Fine-tuning Neural Networks using Keras Tuner and Fuzzy Algorithm

Shawn Kinn Eu Ng
Research School of Computer Science
Australian National University
u7025048@anu.edu.au

Abstract. It is difficult to determine how shallow or deep should your neural networks be to solve a particular issue. That is why understanding hidden layers is necessary for a data scientist in solving complex problems quickly and efficiently. However, it is difficult to determine how many hidden layers to use or any other parameters such as the best learning rate or the number of epochs. There are so many decisions and factors that will affect hidden unit dynamics. This document focuses on the experimentation of cross-validation method and hyperparameters finetuning to improve the accuracy of neural networks. We will also explore the Keras Tuner library and determine if the hyperparameters recommended by Keras will produce better result. Fuzzy Algorithm will also be applied to partition the data to better understand the data. We will determine if the fuzzy algorithm will be able to play a significant role in building our neural networks.

Keywords: Neural Network, Keras Tuner, Fuzzy Algorithm

1 Introduction

"Neural networks reflect the behavior of the human brain, allowing computer programs to recognize patterns and solve common problems in the fields of AI, machine learning and deep learning". It consists of input layers, hidden layers and output layers. Generally, the higher the number of hidden layers, the more complex the network can solve at the expense of a more time-consuming process.

To find minimal size networks, many data scientists have opted to use brute force methods. They do so "by eliminating randomly chosen neurons from trained networks"[1] but that requires experimentation and time-consuming methods. Through a series of experiments, we will show that tweaking the hidden number of neurons, epoch chosen and learning rate, we can improve the accuracy of the model. However, to improve efficiency in terms of workload and time spent on trial and errors, we will be finetuning the network using Keras Tuner. Keras Tuner is a library that helps pick the best set of hyperparameters for each model. We will attempt to "remove redundant or irrelevant units such as nodes, filters or layers"[2] which are not critical for neural network performance according to the Keras Tuner's recommendation.

Additionally, we will be considering the Fuzzy Algorithm to help cluster our data. The clusters will be plotted to provide more information about our data in hopes of identifying the rules that governs the data.

[1] Gedeon, T., 1995. *Indicators of Hidden Neuron Functionality: the Weight Matrix versus Neuron Behaviour*. [online] London. Available at: <http://users.cecs.anu.edu.au/~Tom.Gedeon/pdfs/Indicators%20of%20Hidden%20Neuron%20Functionality%20the%20Weight%20Matrix%20versus%20Neuron%20Behaviour.pdf> [Accessed 27 April 2021].

[2] Yeom, S., Seegerer, P., Lapuschkin, S., Binder, A., Wiedemann, S., Müller, K. and Samek, W., 2021. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115, p.107899.

2 Data Description and Preprocessing

In this experiment, presenters were invited to convey information to our experiment participants. However, half of the presenters were presenting on the belief that the information is accurate while the other half were manipulated into thinking the information was inaccurate, but they had to present it to the participants naturally anyway. While participants watch the presentation, their physiological signals – Blood Volume Pulse (BVP), Galvanic Skin Response (GSR), Skin Temperature (ST) and Pupillary Dilation (PD) were recorded. This was done to determine if a neural network could perform better than the observer's conscious verbal judgement to determine whether the presenter's belief was manipulated.

The data we received recorded 34 columns of BVP data, 22 columns of GSR data, 22 columns of ST data and 38 columns of PD data. For this report, we have processed our raw data into 2 excel files.

(1) The first 'subjective_belief_edited_average.csv' was averaged to allow to receive one single input, one representation for each participant's BP, GSR, ST and PD. In comparison to plotting all 116 columns of data, we can focus on 4 to better visualize and analyze the information.

(2) The second named 'subjective_belief_edited_new.csv' was normalized. "Normalization reduces the differences between the variation range of the different variables"[3] to reduce the likelihood of overfitting to occur. Additionally, normalization introduces a level of "'"orthogonality between layers"[4], allowing the network to learn the parameters in the next layer more efficiently.

3 Methodology: Neural Network

According to the "Detecting the Doubt Effect and Subjective Beliefs Using Neural Networks and Observers' Pupillary Responses" report, the model had to meet the following requirements:
- Neural networks should be trained on the participant's individual and joint judgments.
- All neural networks is built with a sigmoid function with 100 hidden neurons and 2 output neurons.
- The network should use Adam optimization algorithm without any adaption using backpropagation with Cross-Entropy for its loss function.
- Leave-one-out cross-validation (LOOCV) should be utilized.

I will be building a Two- Layer neural network that meets all the requirements of the model but tweaking the number of epochs, the number of layers and the learning rate through a series of trial and error to achieve the highest accuracy. I believe that the model used in the study could be further improved to make better predictions.

Besides pure experimentation, we will also rely on relevant guidelines: For example, the following rules listed to decide the number of hidden layers:
"(1) Number of hidden neurons should be between size of input layer and output layer.
(2) Number of hidden neurons should be 2/3 the size of input layer plus, the output layer
(3) Number of hidden neurons should be less than twice the size of input layer"[5]

Besides the hyperparameters, this paper will also compare two cross-validation methods – namely leave-one-out as specified by the report and the 80/20 rule - training 80% of data and testing with the remaining 20%. This is because the dataset in this case study is sufficiently large and it is highly likely that the leave-one-out method will introduce a higher level of bias to the model.

Lastly, we will be comparing the hyperparameters we chose through experimentation with Keras Tuner's recommended parameters. We will also be retraining the model using the optimized parameters to compare the accuracy before and after using Keras Tuner.

---

[3] Sola, J. and Sevilla, J., 1997. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3), pp.1464-1468.

[4] Jordan, J., 2018. *Normalizing your data (specifically, input and batch normalization).* [online] Jeremy Jordan. Available at: <https://www.jeremyjordan.me/batch-normalization/> [Accessed 27 April 2021].

[5] Heaton, J., 2017. *The Number of Hidden Layers*. [online] Heaton Research. Available at: <https://www.heatonresearch.com/2017/06/01/hidden-layers.html#:~:text=An%20inordinately%20large%20number%20of,adequately%20train%20the%20neural%20network.> [Accessed 27 April 2021].

4 Findings: Neural Network

To determine if we could build a network that performs better than the results generated in the report, we will be analyzing the hyperparameters chosen in the study. The Pareto Principle, the 80/20 rule to split training and testing data is well known for explaining many human phenomena [6] but the study has opted for LOOCV. Therefore, we will be comparing the LOOCV with the 80%-20% train-test (80-20) split results. By comparing the historical loss diagrams, the loss is much lower at 0.21 in LOOCV compared to 0.64 in LOOCV. The statistics suggest that the LOOCV is performing much better at predicting whether the presenter's belief was manipulated compared to 80-20. Additionally, LOOCV achieved an accuracy of 94.29%, only making prediction mistake 5.71% of the time. This suggests that the paper's choice of LOOCV to cross-validate the model is valid.

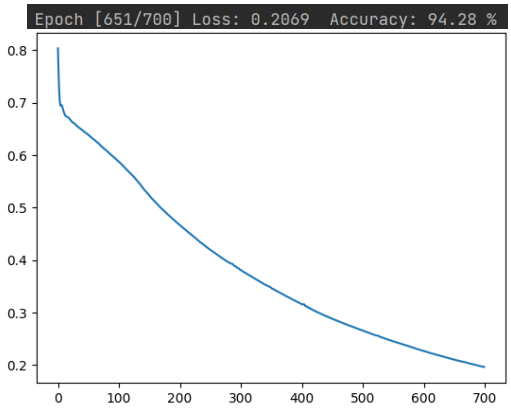


Fig. 1. Historical Loss with LOOCV.

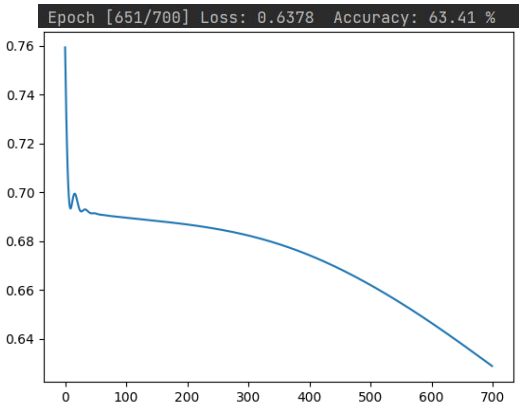


Fig. 2. Historical Loss with 80-20 Split.

LOOCV performed exceedingly well on the testing data, achieving a 100% testing accuracy (Figure 3). However, the statistics computed over the validation data for LOOCV does raise some concern. A 100% training accuracy suggests possible overfitting possibly due to the model's failure to regularize. By overfitting the training set, the model also picked up the noise from the training data and result in a poor generalization of unseen data. The confusion matrix confirms that the model makes the same prediction for all 367 participants and is not a good model as we previously thought.

On the other hand, the model that utilized the 80-20 rule only achieved an accuracy of 45.95% (Figure 4), which is worse than the participants. The participants managed to achieve an accuracy of 54%. However, it is still possible to prune the network, fine-tune the parameters and train the network further.

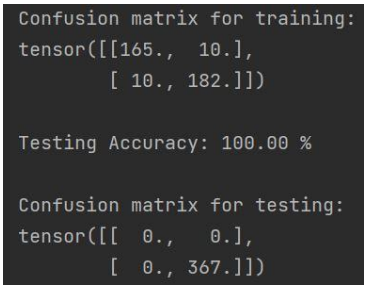


Fig. 3. LOOCV Confusion Matrix.

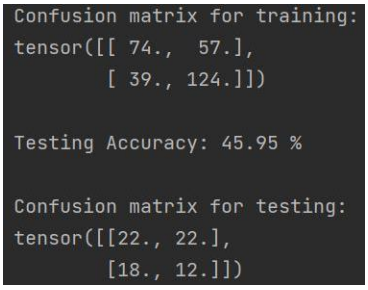


Fig. 4. 80-20 Split Confusion Matrix.

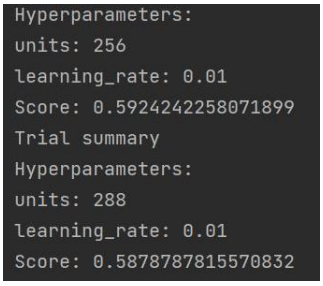


Fig. 5. Keras Tuner Hyperparameters.

By applying Keras Tuner to the model, it will help us choose the right set of hyperparameters for our model. From Figure 5, we can see that the optimal number of units in the first densely connected layer is 256 and the optimal learning rate for the optimizer is 0.01. Using these hyperparameters, we should theoretically be able to achieve an accuracy of 59.2%, performing better than the human participants. By retraining the model, we will receive the result in Figure 6 – improving the accuracy of the model at the expense of higher possibility of prediction mistakes.

---

6 Towards Data Science. 2020. *Finally: Why We Use an 80/20 Split for Training and Test Data Plus an Alternative Method (Oh Yes…)*. [online] Available at: <https://towardsdatascience.com/finally-why-we-use-an-80-20-split-for-training-and-test-data-plus-an-alternative-method-oh-yes-edc77e96295d> [Accessed 1 June 2021].

```
[test loss, test accuracy]: [0.6660192012786865, 0.6712328791618347]
```

Fig. 6. Retraining Model with Keras Tuner

Additionally, Keras Tuner will also be able to tell users the number of training epochs needed in the model. Purely by experiment, I decided to have 700 training epochs in my model since it has the highest probability of producing the highest accuracy. As shown in table 1, the highest accuracy result was in Epoch interval 651 to 700 5 times, much higher than the other intervals.

Table 1. Epoch Intervals with the Highest Number of Accuracies.

Number of Epochs	Frequency of Best Accuracy in the Epoch
51-100	2
251-300	1
301-350	2
401-450	1
601-650	2
651-700	5

However, this took a lot of trial and error. With Keras Tuner, the library will inform us the best epoch for our model, removing the necessity for experimentations and iterations each time we have to build a model. For example, in Figure 7, the optimal epoch for our model is 137.

```
Best epoch: 137
Epoch 1/137
10/10 [==============================] - 1s 18ms/step - loss: 0.9805 - accuracy: 0.4237 - val_loss: 0.6747 - val_accuracy: 0.5753
Epoch 2/137
10/10 [==============================] - 0s 4ms/step - loss: 0.7022 - accuracy: 0.5186 - val_loss: 0.8312 - val_accuracy: 0.4110
```

Fig. 7. Optimal Number of Epoch provided by Keras Tuner

The paper also suggested using a hidden neuron of 100. But figure 8 shows that a hidden neuron of approximately 90 will produce the highest accuracy of 88%, which was the recommendation of Keras Tuner. With the help of Keras Tuner, we will also no longer need to experiment with learning rates (as shown in Table 2).

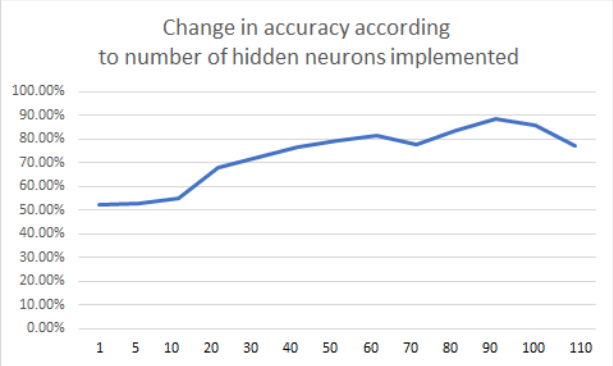


Fig. 8. Experimentation with Hidden Neurons

Table 2. Experimentation with Learning Rates.

Learning Rate	Loss	Training Accuracy	Testing Accuracy
0.01	0.15	96.91%	59.74%
0.001	0.64	59.40%	57.14%
0.0001	0.69	55.67%	41.18%

With the learning rate of 0.01, the model calculated an accuracy of 97.00%. Even though the accuracy is much higher compared to the model trained with Keras Tuner, the training accuracy of 59.74% suggests the model does not do very well with unseen data, possibly due to another overfitting situation. It is highly likely that the model we have will not perform better than a 67% accuracy, as calculated by Keras Tuner.

It is clear that the parameters stated in the report used for the model were not the best choices since it only managed to achieve 60% accuracy. There are definitely improvements we could make to cross validation methods and hyperparameters, such as Epoch and number of layers. Since we do not have rules to determine these parameters besides experimentation, Keras Tuner on the other hand will be able to determine that hyperparameters in the shortest amount of time i.e., epoch 137, learning rate of 0.01 and 256 number of layers for this model will yield the highest accuracy. Therefore, Keras Tuner is definitely beneficial in help fine tune the model without wasting too much time. However, the accuracy is still pretty low and we will have to explore larger datasets to improve our model, a small sample size will not "be fixed by better classifiers or cross-validation approaches" (Varoquaux, 2017).

Since the paper focused on pupillary responses, we will attempt to draw links between variables using fuzzy logic. To better understand the "variations that occur in the uncertain and vague world"[7], to draw meaningful observations from out data, applying fuzzy logic to represent uncertainty.

5 Methodology: Fuzzy Logic

To create a fuzzy control system, we had to decide on our antecedents, consequents, and rules. The system will help us decide if the physiological signals suggest doubts about their belief.

Table 3. Custom Membership Function.

Data	Low	Medium	High
BVP	< 50	50 – 70	> 70
GSR	< 3.0	1.5 – 4.5	3.0 – 6.0
ST	<0.8	0.8 – 1.6	1.6 – 2.4
PD	<0.5	0.5 – 0.9	0.9 – 1.4

For BVP, the normal range is between 50 and 70 beats per minute. So, values below 50 are too low and values above 70 are considered high. For GSR, ST and PD, the data recorded are not within normal range, so we will be categorizing them into 3 intervals of same width. For the purposes of this system, any value categorized as high is considered abnormal and is likely a symptom of disbelief.

To define the fuzzy relationship between the 4 inputs (BVP, GSR, ST, PD) and the output (True/ False), I considered 3 simple rules:

(1) If BVP is low or GSR is low or ST is low or PD is low, then belief is true.
(2) If BVP is medium or GSR is medium or ST is medium or PD is medium, then belief is true.
(3) If BVP is high or GSR is high or ST is high or PD is high, then belief is false.

The average value of each input is passed through the control system to generate the degree of match between the fuzzy input and rules.

Typically, defuzzification is required. However, since the values below 0.5 are false and those above are true. I decided to forego the defuzzification step and round the values to the nearest integer, comparing them directly with the values in the excel sheet for a match.

Fuzzy C-Means Clustering was also conducted using 2 clusters and fuzzy parameter 1.1 (since it provides the highest accuracy as seen in Table 4). Fuzzy C-Means will have 3 iterations, centers at origin, centers at random locations within multivariate gaussian and centers at random value chosen from data to partition the data. The data will then be plotted.

Table 4. Fuzzy Parameter.

Fuzzy Parameter	1.1	1.3	1.5	2	3	4	5	6

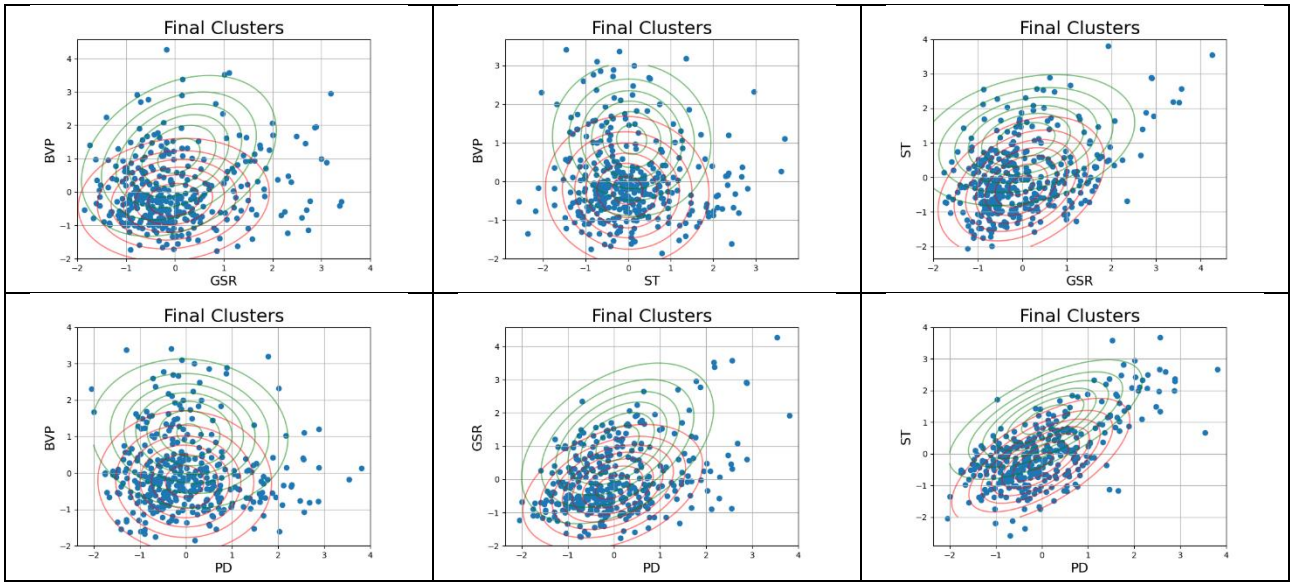
[7] Altmann, W., Macdonald, D. and Mackay, S., 2005. *Basic principles of fuzzy logic and neural networks*.

Accuracy	40.05	38.15	35.5	29.16	26.98	26.43	26.43	26.43

6 Findings: Fuzzy Logic

The rules generated did not produce any meaningful results, having 184 accurate matches and 184 wrong matches. This means that we have a wrong representation of the results. By analyzing the fuzzy clusters, we hope to set rules that could perform better than a 50-50 chance.

Fig 9. Fuzzy Clusters.



From these clusters, we can see that most data in the true and false clusters overlap due to the use of fuzzy parameters. I decided to change the custom membership function for BVP, GSR and PD from 3 intervals to 2 categories, low and high, evenly split in the middle. Since the green rings represents the true cluster while the red represents the false clusters, we came up with 13 rules as shown in the figure below.

Fig 10. Rules according to Fuzzy Clusters.

```
rule1 = ctrl.Rule(BVP['low'] & GSR['low'], Belief['false'])
rule2 = ctrl.Rule(BVP['low'] & ST['medium'], Belief['false'])
rule3 = ctrl.Rule(BVP['low'] & PD['low'], Belief['false'])
rule4 = ctrl.Rule(GSR['low'] & PD['low'], Belief['false'])
rule5 = ctrl.Rule(GSR['low'] & ST['low'], Belief['false'])
rule6 = ctrl.Rule(PD['low'] & ST['medium'], Belief['false'])
rule7 = ctrl.Rule(BVP['low'] & GSR['medium'], Belief['true'])
rule8 = ctrl.Rule(BVP['low'] & ST['high'], Belief['true'])
rule9 = ctrl.Rule(BVP['low'] & PD['medium'], Belief['true'])
rule10 = ctrl.Rule(GSR['low'] & PD['high'], Belief['true'])
rule11 = ctrl.Rule(GSR['low'] & ST['high'], Belief['true'])
rule12 = ctrl.Rule(PD['low'] & ST['medium'], Belief['true'])
rule13 = ctrl.Rule(BVP['high'] | BVP['medium'] | GSR['high'], Belief['false'])
```

However, even by mapping the rules from the C-Means cluster, we do not have any meaningful results. It is once again split into 184 correct matches and 184 wrong ones, not providing us any useful information. This shows that “a fuzzy

system with many rules may be hard to design and have large storage consumption, high computation complexity and poor convergency in parameter tuning[8]"

The problem with Fuzzy Logic control system is that it is entirely dependent on personal knowledge. Since there was no information about the correlation of these 4 variables – BPV, GSR, ST and PD, I made educational guesses according to my research. For example, increase in sweat gland activity indicates discomfort, hence a high GSR should indicate a disbelief in the information received. However, even after at least many different iterations of the rules, I was unable to achieve any meaningful connection. Setting exact, fuzzy rules and membership function is difficult, especially because it does not have the ability to recognize patterns like a neural network. Without the necessary information or expertise, one should stay clear from applying the fuzzy algorithm.

7 Limitation and Future Work

Since the split of the testing and training data was random, the accuracy changes each time and though the results are always within a 5% interval, I am unable to get the exact same result each time, even after applying a seed.

Additionally, this paper does not actually manually carry out pruning so the impact of weight of hidden neurons on the accuracy of the model was not discussed. It will be interesting to see the impact of weights on the accuracy of the neural network. Each network is different, when sand it is not guaranteed that "a deep neural network with more parameters" will perform better than "a small neural network with fewer layers and neurons." [9]

8 Conclusion

There are many different experiments that could be conducted with the dynamics of hidden units. It is important to note that each dataset is different and there is no fixed model that could be applied to all. Even though there is always room to improve the accuracy of our results i.e., by processing our input data, by altering our cross validation model or by finetuning our hyperparameters, it is more important to note that a high accuracy does not necessarily mean a good model. Overfitting is a very common consequence, and we should be careful not to introduce bias to the neural network. More importantly, the Keras Tuner library removes the need to finetune networks through trial and error. The parameters recommended by Keras Tuner provides a high level of accuracy within a much shorter timeframe. As such, there is no longer a need for manual experimentations with our hyperparameters.

In contrast, Fuzzy Logic is a difficult algorithm that needs to be implemented entirely manually. Without the necessary expertise about the field or extensive information about the data, it is hard to set the exact fuzzy rules and membership functions. Hence, not getting the results you were hoping for. Rather than using fuzzy algorithm to build a better neural network, it is more likely that a trained network can be used to refine the fuzzy rules and build the control system.

In conclusion, do be precise about each decision you make while building a neural network. This may include using the right tools, such as Keras Tuner, the right algorithm, the right cross validation method and the right hyperparameters.

[8] Abdi, Y. and Taheri-Garavand, A., 2020. *Application of the ANFIS Approach for Estimating the Mechanical Properties of Sandstones*. Scholarworks@UAEU.

[9] Tran, T., Lee, T. and Kim, J., 2020. Increasing Neurons or Deepening Layers in Forecasting Maximum Temperature Time Series?. *Atmosphere*, 11(10), p.1072.

References

Abdi, Y. and Taheri-Garavand, A., 2020. *Application of the ANFIS Approach for Estimating the Mechanical Properties of Sandstones*. Scholarworks@UAEU.

Altmann, W., Macdonald, D. and Mackay, S., 2005. *Basic principles of fuzzy logic and neural networks*.

Gedeon, T., 1995. *Indicators of Hidden Neuron Functionality: the Weight Matrix versus Neuron Behaviour*. [online] London. Available at: <http://users.cecs.anu.edu.au/~Tom.Gedeon/pdfs/Indicators%20of%20Hidden%20Neuron%20Functionality%20the%20Weight%20Matrix%20versus%20Neuron%20Behaviour.pdf> [Accessed 1 June 2021].

Gedeon, T. and Harris, D., n.d. *NETWORK REDUCTION TECHNIQUES.*

Heaton, J., 2017. *The Number of Hidden Layers*. [online] Heaton Research. Available at: <https://www.heatonresearch.com/2017/06/01/hidden-layers.html#:~:text=An%20inordinately%20large%20number%20of,adequately%20train%20the%20neural%20network.> [Accessed 1 June 2021].

Jordan, J., 2018. *Normalizing your data (specifically, input and batch normalization)*.. [online] Jeremy Jordan. Available at: <https://www.jeremyjordan.me/batch-normalization/> [Accessed 1June 2021].

Sola, J. and Sevilla, J., 1997. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3), pp.1464-1468.

Towards Data Science. 2020. *Finally: Why We Use an 80/20 Split for Training and Test Data Plus an Alternative Method (Oh Yes…)*. [online] Available at: <https://towardsdatascience.com/finally-why-we-use-an-80-20-split-for-training-and-test-data-plus-an-alternative-method-oh-yes-edc77e96295d> [Accessed 1 June 2021].

Tran, T., Lee, T. and Kim, J., 2020. Increasing Neurons or Deepening Layers in Forecasting Maximum Temperature Time Series?. *Atmosphere*, 11(10), p.1072.

Varoquaux, G., 2017. *Cross-validation failure: small sample sizes lead to large error bars*. [online] Available at: <https://hal.inria.fr/hal-01545002/document> [Accessed 1 June 2021].

Yeom, S., Seegerer, P., Lapuschkin, S., Binder, A., Wiedemann, S., Müller, K. and Samek, W., 2021. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115, p.107899.

Ying, X., 2019. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*, 1168, p.022022.

Zhu, X., Qin, Z., Gedeon, T., Jones, R., Hossain, M. and Caldwell, S., 2018. Detecting the Doubt Effect and Subjective Beliefs Using Neural Networks and Observers' Pupillary Responses. *Neural Information Processing*, pp.610-621.