# GROUP 2220: Investigation of Data Deprivation, Data Augmentation and Complexity of a Deep Neural Network when applied to a Binary Classification Problem

Daniel Jordan, Jake Jackson, Theivan Pasupathipillai, and Marco Lorenzetti

(Dated: March 20, 2022)

A deep neural network is developed that obtains an accuracy of 0.9931 on a binary classification problem. The model is deprived of data. It is still able to classify correctly more than 90% with only 60 data samples for training. When only 50 training samples are available data augmentation can increase accuracy from 0.8813 to 0.9237. Two additional deep neural networks are developed for a more complex non linear binary classification problem. One has a larger structure than the other. They achieve accuracy's of 0.949 and 0.957 respectively. The last layer of the larger one is pruned. The mean decrease in accuracy is shown to be less than 0.006 with each neuron pruned.

## INTRODUCTION

A deep neural network(DNN) is trained on a binary classification problem. It is then optimised using a grid search over multiple hyper parameters. The ability of the model to learn the problem when deprived of data is investigated. The technique of data augmentation is used as a possible solution to data deprivation. To further investigate the performances of the DNN, it is trained on a more complex data-set. The complexity of the network is then simplified while maintaining the same performance.

The problem is whether or not a randomly generated point on a 2D plane resides within a set of three linear boundaries. The neural network has a two neuron input layer. One for each position co-ordinate. It has multiple hidden layers. Each hidden layer has the same amount of neurons. Each neuron in every layer has the same dropout rate. The other hyper parameters considered are shown in the table below. The activation function for the final output layer is a sigmoid function. This is standard for binary classification. Four thousand random points are generated and labeled 1 if within all 3 boundaries and 0 if outside a boundary. A grid search is performed over each hyper parameter individually. This is repeated to gain a twice optimised model. This optimised model was trained with data of various sizes. This allows the models performance to be investigated when deprived of data. Data augmentation is then used on small data set that deprived the model. An augmented data point is created by adding a random shift to a genuine data point. The augmented point may cross the boundary. Although its label remains unchanged. It is investigated whether or not augmented data increases or decreases the performance of the model.

| Hyper Parameter | Values Tested |
|---|---|
| Hidden Layers | 1, *2*, **4**, 8 |
| Neurons per Layer | 2, 5, 10, ***20*** |
| Activation Functions | *relu*, sigmoid, **elu**, soft sign |
| Drop Rate | **0**, *0.2*, 0.4, 0.8 |
| Optimizer | SGD, **RMSprop**, Adagrad, Adadelta, *Adam*, Adamax, Nadam |

TABLE I. The various possible hyper parameter values for the neural network model. The initial values tried are in *italic.* The best performing parameters found are in **bold.**

```
x = np.random.uniform(-50,50,size = (N,2))
```

Listing 1: Numpy's random number generation function.

```
grid = GridSearchCV(estimator=model_grdsearch,
                    param_grid=paramTest)
```

Listing 2: Scikit Learn Grid Search Function

To further improve the analysis on DNN, a binary classifier for a more complex data-set is sought. A data-set is generated in which the border between classes is given by a non-linear function. Two DNN are found that perform equally but with different architectural complexities(number of neurons and layers). The study concludes with how many neurons of the last layer of the complex DNN are relevant to maintain the same range of accuracy.

## METHODOLOGY

The model requires data to be trained. Four thousand random points are generated using the code in listing 1.

If these points satisfied the boundary conditions in expression 1,

$$x_1 > -20, \quad x_2 > -40, \quad x_1 + x_2 < 40, \tag{1}$$

then the corresponding label is 1, otherwise 0. This data is split with an 80:20 ratio into a training and validation set. The training data is presented in figure ??. The training data is normalised before it is used for training. This is known to improve the learning speed and reliability of the final model.

This data is used for training and optimisation. The parameter values in table I are to be tested in a grid search using the code in listing 2.

A full grid search over everything shown in table I is very computationally demanding. Thus the model is only partially trained at each grid point test. It is trained for the number of epochs the guessed model (table I) required to reach 0.9 validation accuracy. A full grid search is still not feasible. Starting with the guessed model, each parameter is altered one at a time. The best value is kept and the model is adjusted for the next parameter search. This is repeated for each parameter twice to achieve a twice optimised model. This best found twice optimised model is used in the rest of the investigation.

In order to investigate the effect of data size on the fit, a single large data set of 8000 points (4000 training, 4000 validation) was split into multiple smaller sections. The model was subsequently independently trained on each section. The accuracy of each model was measured by the success of predicting the labels of the 4000 point validation set. In addition, a linearly spaced grid of predicted values was plotted and overlaid with the input data and the original distribution.

Then the augmentation analysis is performed. The fundamental aspect of this phase is the choice of the new data generating function and the fraction of data to be shifted. The fractions chosen for this analysis are 0.25, 0.5 and 0.75. The augmented data is obtained using: $x_i + \alpha \cdot w_i$. Where $x_i$ is the original data co-ordinate, $\alpha$ is a random value between $-0.5$ and 0.5. The $w$ is the variance of the original data-set. The last part of the analysis concerns a new two class data-set **??**. In this case the border between the classes is defined by a non linear function 2.

$$r(\theta) = 10 + 3\,sin(10\,\theta) \qquad \theta \in [0, 2\pi]$$

With a grid-search similar to the one described before, we define a complex DNN that is able to solve our classification problem. The complexity of our model is defined as number of layers and number of neurons per layer. This model is obtained by not allowing the number of neurons per layer to change. This is an arbitrary choice taken for gaining computational time during grid-search. We will try to prune our complex DNN to find a simpler one that can performs similarly. We drop the constraint on the number of neurons per layer, trying to decrease the complexity of the DNN. After some tries we succeeded in finding one simpler DNN with good performance see table II.

The complex and simple DNN have a similar accuracy. Therefore the accuracy dependence on DNN architecture is studied. Accuracy is defined as the fraction of correct predictions on the validation data-set. Starting from the trained complex DNN, random neurons of the last layer are deactivated. This is achieved by manually setting their weights to zero. The model in not retrained. The accuracy of the DNN is evaluated as the number of active last layer's neurons increases. This operation is repeated

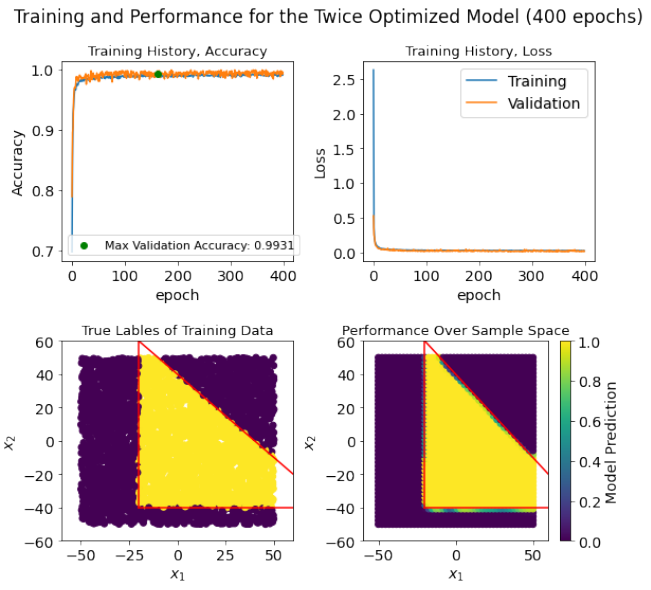

FIG. 1. A collection of demonstrations of the final models performance when trained with the 3200 data points shown.

100 times. The mean and standard deviation of the accuracy is calculated. This is plotted against the active neurons of the last layer. This is only performed on the last layer to save computation time.

| Hyper Parameter | Complex DNN | Simple DNN |
|---|---|---|
| Layers | (50,50,50,50,50) | (20,15,5,3) |
| Activation Functions | elu | elu |
| Optimizer | Nadam | Nadam |
| Weight Initialization | glorot uniform | he normal |
| Accuracy's Achieved | 0.949 | 0.957 |

TABLE II. DNN architectures taken into account

## RESULTS

### Optimisation

The best twice optimised model is performance tested. It is trained on 3200 data points for 400 epochs. At each epoch the training and validation error is calculated and plotted in figure 1. The validation error is greater than the training error and so there is no over-fitting of the data. The highest validation accuracy achieved over 400 epochs is 0.9931. Figure 1 depicts the performance over the entire sample space. The model accurately classifies the positions.

### Data Size

Increasing data size in smaller data sets between 0 - 150, rapidly improves the fit accuracy as demonstrated in the zoom plot of 3. This indicates that the model was

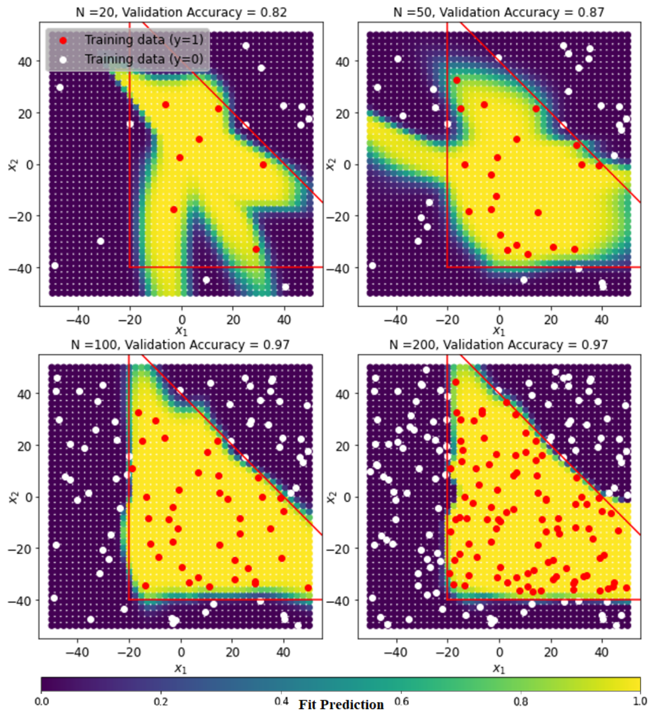

FIG. 2. 4000 Point data prediction grid with the input labeled training data overlayed for various data sizes

too data deprived to determine the underlying distribution. This is shown in the upper proportion of figure 2 where the fit prediction has large spaces spanned without training data. This problem leads to the shown erroneous regions of incorrect label carry through.

As training data size increases the fit prediction begins to accurately constrain the underlying distribution. The distinct region of the triangle is identified with above 90% validation accuracy as few as 60 training data points. Once the key area is well defined the model begins to to plateau as in figure 3. The remaining variation along the edges of the triangle represent a small part of the total plot area and these minor adjustments make small difference in accuracy. This is shown in the lower section of figure 2 where they accuracy remains the same despite doubling the data size.

## Augmentation Analysis

As anticipated in the introduction, augmentation is useful for compensating the lack of data during the fit of a neural network. In this analysis a data size of 50 points was used to train the twice optimized DNN. The model achieves an accuracy of about 0.88 over the large 4000 point validation set. These fifty points are augmented and the model is re-trained. Figure 4 shows the performance's over the entire sample space.

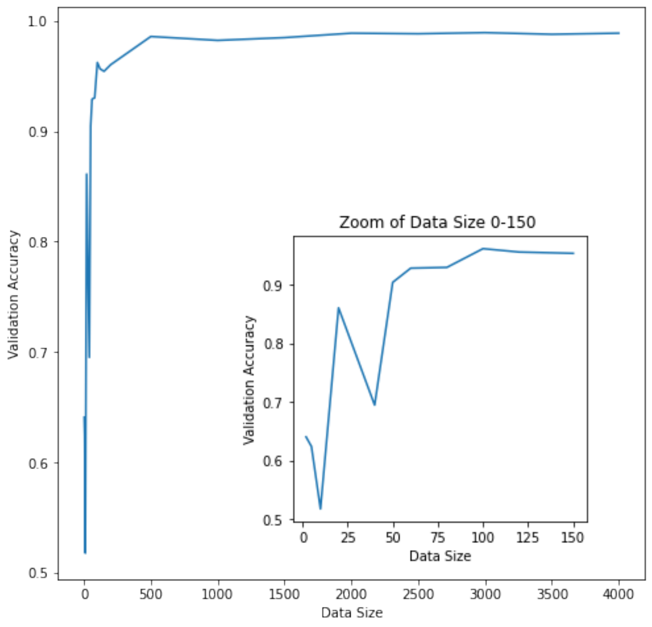

FIG. 3. The variation of the accuracy of the model prediction on the validation data against the size of the data set.

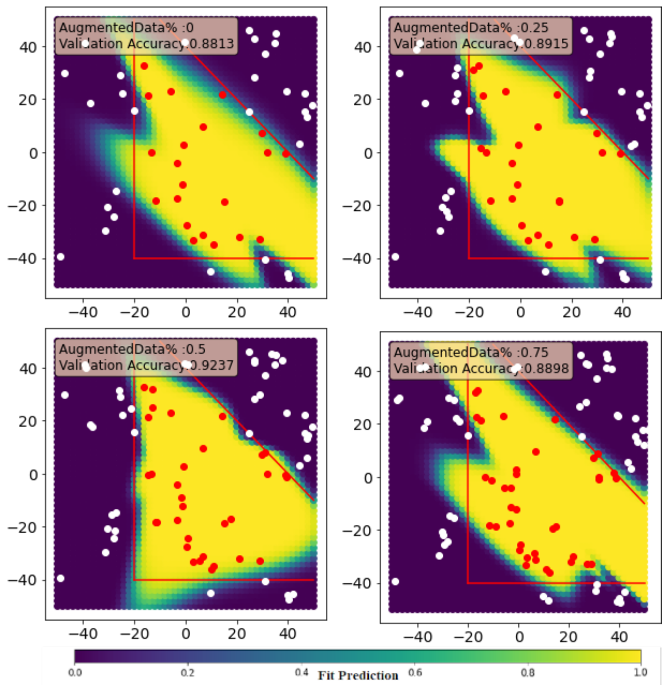

FIG. 4. The variation of the accuracy of the model prediction on the validation data against the percentages of augmented data with $w = std(x)$.

The validation accuracy with no augmentation is measured to be 0.8813. Each degree of augmentation allows the model to perform better than with no augmentation. The best result is an accuracy of 0.9237 when 50% of the data is augmented.

## New Data-set and Architecture Optimization

The two models in II are trained to solve the more complex binary classification problem described by ??. Their performances are evaluated on the entire sample space, figure 5. It is clear that both DNN's can learn the border between classes represented by the orange line. Accuracy's of 0.949 and 0.957 are achieved respectively for the complex and simple DNN.

Studying the performance of the complex DNN whilst varying the number of active neurons in the last layer is the next step of the analysis. Figure 6 illustrates that the mean value of the accuracy decreases with decreasing number of active neurons. This is expected. It is worth noticing that the slope of the change is not steep, meaning that we could cut the number of neurons without a significant decrease in performance. The standard deviation of the mean accuracy is shown by the width of the orange strip in figure 6. It is inversely proportional to the number of active neurons. By taking a weighted linear regression on the mean values of the accuracy we find a slope of 0.0054. The r value is 0.967, showing the trend is linear.

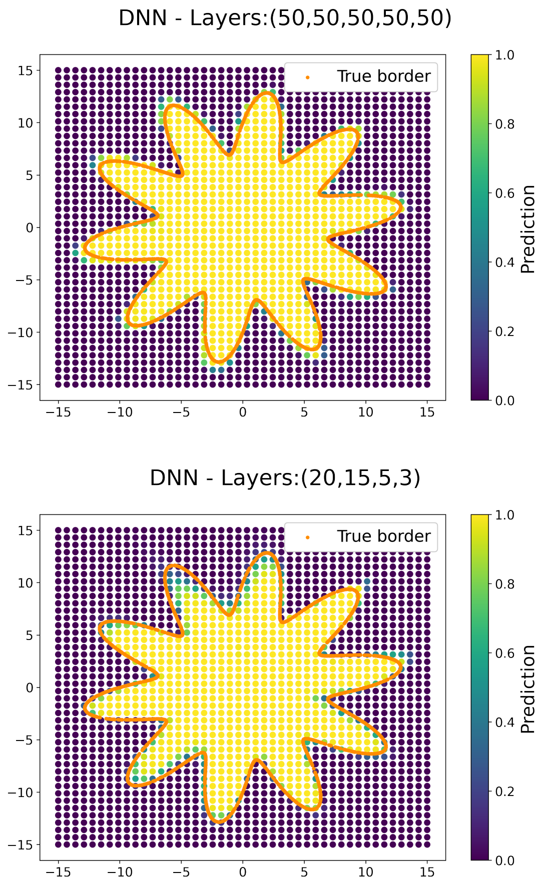

FIG. 5. Predictions on completely sampled space.

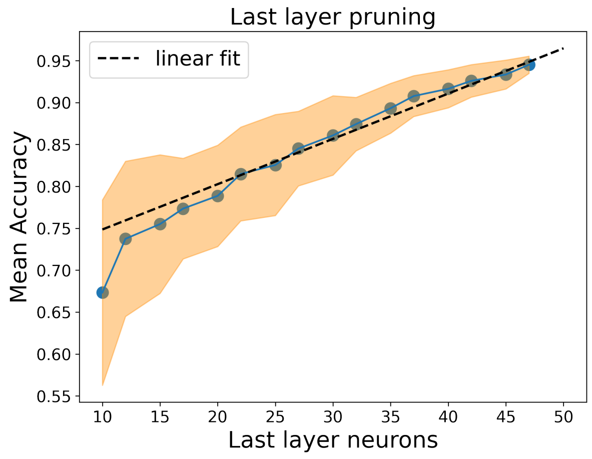

FIG. 6. Evaluation of the accuracy of the complex DNN by pruning the last layer.

## CONCLUSIONS

A guessed neural network model is twice optimised using a partial grid search. When trained the twice optimised model solved the binary classification problem presented with a validation accuracy of 0.9931. The neural network was successful in establishing an accuracy of 90% in a restricted data size of 60 training points. Below this the model often struggles to find the underlying distribution that produced the data, but this is to be expected. The fit preformed well overall, reaching a maximum accuracy of 98 % and was consistently effective on larger data sets.

It has been shown that data augmentation can lead to an improvement in DNN accuracy. With a small amount of data the model struggles to learn the underlying distribution. The model over fits this data. When the augmentation shift is small an augmented data point is likely stay in the same classification region as the original point. The augmented data can cover a region of space that was previously unavailable to the model. This can lead to an improvement in accuracy. For a data size of 50 the accuracy is improved from 0.8813 to 0.9237, when 50% of the data is augmented.

The simple and complex DNN shown in II perform with similar accuracy's on the non linear boundary condition 5. This indicates that the complex model is over complex and a waste of resources.

The accuracy's dependence on the number of active neurons in the last layer is studied. It is clear that when deactivating some neurons the change in accuracy is not very steep, see figure 6. So if for instance we take out one neuron from the last layer, the mean decrease in accuracy will be less than 0.6% according to the slope of the linear regression.

## CONTRIBUTIONS

The entire group contributed to every part but the sections were finalised as so. Daniel finalised the initial grid search optimisation. Jake finalised the data size investigation. Marco Finalised the data augmentation section. Theivan produced the non linear classification boundary and DNN pruning parts of the report.