Effectiveness of Distinctiveness Pruning on an LSTM based Recurrent Neural Network

Campbell Rankine

Research School of Computer Science, The Australian National University

Abstract: An analysis on the effectiveness of *Distinctiveness Pruning* on the Subjective Belief time series dataset. The basic network is a 3 layer simple neural network consisting of 3 input neurons, 50 hidden neurons, and 1 output neuron. The network was trained using the *3 fold cross validation* method and validated once more on a seperate validation set. However both of these fail to provide an advantage over the network proposed by Xuanying Zhu *et Al*[3]. It was found that while distinctiveness pruning did not improve the accuracy it effectively reduced the size of the model, without a significant drop in accuracy.

Keywords: *Distinctiveness Pruning, Recurrent Neural Networks, Leave-One-Out-Cross-Validation, Veracity Judgement*

1 Introduction

According to Bond and DePaulo[1,2] humans are barely able to provide veracity judgements accurate over 54%. This is becoming especially important in today's world with the rise and increase in availability in online news. With false information being widely available to us as internet users it poses the question of whether or not a network can analyse data over time to apply its own veracity judgement. To answer this the Subjective Belief time series dataset provided by Xuanying Zhu *et Al*[3] was applied. The dataset is composed of 4 sections relating to involuntary responses from the human body. These include Blood Volume Pulse, Galvanic Skin Response, Skin Temperature, and Pupillary Dilation. In this dataset both Left Pupil, Right Pupil, and BVP data were used and we apply the *Distinctiveness Pruning* method proposed by Xuanying Zhu *et Al*[3]. The Recurrent Neural Network is constructed for the analysis of the sequence data. After this the network is pruned based on the values from the weight matrix. The technique used to prune using the weight matrix is by merging similar neurons and deleting neurons that counteract each other, otherwise known as *distinctiveness pruning*[4].

2 Method

The Recurrent Neural Network constructed consists of an input layer of 3 for each of the 3 Time Series features we will be using. Behind that there is 50 hidden layer neurons, and then a linear connection to a sigmoid output function. Additionally, the Loss Function of choice was *Binary Cross Entropy Loss*, and the optimiser selected was the *Adam* Optimiser. The Model was trained on 3 fold cross validation on 75% of the total dataset, losses and accuracies were averaged across the 3 folds in training. The other 25% was saved as a final analysis of the accuracy of the model. The pruning technique implemented is based off of the method proposed by Tom Gedeon[4] where similar neurons are merged and counteractive neurons are deleted.

2.1 Data Inspection and Pre-Processing

Each input feature is normalised to be equal length for every pid_vid. However the lengths across all the data vary as we change video. The data is extracted from the PD csv by splitting into Left Pupil(LP) and Right Pupil(RP). We obtain the length of the video by subtracting the first 2 timestamps to find length. We decide which starts later BVP or Pupil data and pick the one that starts later so we know we have data inside the video. BVP data was also found to be significantly larger due to the sample rate. So BVP is compressed at a ratio of 64:60 where data points are simply dropped. The size of the data is equal to 304 as we cannot train the model without pupillary data.

When Left Pupil and Right Pupil are extracted they are split into their pid_vid videos. *Fig 1.* and *Fig 2.* Show that there is an overall increase, for video number one, and that they are similar shapes. BVP, *Fig 3.* however is shown to be quite cyclic with the occasional increase in the amplitude of the cycle. This does differ slightly between data points however the overall pattern of the time series data stays the same. Temp and ADP, *Fig 4.*, *Fig 5.* were considered to be too constant to improve model performance, and in fact were found to hinder it. An attempt was made to average the Temp and ADP and use that as a feature, however that also hindered performance.

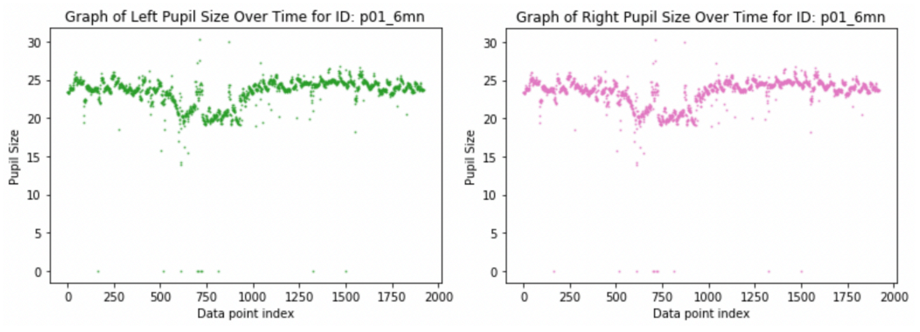


Fig. 3. Graph Left and Right pupil size over time.

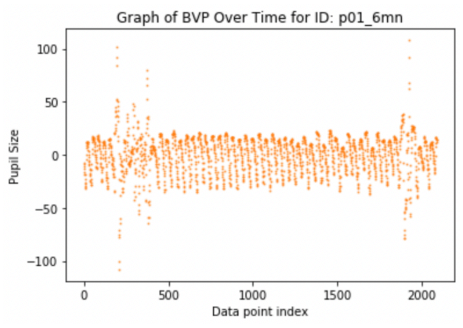


Fig. 4. Graph BVP over time.

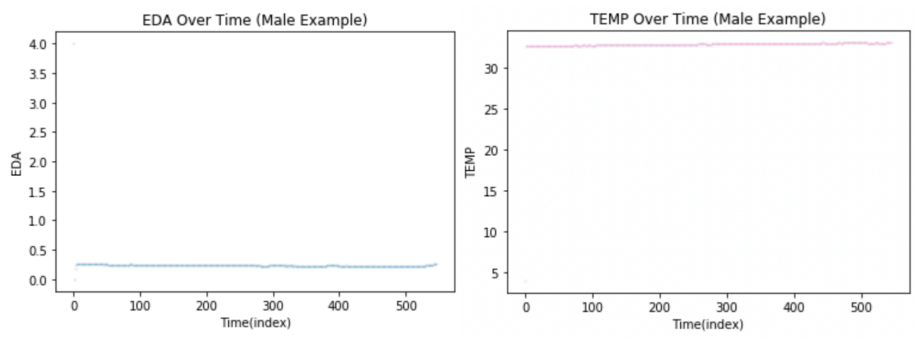


Fig. 5. Graph EDA and Temp over time.

2.2 Define a Neural Network

Given we have 2 input features, Left Pupil and Right Pupil, we must create 2 input neurons for the network. Next we define a hidden neuron size of 50. This is small training LSTM networks is time inefficient and makes it very hard to see results to optimise the hyper-parameters. Anything above 50 hidden neurons negatively affected the performance of the model, however if the model is pruned we inflate the number of similar neurons by increasing the hidden layer size to 100. *Binary Cross Entropy Loss* was selected based off of techniques used by Ho *et Al*[5] as it is optimal for binary classification. *Binary Cross Entropy Loss* is given by:

$$\text{Loss} = -\frac{1}{\begin{array}{c}\text{output}\\ \text{size}\end{array}} \sum_{i=1}^{\begin{array}{c}\text{output}\\ \text{size}\end{array}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log\left(1 - \hat{y}_i\right) \ . \qquad \textbf{(1)}$$

When setting the hyper-parameters of the LSTM network we initialise H0 and C0 to be random numbers in a Tensor of the size *(1,1,hidden_layer_size)*. The dropout rate, which describes randomly disabled connections to force the network into not relying on 1 weight in the layer was set to 0.2 or 20% of the connections. Initially the number of epochs and the learning rate were set to 60 and 0.05 respectively, however after implementing the *3-Fold Cross Validation* the learning rate was adapted to 0.001 and the number of epochs was changed to 25, as that was when the model converged. Additionally, the model was trained on the *Adam* optimiser.

Finally, the training structure for the model was *3-Fold Cross Validation*(3fCV). Due to the inefficient training of LSTM models instead of the *LOOCV* training structure proposed by Xuanying Zhu *et Al*[3], we define a k-fold cross validation to still train over the entire data set and use the entire data set at one point as a test point. The data is shuffled prior to the cross validation. We select 3 folds to increase the size of the test data as that speeds up training while keeping the train set relatively large. Additionally one final shuffled split of the data is used as a validation set. In defining the neural network we provide a way to train the network with a simple back propagation algorithm[3]. This allows for comparison of the models.

Finally the learning rate is adjusted via a cyclic learning rate scheduler *Fig 6*. Based on a cosine function. It takes the learning rate as the absolute maximum and proceeds to lower it over time. The function is described by:

$$(lr/100) + (lr - (lr/100)) * (1 + cos(\pi * (epoch \bmod tmax)) \qquad \textbf{(1)}$$

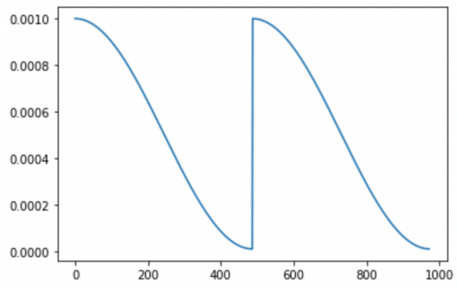


Fig. 6. Graph of Cyclic Learning Rate Scheduler through the number of steps.

This is because as stated by L.N. Smith[9], cyclic learning rate scheduling can increase the speed of convergence. Additionally when applying a more common technique, exponential decay of the learning rate, we observe instances of vanishing gradients described by Hochreiter[10]

2.2 Implement Network Pruning

Pruning a network removes neurons that are redundant or overall detrimental to the training of the network. Therefore it is important to remove these neurons. Assessing the redundancy of the neurons is done by calculating the distinctiveness between nodes. The distinctiveness between to nodes is calculated by:

$$\text{Distinctiveness} = cos^{-1}\left(\frac{\text{A} \cdot \text{B}}{||\text{A}||\,||\text{B}||}\right) \qquad \textbf{(1)}$$

After calculating the distinctiveness between two neurons the evaluation is as follows. If the angle of separation is less than or equal to 55 degrees the neurons A and B are merged by adding the weights and removing neuron B[4]. This is because the two nodes are too similar and adding them results in an average of the 2. Otherwise if the angle of separation is greater than or equal to 125 degrees the neurons A and B are both removed as they are counteractive. The values 125 and 55 were selected as the values described in[4] do not prune the network as it is sparse. The method for removing neurons is to set the weights of the neurons in the pattern set and the output activation set and the bias of the pattern set equal to 0. In each training cycle the largest angle and the smallest angle between the hidden to hidden weights and biases of two neurons are calculated. Since the hidden to hidden neurons are calculated we must Transpose the weight matrix for dimensions to lineup with the output activation weight matrix. If these are greater than or less than the thresholds it chooses the pair that is most distant from its threshold. If these are equal values then the largest angle will be pruned. The selected neuron is returned to a list of removed neurons after being pruned. Before the network prunes the most redundant neuron it sets all of the weights and biases of neurons that have already been pruned equal to 0. This way the neurons stay pruned. The network then retrains after the pruning until the number of epochs is complete. This is demonstrated by *Fig* 7. One of the disadvantages of *Distinctiveness Pruning* is that it is computationally complex[4] and was found to significantly increase the run time in training. Another issue with *Distinctiveness Pruning* is that when adjusting the weights it lead to errors in the back propagation algorithm due to the Tensors used by PyTorch. These issues were fixed with the line:

```
with torch.no_grad():
```

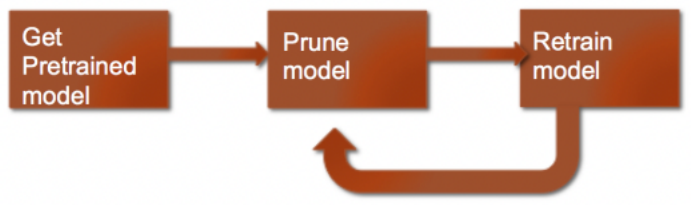


Fig. 8. Figure for the training cycle of a pruned network.

2.4 Displaying Results

To see results of the model the angles and indices of neurons merged and the neurons deleted were displayed, along with which neuron was deleted. These are done in the training stage so that we can see which neurons have been pruned. The formatting for this is in a python list containing:

```
[first index, second index, angle, binary value to show if removed/merged]
```

These values are printed at the end of every training cycle. Also a graph of the loss and accuracy over epochs for both training and testing is provided, along with the confusion matrix for each. Lastly the loss, accuracy, and confusion matrix for the validation set is displayed at the end. Test accuracy was calculated by calculating the mean testing accuracy across each of the three folds.

3 Results and Discussion

The performance of the pruned model versus the dropout model is seen in the evaluation metrics provided.

3.1 The LSTM Network

The training loss, seen in *Fig 9.* overall decreases and converges at around 25 epochs. It bounces up and down at relatively regular intervals which could potentially be from the cyclic learning rate scheduler.

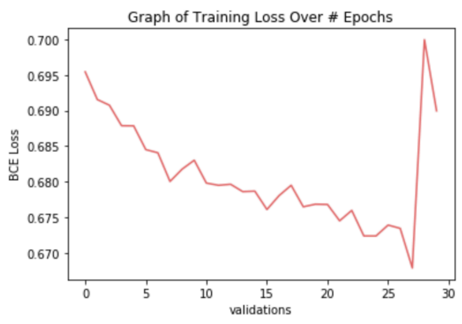


Fig. 9. Graph the training loss with respect to the current epoch

The testing accuracy follows a similar pattern to the inverse of the training loss. However the testing loss, *Fig 10.* seems to also be the inverse of the training loss. It increases through the number of epochs. The testing accuracy has large peaks and strong valleys that coincide with the peaks and valleys of the testing loss.

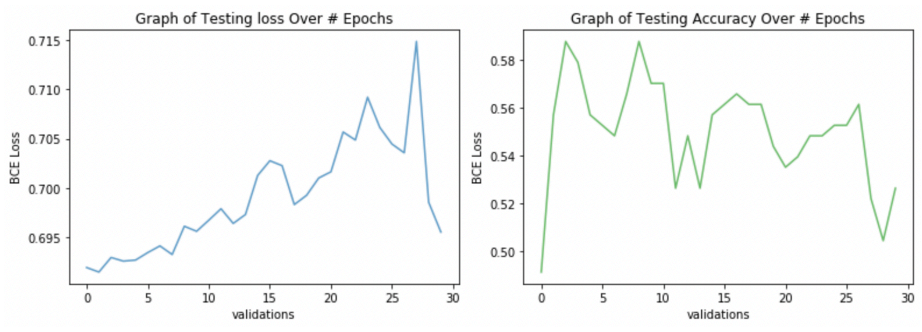


Fig. 10. Graph the testing loss and testing accuracy with respect to the current epoch

The final training loss was 0.7848 and the final training accuracy was 62.47%. The final testing loss was 0.6891 and the final testing accuracy was 62.82%. The confusion matrices of each, *Fig 11*. show that in both the training and testing predictions the rnn was very good at predicting the correct negative values.

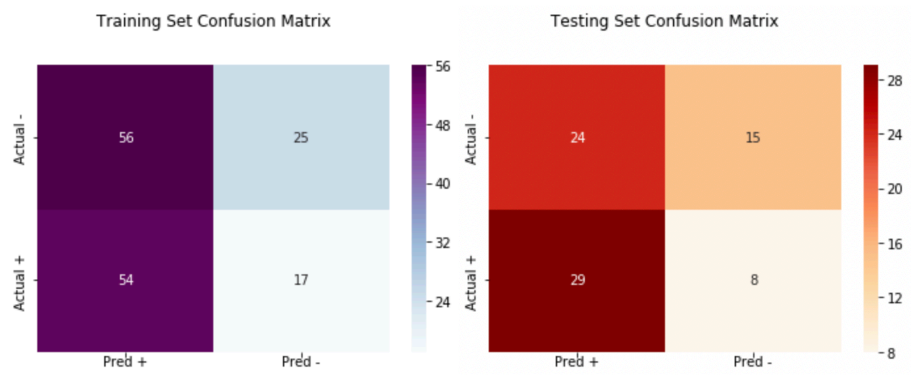


Fig. 11. Heat map of the the training and testing confusion matrices.

The Validation Loss and accuracy for the LSTM model was 0.7075 and 60.42%. And the confusion matrix *Fig 12*. Is below. The difference between the train and test loss/accuracy and the validation loss/accuracy is high.

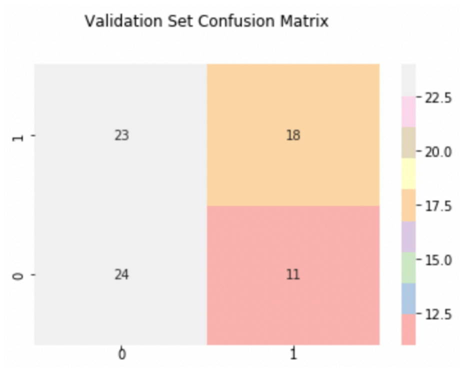


Fig. 12. Heat map of the the training and testing confusion matrices.

3.2 The Pruned LSTM Network

The training loss of the pruned network, seen in *Fig 13*. overall decreases and converges at around 25 epochs. It bounces up and down at relatively regular intervals which could also potentially be from the cyclic learning rate scheduler.

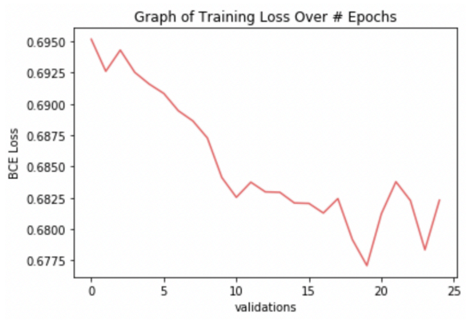


Fig. 13. Graph the training loss of the pruned network with respect to the current epoch

The testing accuracy follows a similar pattern to the inverse of the training loss again. The Accuracy of the testing model stays consistently around 55.7% after about 13 epochs and slightly drops off a little bit as the loss increased, before rising back up. The loss follows a similar curve to the accuracy.

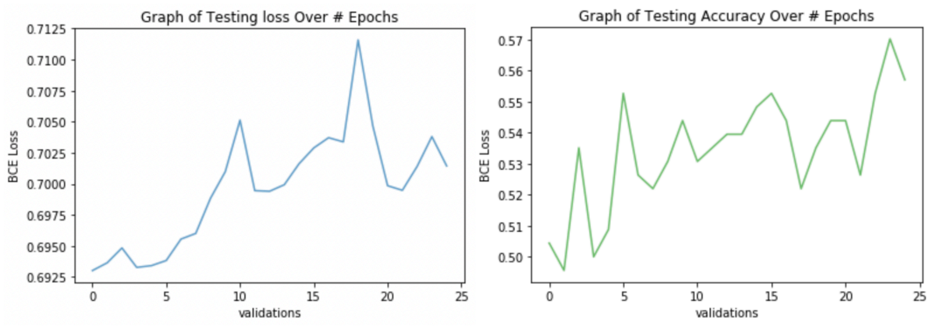


Fig. 14. Graph the testing loss and testing accuracy with respect to the current epoch

The final training loss was 0.6678 and the final training accuracy was 55.04%. The final testing loss was 0.709 and the final testing accuracy was 55.7%. The confusion matrices of each, *Fig 11*. show that the pruned LSTM network was better at predicting positive values than negative values

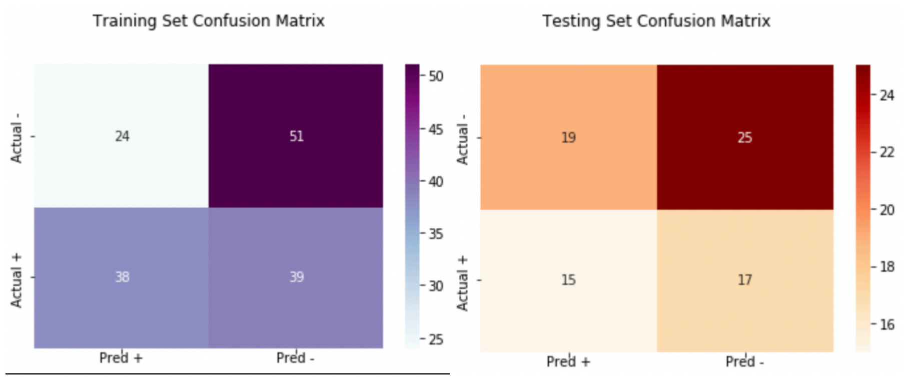


Fig. 11. Heat map of the the training and testing confusion matrices.

The Validation Loss and accuracy for the Pruned model was 0.745 and 64.47%. And the confusion matrix *Fig 12*. Is below. The difference between the train and test loss/accuracy and the validation loss/accuracy is high.

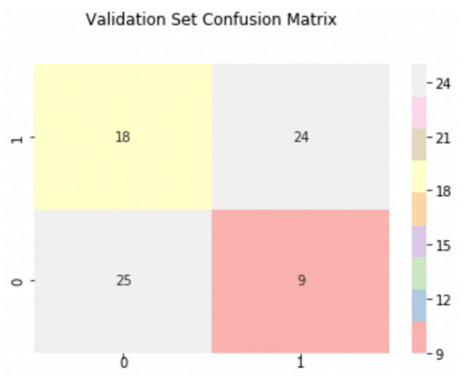


Fig. 12. Heat map of the the training and testing confusion matrices.

3.3 Discussion

The pruned model was able to reduce the overall size of the model by pruning some of the neurons while mostly maintaining validation accuracy, with only a drop of 1.31%, while reducing the size of the model by 9 connections, as seen by the list of pruned connections in the Appendix. Both models had similar training structures and training accuracy. However the pruned model seemed to perform better through the epochs with testing accuracy compared to the unpruned model. The unpruned model was able to score higher on the validation set. Both models were more capable of predicting positive values than the negative values. Both models hover and experience peaks and valleys in both the loss and accuracy during training. The testing accuracy for both also follows a similar shape. Due to these reasons it seems like both models also reverse the findings by Xuanying Zhu *et Al*[3] in predicting more of the positive values than the negative values.

The model developed by Xuanying Zhu *et Al*[3] also uses pupil size metrics to predict veracity judgements. What they find is an overall accuracy of 58%[3] However this data was data that was already previously observed by the network. Specificity of the their model was 50% and sensitivity was 66.7%[3]. Comparing this to the unpruned LSTM Network constructed above in the validation set we find a specificity of 56.1% and a sensitivity of 31.43%, *Fig. 5*. The unpruned model is therefore significantly worse at detecting the trust condition[3]. Comparing their accuracies then reveals that in general the unpruned model performed worse. This may have been an effect of the unreliability in training and the structure of the LSTM network.

Comparing the results found in Xuanying Zhu *et Al*[3] to those found by the pruned network we show that the difference in overall accuracy and validation accuracy between the two models is 4.05%. This is a very small difference in accuracy showing these networks overall perform similarly. The confusion matrix of the pruned network, *Fig. 9*, shows a specificity of 57.1% and a sensitivity of 73.5%. This is interesting as it follows the same patterns from the findings in Xuanying Zhu *et Al*[3] for both the neural networks and humans. Humans were found to have a specificity of 40% and a sensitivity of 60%[3]. Mirroring to a more accurate extent the results found by their network. The difference with the pruned network is that it performs very similarly both both sensitivity and specificity, leading us to believe that the pruned LSTM network is better at predicting the negative, not lying, values than predicting if someone is presenting false information. However due to the similarity of all of the models in accuracy, and training, we must conclude that while pruning does not improve accuracy it successfully reduces the size of the model, via the weight matrix, without a significant drop in accuracy.

3.4 Limitations

While deep learning techniques were applied, LSTM networks in themselves have a few issues. For instance taking a long time to train, and vanishing gradients. These were an issue to deal with for most of the training. Occasionally the network would stop learning entirely after the first epoch. Not only this but the training time for the LSTM model was incredibly long for the 25 epochs so it was decided that we should only use Pupillary data, and not BVP. Also it was impossible to see if the network converged beyond 50 epochs. There was one training run that went to 60 epochs that seemed to show the network converging at 30, but there is no way to know if the network would have improved beyond 60 epochs.

4 Future Work

Due to the limitations provided by the LSTM Network an extension of this model would be to try a GRU structured RNN, which could potentially deal with this specific time series data better. Another good approach would have been via a 2-D convolutional network. Seeing if we could replicate similar accuracies with a 2-D convolutional network would be a good approach as 2-D CNN's train significantly faster and would've been much more efficient in modelling the data, if this property held. One final approach would have been to have taken the output of the LSTM network, and the average temp and EDA and inputting these values into a simple neural network of input size 3, and classifying the data with the final simple network. So the overall topology of the network would include 2 neural networks and all 5 features.

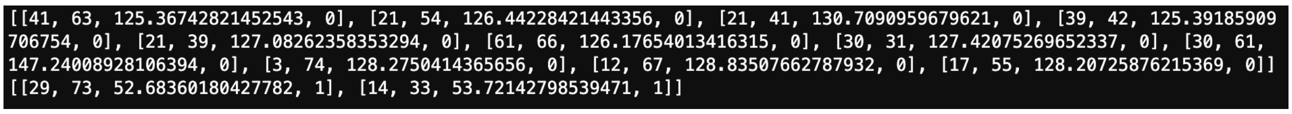
5 Conclusion

Distinctiveness pruning on a Recurrent Neural Network was attempted to see if we could prune networks based on their values in the pattern set. Pruning did not increase the validation accuracy between the pruned and unpruned models, however it did decrease the size of the pruned model. This provides an advantage over the unpruned recurrent neural network as it will be able to evaluate data faster after pruning. Pruning also seems to change the specificity and sensitivity when compare to the findings in *Xuanying Zhu et Al*[3]. The subjective belief problem seems to have been solved. We can construct a network that predicts veracity judgements more accurate when compared to human judgements.

References

1. Bond Jr., C.F., DePaulo, B.M.: Accuracy of deception judgments. Pers. Soc. Psychol. Rev. 10, 214–234 (2006)
2. DePaulo, B.M., Bond Jr., C.F.: Beyond accuracy: bigger, broader ways to think about deceit. J. Appl. Res. Mem. Cogn. 1, 120–121 (2012)
3. Zhu, X., Qin, Z., Gedeon, T., Jones, R., Hossain, M. Z., & Caldwell, S. (2018, December). Detecting the Doubt Effect and Subjective Beliefs Using Neural Networks and Observers' Pupillary Responses. In International Conference on Neural Information Processing (pp. 610-621). Springer, Cham.
4. Gedeon, TD.: Indicators of Hidden Neuron Functionality: the Weight Matrix versus Neuron Behaviour. Centre for Neural Networks, Kings College, London, 1995.
5. Ho, Yaoshiang & Wookey, Samuel. (2019). The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2962617.
6. Gedeon, TD, Harris, D, "Network Reduction Techniques," Proc. Int. Con$ on Neural Networks Methodologies and Applications, AMSE, vol. 1, pp. 119-126, San Diego, 1991.
7. Xue Ying 2019 J. Phys.: Conf. Ser. 1168 022022, 2019.
8. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.
9. L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 464-472, doi: 10.1109/WACV.2017.58.
10. Hochreiter, Sepp. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 6. 107-116. 10.1142/S0218488598000094.

Appendix:



```
[[41, 63, 125.36742821452543, 0], [21, 54, 126.44228421443356, 0], [21, 41, 130.7090959679621, 0], [39, 42, 125.39185909
706754, 0], [21, 39, 127.08262358353294, 0], [61, 66, 126.17654013416315, 0], [30, 31, 127.42075269652337, 0], [30, 61,
147.24008928106394, 0], [3, 74, 128.2750414365656, 0], [12, 67, 128.83507662787932, 0], [17, 55, 128.20725876215369, 0]]
[[29, 73, 52.68360180427782, 1], [14, 33, 53.72142798539471, 1]]
```