Shallow Vs Deep Learning for Hidden Layer Reduction with Eye Gaze Data

Patrick Smith
u5978630@anu.edu.au

Research School School of Computer Science
The Australian National University

Abstract. This paper investigates the efficacy of deep learning to extend previous work in neuron pruning for a binary classification task. The goal of this investigation is to see whether a deep learning approach used in conjunction with neuron pruning can realise additional benefits of generalisation and accuracy under the same classification task and neural pruning technique. The classification task itself is to predict whether an image has been digitally manipulated by virtue of a viewing participant's "eye gaze"[1] data, provided by a study conducted at The Australian National University. This dataset is pre-processed, with the most appropriate hyperparameters nominated for the model based on neural network design research and trials with data. Results showed that deep network performance had little test accuracy improvement (≤ 1) over its shallow counterpart, but had performance gains in its generalisation ability throughout training.

Keywords: Deep Learning · Neural Networks · Neuron Pruning · Network Reduction · Eye Gaze.

Introduction

Deep learning is among the fastest growing and most exciting disciplines in Artificial Intelligence (AI). Deep learning excels at tasks which are inherently complex, where no clear pattern in data or deterministic methods to translate that data into desirable outputs are readily available. Deep learning models rely on more hidden layers than traditional one hidden layer networks allow for, in the hopes that these inner layers will be able to extract differing order features to learn novel patterns of data and achieve interesting results [1]. Different deep learning approaches use augmented network structures where functions and layer connection patterns vary depending on the classification task. These approaches influence the way in which data is propagated through the network in the hopes that a certain approach to network structure will imply network intuition [1].

The motivation for this paper in applying deep learning arises from the choice of the dataset and previous work conducted to develop a binary classifier [9]. This dataset is atypical for the application of a deep learning model for two reasons. Firstly, the data is not particularly unstructured, like a matrix of pixels that make up a particular image (image classification task) and secondly, the data does not appear to have features which contain non-deterministic or complex patterns like human speech or written prose (speech recognition or natural language processing tasks). However as the dataset's results identify, despite participants failure to determine image veracity only slightly better than chance, "...extended visual attention to manipulated regions was associated with greater accuracy" [2]. Thus, this paper's application of deep learning seeks to explore whether a deeper model trained with distinctiveness neuron pruning can offer more insight into a participant's ability to discern an image as being legitimate from their eye gaze. We also seek to combine this with all other data provided on the participant in the hopes that our model will be able to pickup on undiscovered patterns of participant behaviour. More broadly, we see whether available patterns of participants actions can actually imply their subconscious ability to determine digital image veracity.

Using decades of pruning research, this paper's deep learning technique builds on the pruning technique for the shallow neural network as in [9], who's technique to prune neurons by neuron activation vector angles is directly inspired by research from [5,4]. This neuron pruning will occur during training after data pre-processing (feature selection, normalisation) is conducted, with the express desire to get more consistent results out of both shallow and deeper models.

[1] refers to the place where one's eyes are looking, or gazing

1 Dataset and Pre-Processing

1.1 Dataset Background

The data used in this investigation is from an experiment investigating to what extent humans are able to perceive manipulated images [2]. Part of this experiment recorded the eye gaze of participants as they viewed images which were either digitally manipulated or left as-is [2].
The features collected for each data point in the experiment are in Table 1 below. The "num_fixs" and "inside_num_fixs" features refer to the amount of times the participant looked inside or outside a manipulated region of the image, respectively, and the "fixs_dur" and "inside_fixs_dur" features correspond to the cumulative duration of eye gazes for inside or outside a manipulated region, respectively. Coupled with this is the other important feature provided, "image manipulated", which is whether the image is actually manipulated (denoted as "1") or not (denoted as "0").
Other auxiliary information provided is the participant's identifier, "participant_ID", and whether the participant decided, "vote", the given image was manipulated (denoted as "1"), or wasnt manipulated (denoted as "0").

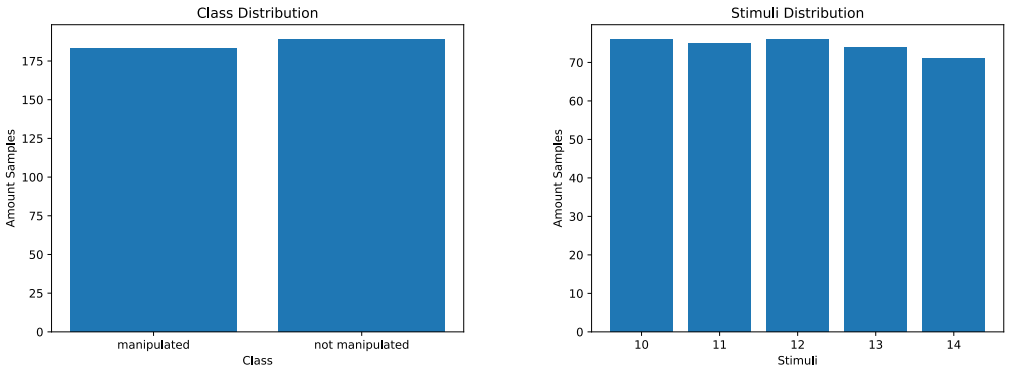
Table 1: dataset features

participant_ID	num_fixs	fixs_dur	inside_num_fixs	inside_fixs_dur	image	image manipulated	vote

The experiment produced a total of 1071 data points, but in our investigation we use only a subset of these points, being 372 data points. Table 1's features are the features which we use and evaluate on.

1.2 Dataset Analysis and Pre-Processing

Sample Distribution Balance Important to the training of any classification model in machine learning is the inspection and analysis of the summary statistics of the dataset [12].
In terms of class and stimuli distribution, the data is well balanced. Fig. 1a and shows a minor class imbalance with Fig 1b similarly showing a minor sample distribution imbalance per-stimuli.



(a) Distribution by Class (b) Distribution by Stimuli

Fig. 1: Distribution Summaries

To ensure that this imbalance truly is negligible, we employ the following adapted distribution entropy function,

$$Balance = \frac{H}{\log k} = \frac{\sum_{i=1}^{k} \frac{C_i}{n} log \frac{C_i}{n}}{\log k} \tag{1}$$

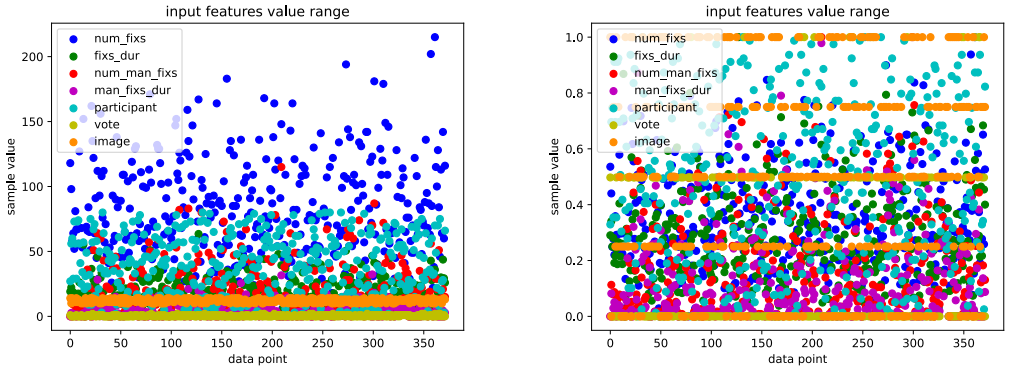
and applied for class distribution and stimuli distribution both give us a value of 0.999 which is sufficiently close to 1, and can therefore be assumed to be balanced [13]. With this calculation, input re-sampling is not required.

Feature Selection Previously, in [9], feature selection was conducted for inputs believed to either be of no use to the network or else were already accounted for by other features. For instance, eye gaze fixation was not encoded based on location of the image, but instead frequency of focus and time elapsed in those regions of interest. Thus, this "image" feature was removed entirely. The removal of these superfluous features was undertaken in [9] as they were thought to have not helped the network learn or may in fact hinder the performance of the network.
In our investigation, features like "vote", "participant_ID", and "image" was retained, as we believed there could be some hidden pattern between how a particular participant voted along with their eye gaze data for a particular image and the true state of the image (digitally modified or not). Thus, we retained all seven input features with the intent to discover whether the deeper model could discover patterns in the data and predict image veracity ("image_manipulated") based on otherwise useless features.

Normalisation Normalisation ensures that the network does not have an unfounded bias towards a particular by virtue of its value range [10]. The eye gaze data had features whose ranges had vastly different magnitudes. We therefore normalised the data by its z-score,

$$Z = \frac{x - \mu}{\sigma}$$

and Fig 2 showcases the difference this normalisation makes for the dataset.



(a) Pre-Normalisation (b) Post-Normalisation

Fig. 2: Data Normalisation

Note that for certain features like "image" and "vote" are not continuous values, hence the need to normalise them as either one or the other is important so that the model can have a better idea of recognising differences between values. This explains why these features occupy compartmentalised values as seen in Fig 2b.

1.3 Dataset Split

To optimise for both dataset size and model evaluation, we opted for k-fold Cross-Validation for a 80/20 train/test split of our data. That is, we employed k-fold for 80% of our data, and set aside the remaining 20% for results and reporting. The k-fold splitting enables well-curated hyperparameter tuning and selection, as enumerated in Section 2.2. After this process, we can then report model's expected accuracy on the remaining 20%, where results can be analysed and discussion of our deep learning model's ability can occur free of bias, as in Section 3.

k-fold Cross-Validation This technique calls for a rolling train/test approach, where for each of the k iterations, the model is exposed to and then evaluated on some proportion of the overall data. Typically used for situations with especially unbalanced class or stimuli distributions, we employ k-fold to assure our hyperparamater is as even as possible. Based off [7] we opt for a $k = 5$ shown in Fig. 3, as we have only 372 datapoints of which 20% are used for reporting of final results. Note that all work to derive our deep network's structure and design in Section 2.2 uses an averaged accuracy and loss across $k = 5$ folds.

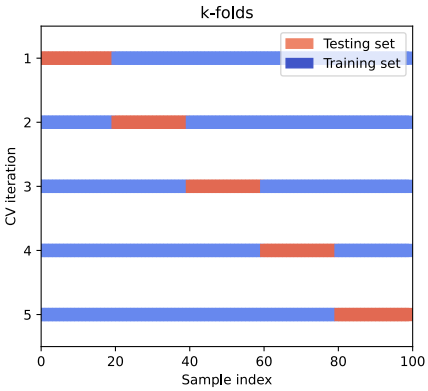


Fig. 3: k-fold visualisation

2 Method

2.1 Neural Pruning Technique

The neural pruning technique we have employed is known as the "Gedeon Method". This method relies on computing hidden neuron activation vector angles, where an activation vector is the sum of all activations over the input pattern space, and the angle is the vector angle between the activation vectors of two hidden neurons [4]. Once you have a table of these activation vector angles, the idea is to prune neurons based on a set of angle rules [4]. The subsequent actions for two neurons i and j tested in this paper are based on [5], and are:

$$action(i,j) = \begin{cases} prune\_one, & \text{if } angle(i,j) \leq 15° \\ prune\_both, & \text{if } angle(i,j) \geq 165° \\ nothing, & \text{otherwise} \end{cases} \tag{2}$$

where i and j represent the activation vectors for neurons i and j, *prune_one* defined as the pruned neuron has its weights added to the remaining neuron, and *prune_both* defines both neurons being removed.
The vector angle formula used in this paper differs from [4], and is:

$$angle(i,j) = \arccos \frac{V_i * V_j}{\|V_i\| * \|V_j\|} \tag{3}$$

Synthesising steps from [5,4], the process we followed for pruning a hidden layer's neurons during training was:

1. train for an epoch
2. compute activation vector angles to all other activation vectors in selected layer
3. for a given angle, prune according to (2)

It is noted that the above algorithm does not require re-training. This is because an angle which is $\geq 165°$ are complementary and cancel eachother out, thus removing both should have no effect on the network's accuracy [5]. When angles are $\leq 15°$ the weights of the pruned neuron is added to the remaining neuron, therefore information can be seen as being retained despite the neuron's deletion [5].

2.2 Network Design

As in [9], we opted for a feed-forward network trained via backpropagation of error measures. We wanted to first select the optimal hyperparameters for the non-pruned version and we did so through a series of training/validation k-fold iterations as described in Section 3. Firstly, we used some formulas from literature to determine appropriate amount of layers of our deep network. Then, we ran tests to determine epoch and optimiser hyperparameters, comparing the results to the shallow network from [9].

Hidden Layer Properties Literature for deep network hyperparameter selection expresses the importance of selecting appropriate values for the size per and amount of hidden layers [8]. For the express purpose of this deep network investigation, a fairly arbitrary choice of three layers were selected to represent the hidden units. The amount of neurons in each hidden layer was constrained by the following formula:

$$N_h = \frac{N_s}{\alpha \times (N_i + N_o)} \tag{4}$$

N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

provided by assertions from [3]. With an α value of 5 selected, the number suggested by the formula is about 5.5 neurons per layer. This value mediated the decision to create a physically convergent overall network structure (Fig 4), where hidden layers later in the network received less neurons than those earlier on.

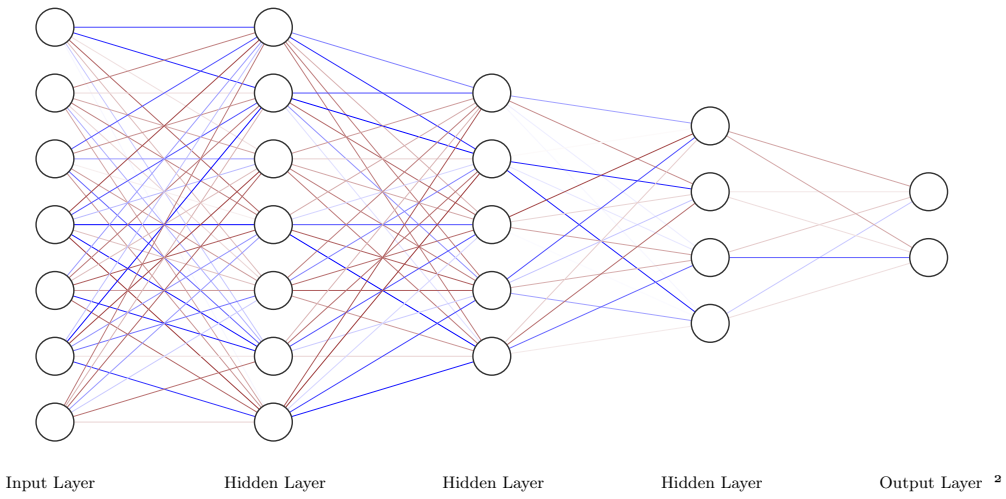
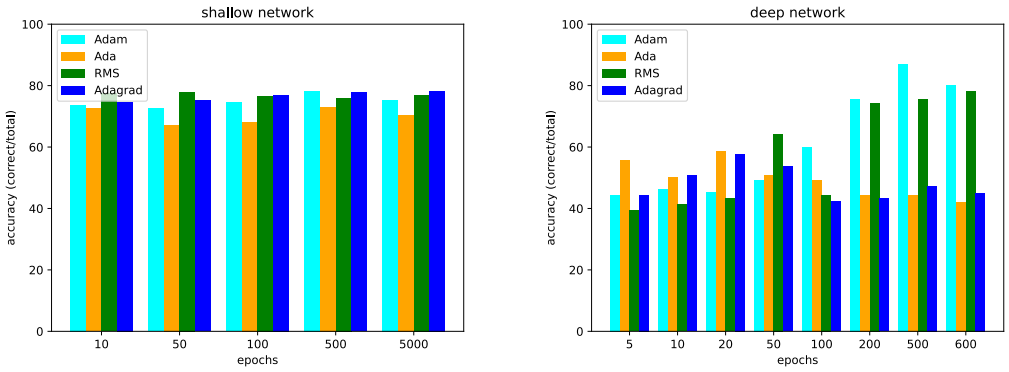


Fig. 4: deep network structure

Epoch and Optimiser Tests After selection of these basic hyperparameters, we opted to select our epoch and optimiser hyperparameters by running a series of tests with our training data. Fig 5 showcases the results of these tests.



(a) (b)

Fig. 5: hyperparameter tests, unpruned

The Adam optimiser for 500 epochs consistently outperformed any other optimiser x epoch combination and thus was selected as the hyperparameters for the deep network.
Our deep network thus has the following hyperparameters:

1. is fully-connected
2. is 3 hidden layers (with input and output layer), ie **7:7:5:4:2**
3. has a learning rate of **0.01**
4. is trained over **500 epochs**
5. uses the **Adam optimiser** function, as available in Pytorch
6. uses the **Cross Entropy Loss** function, as available in Pytorch

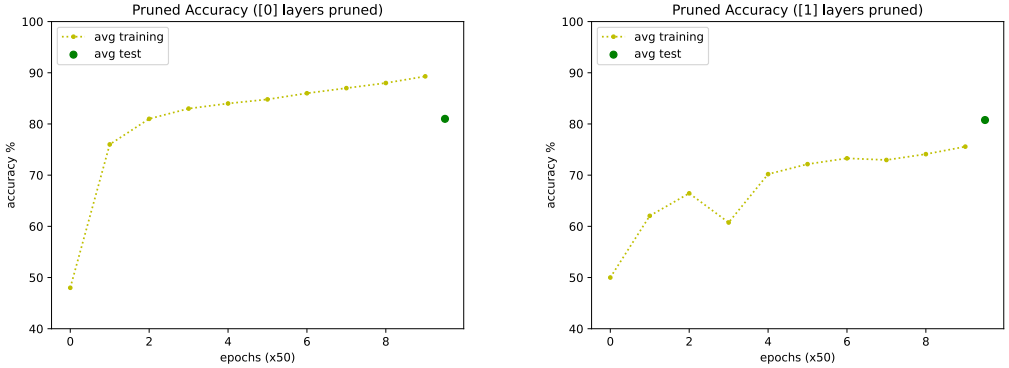
with Fig 4 visualising this network layout.

2.3 Experiment Design

The selected hyperparameters did not change across deep and shallow networks. This was to allow effective analysis both on the pruning process during training, and to evaluate the accuracy and generalisation of the deep network's performance given the test set. Both networks also used the same pre-processed data, with the exception of the deeper network which used 3 additional (7 total) features. Training and testing is done with the same 80% and 20% split of data respectively, for both models. We treat the shallow network as the control and the deep network as the experimental, and akin to [9], the results for training and testing are averaged.

3 Results and Discussion

The test results showed that the deep network accuracy was about the same as the shallow network's for pruning taking place only on the second hidden layer (shown as hidden layer's index). This test accuracy is shown as the green dot in both plots, with 81.2% being the shallow network's and 82.1% being the deep network's. These results are graphed in Fig 6:



(a) shallow (b) deep

Fig. 6: Shallow vs Deep

We note the vast difference between the shallow and deep models' training and testing data. The shallow model in Fig 6a had a training accuracy that, for at least half of training, "dominates"[2] the final testing accuracy. Conversely, the training accuracy of the deeper model in Fig 6b more steadily climbs but remains below the final testing accuracy throughout training. We attribute this train/test difference improvement to be the combination of the distinctiveness pruning and the general behaviour of deep networks.
The distinctiveness pruning technique which we conducted was similar to the popular "dropout"[3] method but for two reasons:

[2] dominate here merely refers to being a greater accuracy
[3] "dropout" here randomly selecting neurons to be removed from the network during a few training cycles

1. pruned neurons were selected according to their distinctiveness rather than at random
2. these neurons did not re-join the network or contribute to predictions again

For both pruning and dropout, neurons are "taught" not to rely on other similar neurons and instead be more self-reliant [6]. This is especially critical for a deep (more than 1-2 hidden layer) network, where the multiple layers of hidden units are prone to surmise patterns over the training data that do not truly exist. Using techniques such as pruning encourages robustness of the network and thus also improves its generalisation [11].

4 Conclusion and Further Work

While test set accuracy between shallow and deep pruned networks are mostly the same for a particular pruning regime, the ability for the model to generalise was far better in the deeper network than for the shallow one. Despite being a more impressionable network by virtue of its depth [1] the pruned network variants could consistently resist overfitting as they were able to remove non-distinct and therefore overly coexistent neurons.
Furthermore, the deep model appeared to be no better at resolving patterns from the extra features which were unavailable to the shallow model. This could be due to there not being any relevant or important data for the task of predicting image veracity, where the shallow network's input feature selection is more logical [9]. As for investigations with this dataset and network structure, deep learning research and anecdotes of improved performance with more hidden layers likely applies better to that of convolutional or recurrent models with data better suited and more applicable to those structures.
Thus to extend this investigation, future work with this dataset would be to assert a more practical and applied selection process for hyperparameters, and more crucially, experimenting with different data pre-processing techniques for preparation with a deep network. For instance, a data rasterisation technique to generate image data from this non-image dataset in conjunction with a convolutional network structure may achieve better accuracy. Papers such as [8] discuss this possibility using dimensionality reduction to generate image data with strictly non-image data. Furthermore, considerations of using test patterns (separate to that of the test set) to signal training termination would remove the process of epoch selection, further reducing the potential to "overtrain-by-training".

References

1. Benuwa, B., Zhan, Y., Ghansah, B., wornyo, D., Banaseka, F.: A review of deep machine learning. International Journal of Engineering Research in Africa **24**, 124–136 (06 2016). https://doi.org/10.4028/www.scientific.net/JERA.24.124
2. Caldwell, S., Gedeon, T., Jones, R., Copeland, L.: Imperfect understandings: A grounded theory and eye gaze investigation of human perceptions of manipulated and unmanipulated digital images (308) (July 2015)
3. Demuth, H.B., Beale, M.H., De Jess, O., Hagan, M.T.: Neural Network Design. Martin Hagan, 2nd edn. (2014)
4. Gedeon, T.D.: Data mining of inputs: Analysing magnitude and functional measures. International Journal of Neural Systems **08**(02), 209–218 (1997). https://doi.org/10.1142/s0129065797000227
5. Gedeon, T., Harris, D.: Network reduction techniques. Proceedings International Conference on Neural Networks Methodologies and Applications (1991)
6. Geirhos, R., Temme, C.R.M., Rauber, J., Schütt, H.H., Bethge, M., Wichmann, F.A.: Generalisation in humans and deep neural networks (2020)
7. Jung, Y., Hu, J.: K-fold averaging cross-validation procedure. Journal of nonparametric statistics **27** (2015)
8. Sharma, A., Vans, E., Shigemizu, D., Boroevich, K.A., Tsunoda, T.: Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. Scientific reports **9**(1), 1–7 (2019)
9. Smith, P.: Efficacy of neural network reduction by distinctiveness of hidden neurons. ABCs ANU Bio-inspired Computing conference **4** (05 2021)
10. Sola, J., Sevilla, J.: Importance of input data normalization for the application of neural networks to complex industrial problems. Nuclear Science, IEEE Transactions on **44**, 1464 – 1468 (07 1997). https://doi.org/10.1109/23.589532
11. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014), `http://jmlr.org/papers/v15/srivastava14a.html`
12. Wei, Q., Dunbrack, R.L.: The role of balanced training and testing data sets for binary classifiers in bioinformatics. PLoS ONE **8**(7) (2013). https://doi.org/10.1371/journal.pone.0067863
13. Zighed, D.A., Ritschard, G., Marcellin, S.: Asymmetric and sample size sensitive entropy measures for supervised learning. Advances in Intelligent Information Systems Studies in Computational Intelligence p. 27–42 (2010). https://doi.org/10.1007/978-3-642-05183-8_2