned to the purpose of the network pruning algorithm. For this specific task the aim is to reduce the hidden layer size while also maintaining accuracy, and so a λ value of 50 is most appropriate to balance accuracy and efficiency.

3.4 Comparison of techniques

For comparison of the two network reduction techniques detailed in this report, neural networks with a variety of hidden layer sizes were trained and then pruned with four pruning functions: distinctiveness pruning with threshold angles of 15 and 45 degrees, and GA pruning with λ values of 0 and 50.

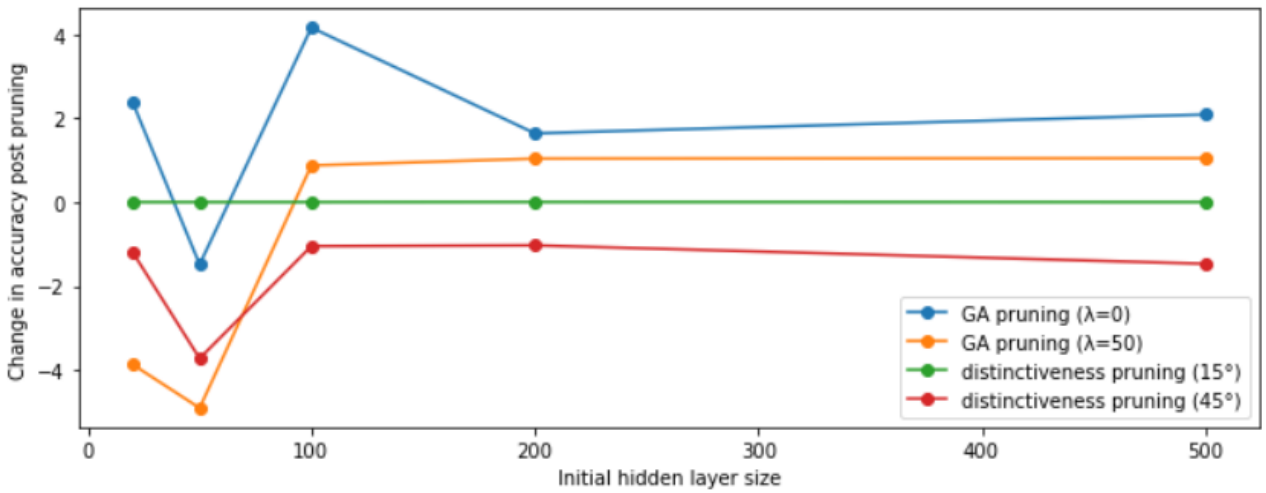

Fig. 4. Network final testing accuracy for the four pruning techniques compared to the initial unpruned network, implemented over a range of initial hidden layer sizes.

Figure 4 shows the impact each pruning technique had on the accuracy of a fully trained neural network. As expected, both GA pruning techniques were able to slightly enhance the accuracy of the reduced network, and the algorithm with $\lambda = 0$ showed the biggest improvement. Distinctiveness pruning with a threshold angle of 15 degrees showed no significant effect due to minimal network reduction, while increasing the threshold angle to 40 degrees showed a slight decrease in accuracy. The differences between the results of these two algorithms can be rationalized by their mechanisms: distinctiveness pruning makes no attempt to improve accuracy and simply attempts to remove redundant pairs of neurons, while GA pruning takes both accuracy and network reduction into account in its fitness function.

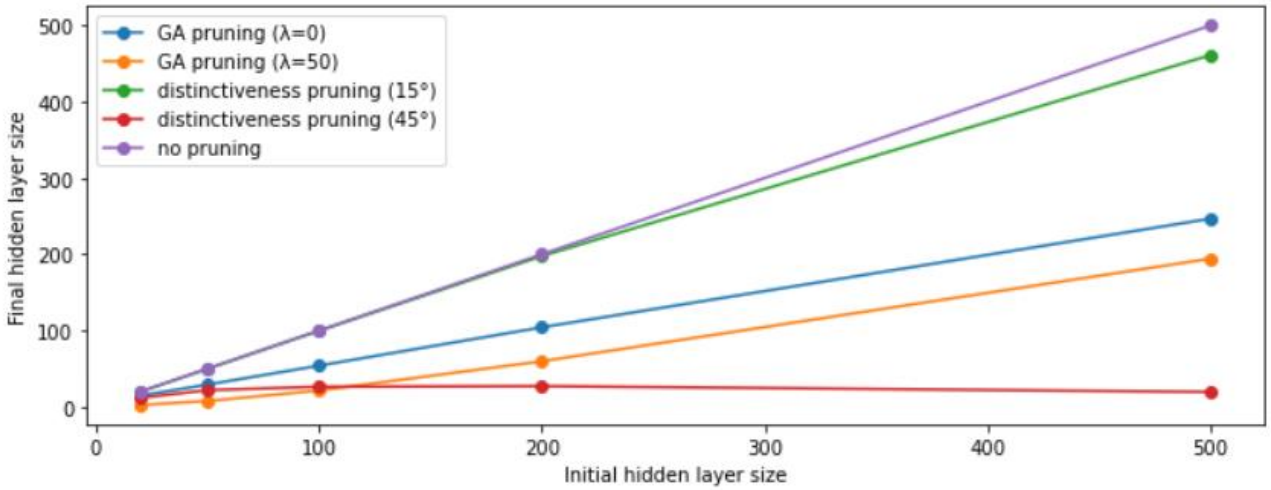

Fig. 5. Final hidden layer sizes for the trained neural network and the four implemented pruning techniques on a range of initial hidden layer sizes.

Comparison of the extent of network reduction is also needed to evaluate these pruning techniques. Distinctivness pruning with a threshold angle of 15 degrees again shows very minimal effect, and only begins to significantly prune the network when the hidden layer initially has 500 neurons. Conversely, with a threshold angle of 45 degrees the algorithm consistently reduced the network down to between 20 and 30 hidden neurons regardless of the initial network size. This is likely to be the optimal size of the network for this task, since this is when all hidden neurons are sufficiently distinct from each other. Both GA pruning techniques tested show similar degrees of reduction, with the technique with $\lambda = 50$ pruning 50 more neurons than that with $\lambda = 0$ for initial hidden layer sizes of 100, 200 and 500 neurons. This is expected due to the incentive the additional hidden layer penalty provides to further reduce the network.

These results show that distinctiveness pruning with a threshold angle of 45 degrees is able to prune a large network down to its most optimal size with a 1% reduction in absolute accuracy. GA pruning with $\lambda = 50$ provides a 1% improvement in absolute accuracy while significantly reducing the network, although the size of the final network remains proportional to that of the initial. GA pruning with $\lambda = 0$ reduces the network to a lesser extent but improves

absolute accuracy by 2%, making it suitable for post-training network optimization. This algorithm also produces the network with the best classification accuracy in this paper – the final network has an accuracy of 25.2% with 54 hidden neurons, pruned down from an initial size of 100 neurons.

3.5 Limitations and future work

The results shown here show that the distinctiveness pruning technique as presented in [6] has little effect on network reduction. Instead increasing the threshold angle to 45 degrees effectively reduces the network down to its optimal size with only a slight decrease in accuracy. This raises questions about the legitimacy of the distinctiveness measure used. Neuron functionality is likely to be more complex than the pairwise approximation used here. Future studies should investigate methods of evaluating groups of neurons to determine if the total group contributions are similar or complementary.

The fitness function used to evaluate the population is one of the most important aspects of GAs. This paper only tested two common fitness functions, and while the results here are promising, more experimentation is needed. The ideal fitness function should reduce the network down to the same size regardless of the initial architecture. The results here also suggest that GA pruning could be used as for post-training optimization to improve both efficiency and accuracy. More research with other datasets and tasks is needed to determine the usefulness of this method.

Both pruning techniques investigated in this paper fall into the one-shot class of algorithms and are implemented without any post-pruning training. Extra training time after pruning has been shown to improve network accuracy by allowing the network to adjust itself [12]. It would be beneficial to use either fine-tuning or weight rewinding on the pruned networks shown here and compare the results to those proposed in this paper. Additionally, distinctiveness pruning may be improved by letting the process unfold iteratively rather than just at the end of training. The algorithm could prune the most similar and complementary pairs of neurons after a set number of iterations, then retrain and repeat this whole process until the network is sufficiently compressed. This method has been implemented elsewhere with promising results [12].

The secondary goal of this study was to improve the accuracy of FER models trained on the SFEW dataset. While the classification accuracy of the network here is too low to be a reliable resource, it still outperforms that of the previously studied SVM classifier with an additional 6.2%. These results show that neural networks can perform as well as SVM classifiers with these descriptors. although both methods require a more rigorous implementation scheme in order to withstand the rough conditions of the SFEW database and maintain reliable classification accuracy.

Improvements in classification accuracy could be achieved in two main ways: by increasing the size of the dataset, or by implementing a more rigorous deep learning network. Training may be impeded by the relatively small size of the SFEW database. The database could be expanded by performing image manipulation to generate additional samples, and this would have the added benefit of improving the generalizability of the trained model. Other more extensive facial expression databases could also be used, like the 2019 Aff-Wild2 database [11].

Another limitation of the database used here is the lack of image preprocessing. The images need to be properly preprocessed in order to focus the model's attention on the subject's face while ignoring the rest of the image. This could be done by using a single Single Stage Headless (SSH) detector to crop the image and then extracting facial landmarks [11]. The latter process would also aid in making the model invariant to facial orientations. Convolutional neural networks (CNNs) have shown to be successful in a range of facial recognition tasks, and thus may also be useful for FER.

4 Conclusion

A three-layer neural network was implemented to classify facial expressions from the SFEW database, with 10 image features extracted using LPQ and PHOG. The resulting network was found to show comparable accuracy to a previous baseline SVM model [1], although both models are not sufficiently detailed to be effective in classifying expressions in rough conditions. Further research is needed to improve the generalizability of these FER algorithms for future applications.

Two pruning methods are proposed and implemented for this classification task: distinctiveness pruning and genetic algorithm (GA) pruning. Both techniques aim to reduce network size while maintaining accuracy. The effect of the threshold angle in the distinctiveness pruning algorithm was investigated. It was found that 15 degrees was not sufficient to significantly reduce the trained network, but 45 degrees consistently reduced the network down to its optimal size with a 1% diminishment in accuracy. Two different fitness functions for GA pruning were investigated,

along with the effect of the penalty coefficient λ. It was found that the penalty was most effective when it was made proportional to the initial hidden layer size, and the extent of network reduction increased with a larger λ. Comparison of multiple pruning techniques across differently sized initial architectures showed that distinctiveness pruning with a threshold angle of 45 degrees was the most efficient method for network compression without significant loss in accuracy, while GA pruning with λ = 0 showed promise as a post-training network optimization tool. These methods could both benefit from additional research into post-pruning fine-tuning and weight rewinding.

References

1. A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Static Facial Expression Analysis in Tough Conditions: Data, Evaluation Protocol and Benchmark. In *1st IEEE International Workshop on Benchmarking Facial Image Analysis Technologies BeFIT, ICCV2011*., 2011.
2. R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-PIE. In *Proceedings of the Eighth IEEE International Conference on Automatic Face and Gesture Recognition,* FG'2008, pages 1–8, 2008.
3. M. Pantic, M. F. Valstar, R. Rademaker, and L. Maat. Webbased database for facial expression analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, ICME'05, pages 317–321, 2005.
4. M. J. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. In *Proceedings of the IEEE International Conference on Automatic Face Gesture Recognition and Workshops*, FG'98, 1998.
5. A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Acted Facial Expressions in the Wild Database. In *Technical Report*, 2011.
6. T. Gedeon, D. Harris. Network Reduction Techniques. In *Proc. Int. Conf. on Neural Network Methodologies and Applications, AMSE, vol. 1, pp. 119-126, San Diego, 1991a. Proc. 9th Ann. Cog. Sci Soc. Conf. , Seattle*, 1987.
7. D. Whitley, T.Starkweather and C. Bogart. Genetic algorithms and neural networks: optimizing connections and connectivity. In *Parallel Computing 14*, pages 347-361, 1990.
8. M. Gethsiyal Augasta, T. Kathirvalavakumar. Pruning algorithms of neural networks – a comparative study. In *Cent. Eur. J. Comp. Sci*, 3(3), pages 105-115, 2003.
9. P. G. Benardos, G.-C. Vosniakos, Optimizing feedforward artificial neural network architectures. In *Eng. App. Artif. Intelligence*, 20, pages 365-382, 2007.
10. S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithms: past, present, and future. In *Multimedia Tools and Applications*, 80, pages 8091-8126, 2021.
11. D. Kollias, S. Zafeiriou. Expression, Affect, Action Unit Recognition: Aff-Wild2, Multi-Task Learning and ArcFace. In *British Machine Vision Conference*, 2019.
12. A. Renda, J. Frankle, and M. Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations.* 2020.