Anger Classification by Feed Forward Neural Network and A Class Threshold Manipulation Technique

Stavros Dimos
U5825599@anu.edu.au

[1] Research School of Computer Science, Australian National University, Canberra, Australia

Abstract. This paper replicates a simple technique for manipulating class thresholds in neural networks. A three-layer feed forward neural network was generated, with a genetic algorithm for hyperparameter selection, and trained on a dataset of pupillary responses for the purpose of anger classification. The threshold manipulation technique was then used to maximize accuracy for individual classes and assess the effect this had on accuracy for the model. Significant increases in individual class accuracy were achieved with little to no loss of accuracy for this model over the entire dataset.

Keywords: Affective Processing, Neural Network, Classification, Genetic Algorithm, Hyperparameter Selection

1 Introduction

The dataset used in this paper is a summarized form of the dataset used by Chen, Gedeon, Hossain and Caldwell [1]. This dataset was used to model anger in observed individuals, as either genuine or posed, based only on pupillary information acquired from a number of observers. In that paper, their model performed at 95% accuracy compared to the 60% accuracy of verbal predictions by those same observers, showing that machine learning techniques could achieve considerably higher accuracy than the observers could, using their own physiological signals.

This paper aims to use the technique outlined by Milne, Gedeon and Skidmore [2], in their paper classifying dry sclerophyll. Specifically, manipulating the class split threshold of a neural net classifier such that rates of false positive and false negative classifications can be shifted and balanced. The purpose of this is twofold. Firstly, to replicate the results of Milne, Gedeon and Skidmore [2], by finding the boundaries where false positive and false negative classifications may be minimized. Secondly, to determine whether this technique may be utilized to improve the model accuracy, either overall or within individual classes. In the ideal case, the output of the model indicates a probability or confidence in the classification, meaning that increasing the threshold for the positive class leads to increased accuracy over that class.

As this is a technique for making better use of simple neural networks on more complex data, the network cannot have perfect or near perfect accuracy for this technique to produce meaningful results. It would defeat the purpose of using the technique in the first place.

As such, the form of the data being used is particularly useful for this paper because, as a summary, it is strictly less informative than the original set would be. A simple feed forward neural network with few processing layers was unable to achieve the 95% accuracy reported by Chen, Gedeon, Hossain and Caldwell [1]. Rather, the model used for this paper reported 83.3% accuracy, the extreme high end of the 27.8%-83.3% accuracy range reported by Qin, Gedeon, Chen, Zhu and Hossain [3].

2 Method

2.1 Data Preprocessing

The data was preprocessed before it was acquired. The original time series data was transformed into 6 summary inputs and two target outputs, video and label. The summary dataset was normalized within the range 0-1 to improve the rate at which the model converges. Additionally, the video target column was removed as we are only concerned with predicting the label.

2.2 Neural Network Model

A variety of neural network topologies were tested via genetic algorithm, as will be discussed in the next section, to find the best performing model. The one used for threshold manipulation in this paper is a fully connected, feed forward network. The network has 12 hidden units, split 10 and 2 into 2 processing layers

All networks tested used a sigmoid activation function, noting that we are not concerned with the sigmoid vanishing gradients as there are at maximum two processing layers. The network utilizes binary cross entropy for the loss. This is the standard choice for classification tasks such as this. It also employs a stochastic gradient descent optimizer with a learning rate of 0.05. All of these are implemented in Pytorch [4].

The dataset was split and randomized through Scikitlearn's train_test_split function [5] into a 70% training set and 30% holdout set for testing. In the case of the best performing model, training was done over 300,000 epochs of the training data, though minimum loss on the final model was achieved at 267,000 epochs.

2.3 Selection of Hyperparameters - Genetic Algorithm

The inputs, output, learning rate and maximum number of epochs for the neural network were hand-picked at 6, 1, 0.05 and 300,000 respectively. The three remaining features, selected by genetic algorithm were:

1. Number of hidden units in the first hidden layer
2. Number of hidden units in the second hidden layer
3. Whether the network actually uses both hidden layers or just the first

The genetic algorithm used for this experiment is quite standard. There are 5 members in the initial population as well as all succeeding populations. The simulation runs for 5 generations, producing a total of 25 candidate solutions, narrowed down to 5 (the best of each generation) and further narrowed down to a single solution, representing the best hyperparameter setup. Ideally, the use of this algorithm would grant a decrease in training time for the model, as was reported by Han, Choi, Park and Hong [6] in their paper discussing the impact of genetic algorithms for hyperparameter choice on the verification time of neural networks.

The first generation was produced randomly. Each subsequent generation was produced via single point crossover of individuals from the previous generation, subject to a mutation rate of 0.05 per bit. Parents were selected by weighted probability of relative fitness, ie. Fittest individuals are more likely to be selected.

The features outlined earlier were encoded in each member of the population, represented by 8 bits:

4 bits - 1 to 16 hidden units - 1st layer	3 bits – 1 to 8 hidden units – 2nd layer	1 bit – Boolean

As bits 1-4 and 5-7 both represent a number, there were several options for the encoding. The major consideration here was the effect of crossover on the number. For example, if we were to switch just the leftmost bit of a 4-bit binary encoding, the number could more than double or reduce by half despite, for example, 1111 and 0111 sharing the majority of their bit values. Gray code was eventually chosen because of it's adjacency property, ie. Numbers in close proximity, eg 5 and 6, are always only a single bit shift away from each other. The effect of this is that similar members of the population will often produce children close to their own representation.

2.3 Evaluation/Selection of Model – Fitness Function

Each network was trained on the training set until test set loss was minimized, in order to find the best possible model for each topology without overfitting the training data. These models were evaluated against each other by comparison of both test set loss, as per binary cross entropy, and test set accuracy. In a balanced dataset such as this, where both target classes are well represented, accuracy over the entire set is likely to be a good indicator of model performance because the test set would have similar proportions of both classes in the typical case.

Based on the above, the fitness of any individual in the population is defined as their accuracy on the test set, this is the metric by which the final model is judged. Similarly, the relative probability of selection as a parent of the next generation is defined as:

$$P(\text{SelectedAsParent}) = \text{Accuracy}/(\text{SummedAccuracyOfGeneration})$$

For example, consider a generation of just two individuals, A and B, with accuracy 100% and 50% respectively. A would have a 66.6% chance and B would have a 33.3% chance of being selected as a parent for each selection cycle.

2.4 Threshold Manipulation of Model

The chosen neural network was used to predict both the training and test sets at varying thresholds of class distinction. The threshold between class 0 and class 1 (posed and genuine) was varied between 0.3 and 0.7 in discrete increments of 0.05. With class 1 as the positive class and class 0 as the negative class, the true positive, false positive, true negative and false negative rates of classification, as well as accuracy, were recorded for each threshold. The effectiveness of threshold manipulation for improving the model with respect to each class was simply measured as the accuracy of predictions of that class. For example the accuracy of a model with respect to class 1 would be:

$$Acc(PosClass) = TruePos/(TruePos+FalsePos)$$

If, for example, positive class accuracy is increased when the threshold is increased, that would suggest that the output of the network might be a confidence value, especially if the same was true for the negative class in the opposite direction.

3 Results

3.1 Neural Network

The neural network selected for this experiment via genetic algorithm had an accuracy of 88.6% on the training set and 83.3% on the test set. Loss on the test set was minimized after 267,000 epochs of the training data. This network was produced in generation 2 of 5 and had an encoding of 1111-011-1. This means the network had two processing layers, with 10 units in the first and 2 in the second.

Fig. 1. Test set loss over time. The model at 267,000 epochs, with lowest loss, was used for this paper.

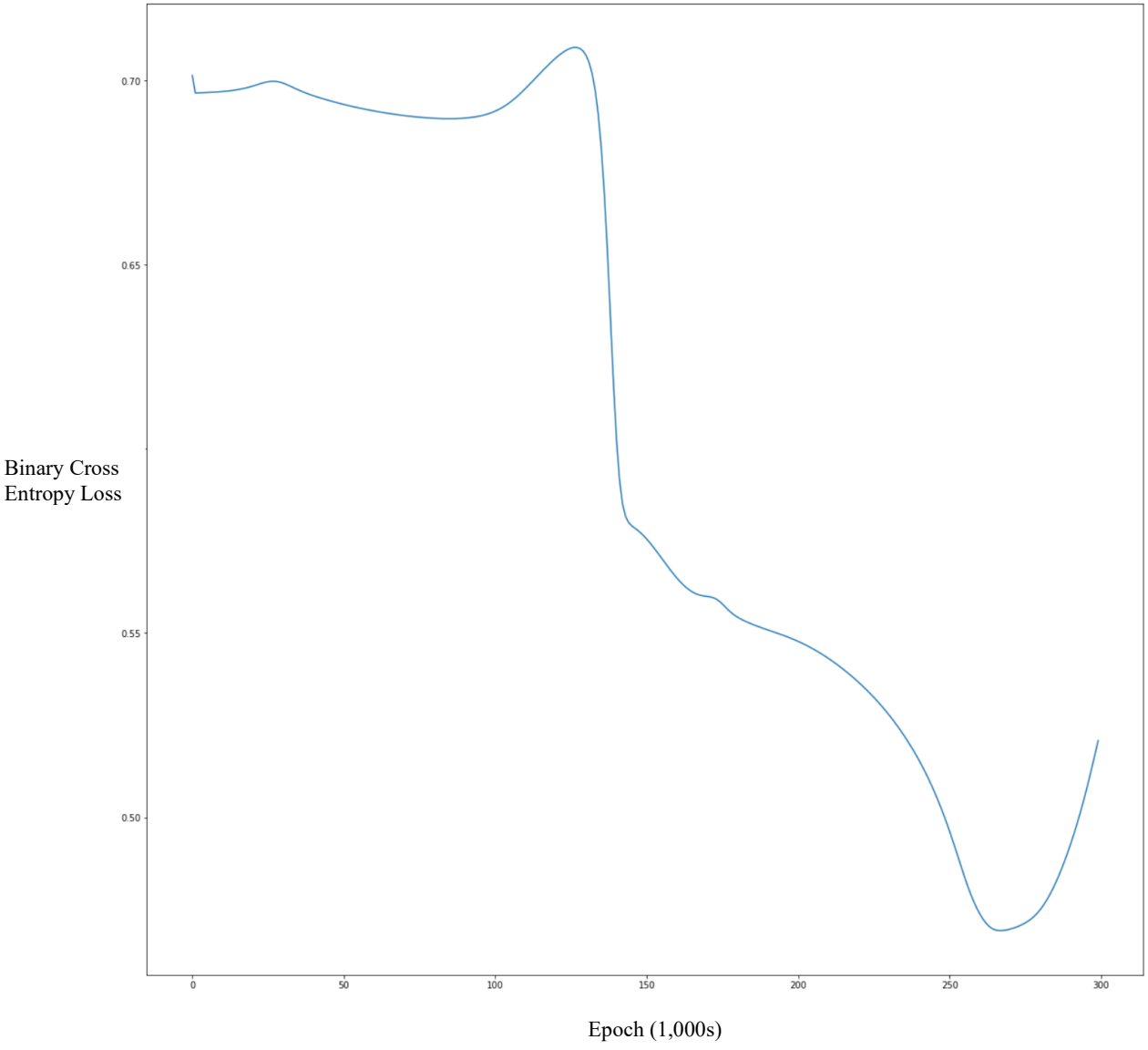
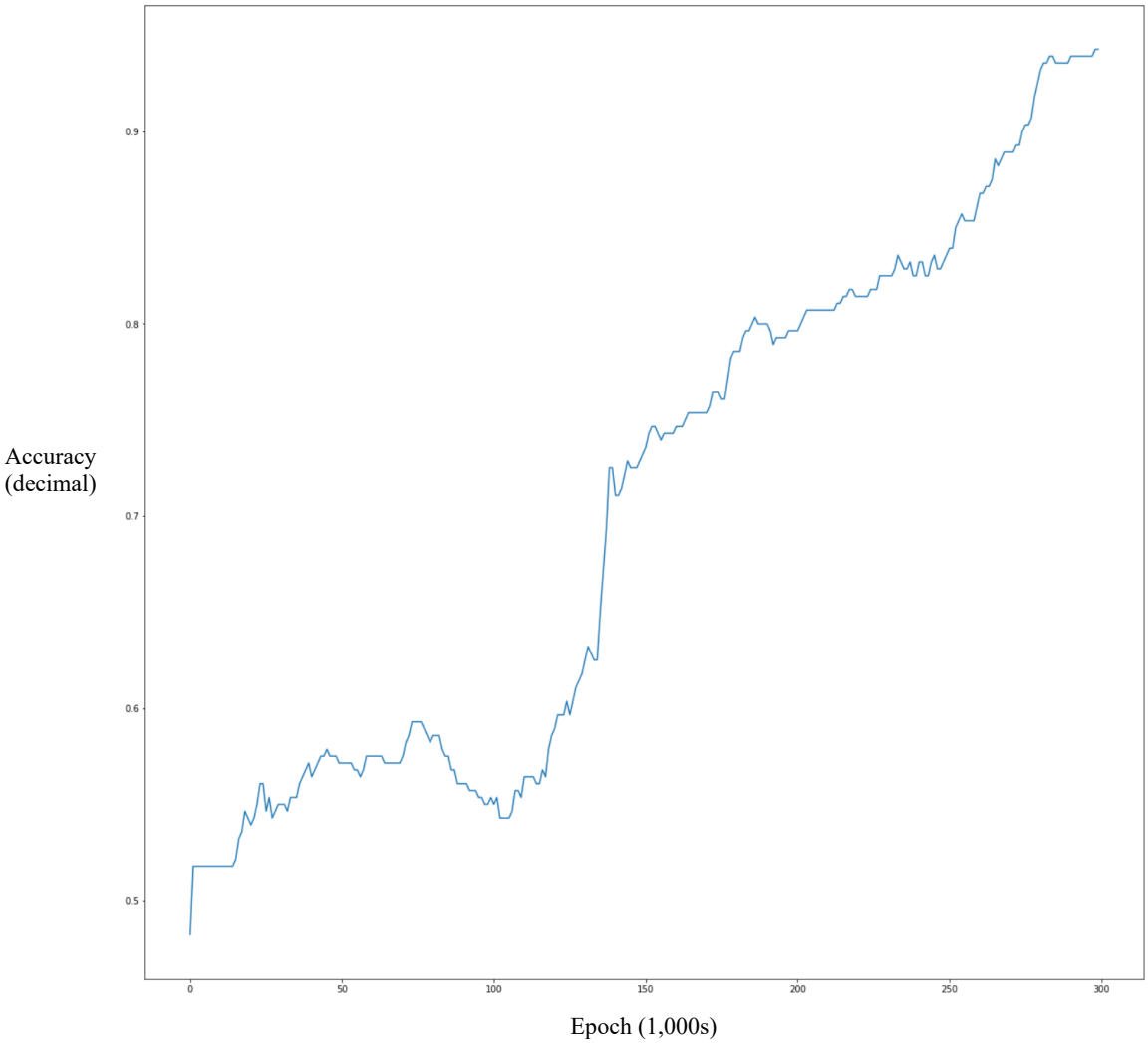


Fig. 2. Test set accuracy over time.



3.2 Threshold Manipulation

Table 1. Results of threshold manipulation on the selected model over the training set. Positive = genuine anger

Threshold	TruePositive	FalsePositive	TrueNegative	FalseNegative
0.3	145	28	107	0
0.35	143	26	109	2
0.4	143	22	113	2
0.45	136	18	117	9
0.5	131	18	117	14
0.55	125	17	118	20
0.6	123	14	121	22
0.65	118	14	121	27
0.7	114	9	126	31

Table 2. Results of threshold manipulation on the selected model over the test set.

Threshold	TruePositive	FalsePositive	TrueNegative	FalseNegative
0.3	51	19	46	4
0.35	51	17	48	4
0.4	50	15	50	5
0.45	49	14	51	6
0.5	49	14	51	6
0.55	47	11	54	8
0.6	44	10	55	11
0.65	43	9	56	12
0.7	41	8	57	14

Table 3. Accuracy of the model for each class at each threshold for the training set

Threshold	Positive Class Accuracy	Negative Class Accuracy	Accuracy Over Entire Set
0.3	83.82%	100.00%	90%
0.35	84.62%	98.20%	90%
0.4	86.67%	98.26%	91%
0.45	88.31%	92.86%	90%
0.5	87.92%	89.31%	89%
0.55	88.03%	85.51%	87%
0.6	89.78%	84.62%	87%
0.65	89.39%	81.76%	85%
0.7	92.68%	80.25%	86%

Table 4. Accuracy of the model for each class at each threshold for the test set

Threshold	Positive Class Accuracy	Negative Class Accuracy	Accuracy Over Entire Set
0.3	72.86%	92.00%	81%
0.35	75.00%	92.31%	83%
0.4	76.92%	90.91%	83%
0.45	77.78%	89.47%	83%
0.5	77.78%	89.47%	83%
0.55	81.03%	87.10%	84%
0.6	81.48%	83.33%	83%
0.65	82.69%	82.35%	83%
0.7	83.67%	80.28%	82%

Fig. 3. Graphical depiction of table 3. Training set class accuracy plotted against the thresholds used

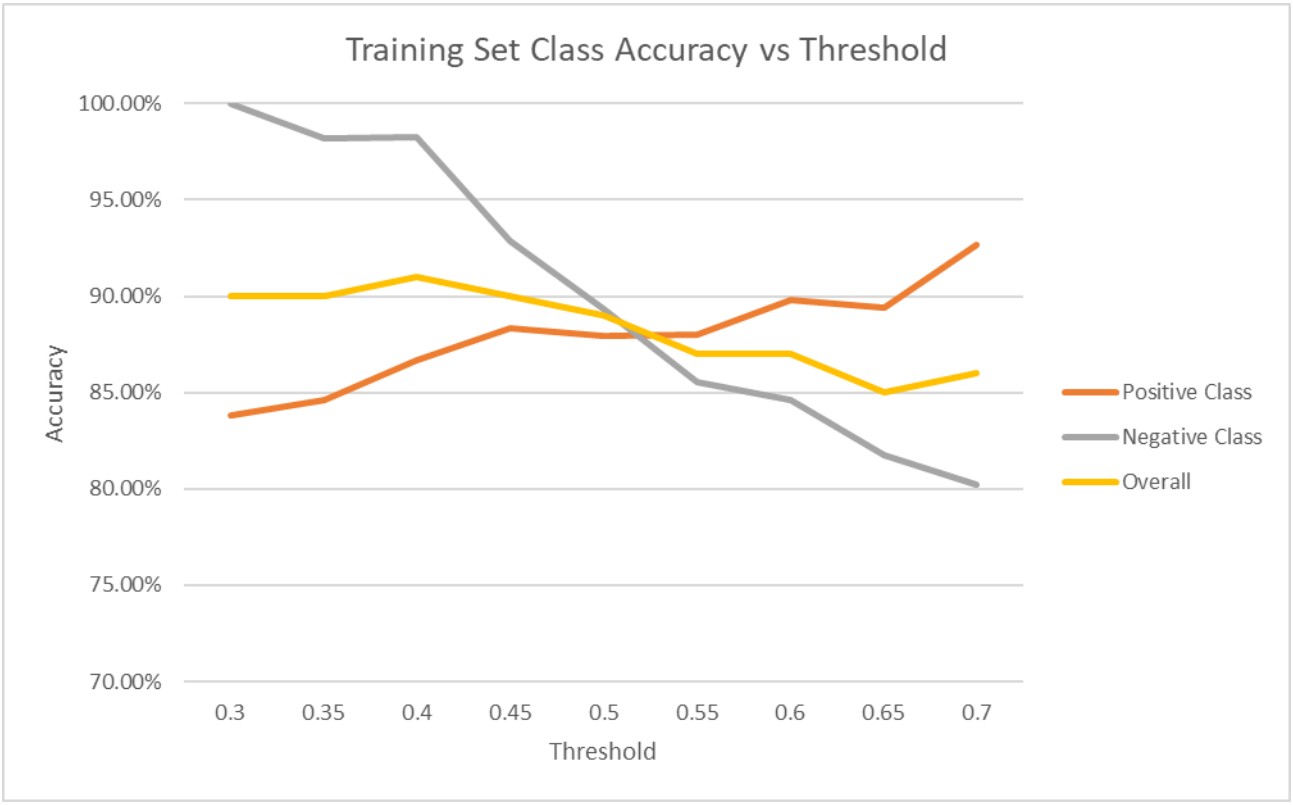
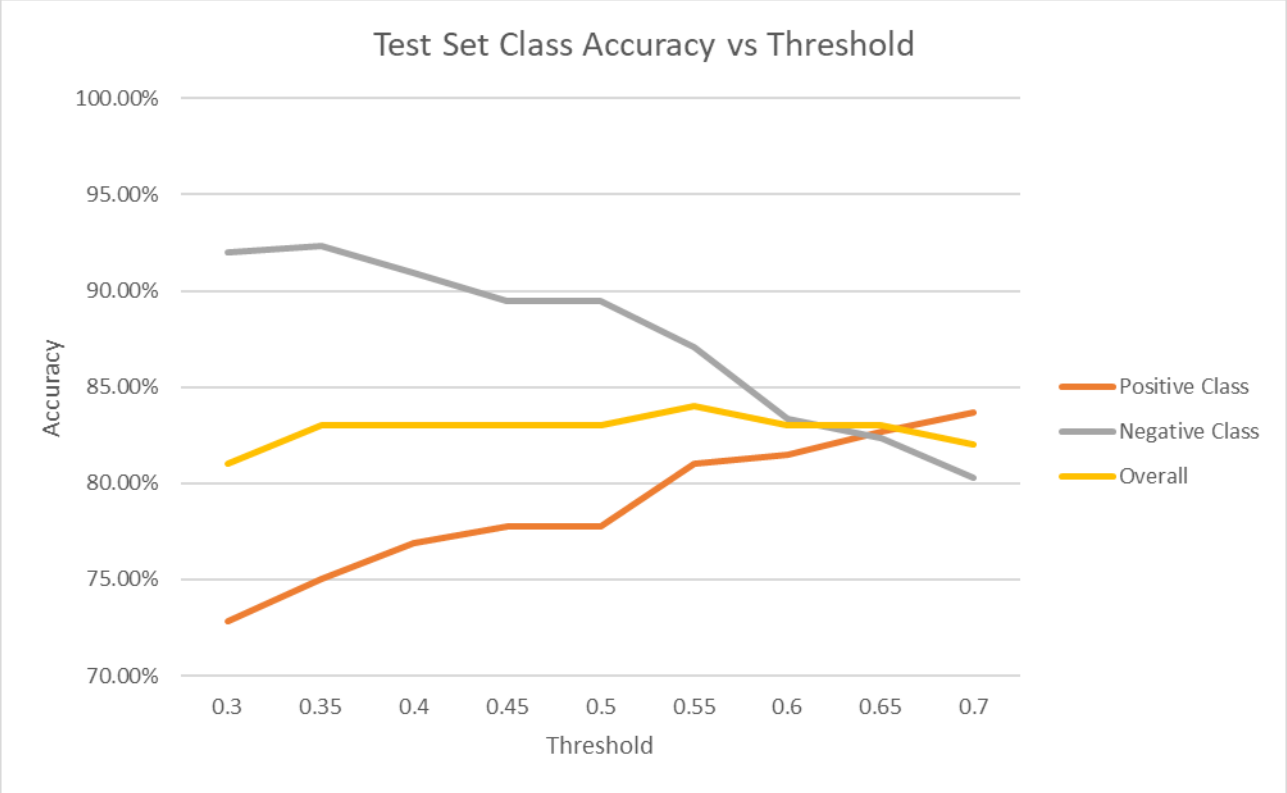


Fig. 4. Graphical depiction of table 4. Test set class accuracy plotted against the thresholds used



4 Discussion

The results show that increasing the threshold decreases false positive classifications and, similarly, decreasing the threshold decreases false negative classifications. This is in line with the results of Milne, Gedeon and Skidmore [2] except that false positive results for this model could not be reduced to 0 for a threshold of 0.7 and, similarly, false negative results could not be reduced to 0 for a threshold of 0.3. Notably, false negatives were reduced to 0 on the training set at a threshold of 0.7. These results on the test set indicate that the overlap of positive and negative classification regions for this set is greater than that of the dry sclerophyll classification dataset.

Increasing threshold also increased accuracy for the model over both the training and test sets for the positive class. Similarly, decreasing the threshold increased the accuracy of the model for negative class predictions, again for both training and test sets. The positive class saw an 8.86 percentage point increase for the training set and a 10.81 percentage point increase for the test set. The negative class saw a 19.75 percentage point increase over the training set and 11.72 percentage point increase over the test set. In summary there was a positive correlation between threshold and positive class accuracy and inverse correlation between threshold and negative class accuracy.

Notably, Milne, Gedeon and Skidmore [2], saw a decrease in model accuracy of roughly 30 percentage points when altering the classification threshold. In stark contrast to their results, the model used for this experiment saw little to no decrease in overall accuracy, i.e. for the combined training and test sets, given the significant increase in accuracy for single classes. When the threshold was decreased from 0.5 to 0.3, the training set actually saw an increase of 1.0 percentage points in overall accuracy and the test set saw a decrease of 2.0 percentage points. When the threshold was increased from 0.5 to 0.7, the training set saw a decrease of 4.0 percentage points in overall accuracy and the test set saw a decrease of 1.0 percentage points. Based on these results, there would seem to be no significant change in overall accuracy with changes in threshold, at least within the 0.3-0.7 range. This could be for any number of reasons, though one notable difference between this dataset and the dry sclerophyll dataset is the number of classes, possibly impacting accuracy when looking at the entire set with this method.

This suggests that either most predictions are placed at the extremes of the 0-1 range or that predictions in the 0.3-0.7 range have roughly 50% accuracy and both classes are evenly distributed across the range, thus having only a small effect on overall accuracy. As an explanation for this, imagine that all the predictions in the 0.3-0.7 range swapped from class 0 to class 1 at once. If half of them were wrong before, then the other half are now wrong when the threshold is moved. If correct and incorrect predictions are placed closely and to a roughly 1-1 ratio, this would mean that even small movements of threshold would appear to have little effect on overall accuracy, as was seen in the results.

5 Conclusion and Further Work

This paper demonstrated that threshold manipulation can be used to improve the accuracy of a simple neural network with respect to single classes. This particular outcome is in line with the results on the Dry Sclerophyll Forest Dataset [2]. Significantly, this paper was able to achieve increased single class accuracy, upwards of 10.0 percentage points in some cases, with little to no degradation, roughly 1.0-4.0 percentage points, of accuracy for the entire set.

As noted in the discussion, the number of classes in the dry Sclerophyll Forest dataset is greater than that of the anger dataset used for this paper. In order to verify that this technique for manipulating false positive and negative rates is robust, and that overall accuracy can be maintained, it would be best to repeat this experiment with an expanded class range. As pupillary data is indicative of affective processing for a variety of emotions [7], the most logical additions to the classes would be other sets of emotion-based data, for example genuine and posed happiness, fear, etc.

Regarding the neural network used for this paper. Training time took 300,000 epochs, even on a normalized training set, for each individual in the genetic algorithm. This is significantly lower than the 3,000,000 epochs for the raw data but is still a significant bottleneck. Considering that this genetic algorithm runs 5 generations of 5 individuals per generation, ie. 7,500,000 epochs of the training set over the entire period, it is difficult to justify the inefficiency. As such, this paper would recommend the use of a learning rate scheduler in order to optimize the training times of each individual.

Overall the results were promising and indicate that threshold manipulation might be useful in specific problems. For example, where one would need to maximize certainty in outcomes regarding a single class while maintaining the integrity of the model with regards to accuracy over the entire set.

References

1. Chen, L., Gedeon, T., Hossain, M. Z., & Caldwell, S. (2017, November). Are you really angry?: detecting emotion veracity as a proposed tool for interaction. In Proceedings of the 29th Australian Conference on Computer-Human Interaction (pp. 412-416). ACM.
2. Milne, L.K., Gedeon, T.D., Skidmore, A.K. Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood.
3. Qin Z., Gedeon T., Chen L., Zhu X., Hossain M.Z. (2018) Artificial Neural Networks Can Distinguish Genuine and Acted Anger by Synthesizing Pupillary Dilation Signals from Different Participants. In: Cheng L., Leung A., Ozawa S. (eds) Neural Information Processing. ICONIP 2018. Lecture Notes in Computer Science, vol 11305. Springer, Cham
4. Paszke, Adam, Gross, Sam, Chintala, Soumith, Chanan, Gregory, Yang, Edward, DeVito, Zachary, Lin, Zeming, Desmaison, Alban, Antiga, Luca & Lerer, Adam (2017). Automatic differentiation in PyTorch.
5. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 2825-2830.
6. Han, J., Choi, D., Park, S. et al. Hyperparameter Optimization Using a Genetic Algorithm Considering Verification Time in a Convolutional Neural Network. J. Electr. Eng. Technol. 15, 721–726 (2020)
7. Partala, T. & Surakka, V. (2003) Pupil size variation as an indication of affective processing. International Journal of Human Computer Studies 59, 185–198.