Case Study 6: Particle Detection

Matt Farrow

July 18, 2022

1 Introduction

The goal of this study is to develop a dense neural network that maximizes accuracy when detecting the particle.

2 Methods

## 2.1 Data Examination

An initial examination of the data revealed 7,000,000 observations and 29 features, including label, the response variable, f0 – f26, and mass. There is no missing data, and the response variable is almost exactly weighted between the two classes (Table 1).

|  |  |
| --- | --- |
| Class | Count |
| 1.0 | 3,500,879 |
| 0.0 | 3,499,121 |

Table : Response Variable Count

The only work that was done prior to building the model was to convert the response variable, label, to an integer.

## 2.2 Model Preparation & Execution

The data was split into test and training data sets using a 90/10 split with a stratified shuffle and a MinMaxScaler was applied. To set a baseline for the neural network’s performance, I first ran a model using logistic regression to determine how it performed on the data. That model produced an accuracy score of 83.7%.

Two neural networks were built for this project in order to explore how different parameters affected the final accuracy performance. The first network was comprised of an input layer, two dense layers with rectified linear unit (relu) activation functions, and a final dense layer with a ‘sigmoid’ activation (Figure 1).

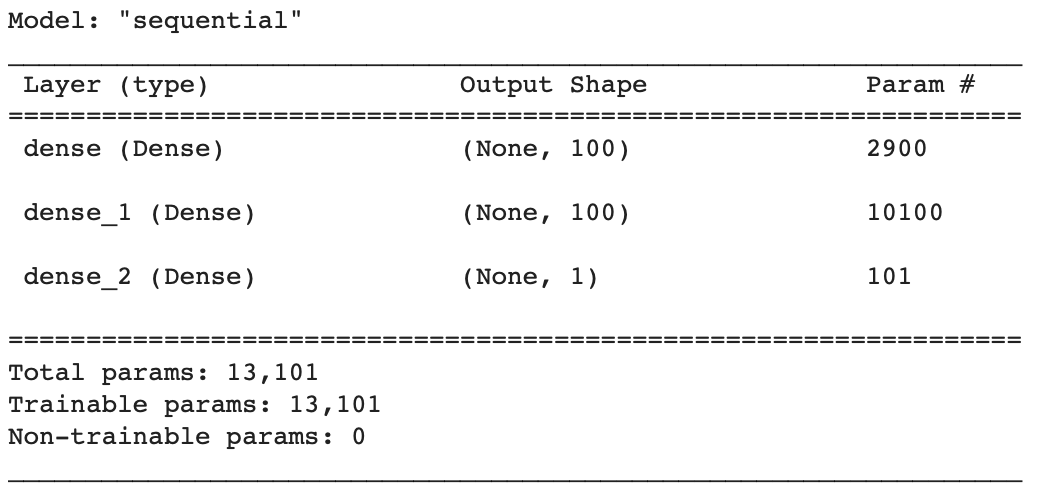


Figure 1: Model 1 Summary

Finally, the model was compiled using a binary cross-entropy loss function with an ‘adam’ optimizer. This network was fit using batch sizes of 10,000 and 1,000 epochs.

The second neural network alternated dense and dropout layers in an effort to prevent overfitting and ran with batch sizes of 1,000 and 100 epochs.

Table

Description automatically generated

3 Results

## 3.1 Model 1

In plotting the loss of the training and testing data, I noticed a relatively smooth decrease in the training loss (Figure 2). The testing loss follows a similar slope, but there was much more variation among the epochs.



Figure : Loss Over Epochs

The classification report for the first neural network shows the model performed with an accuracy of 88%. This result was similar to other variations of batch size and epochs (Figure 3).

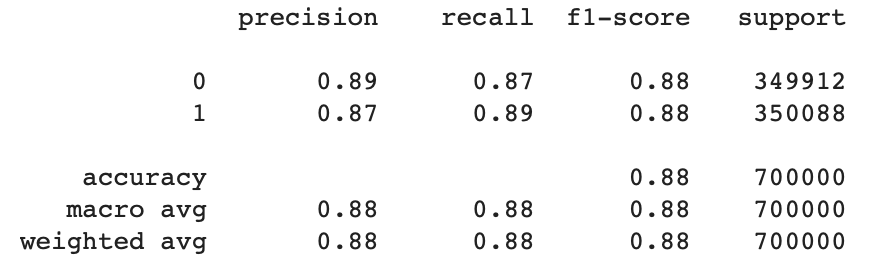


Figure : Model 1 Classification Report

The confusion matrix shows where misclassifications occurred. There were almost 37,000 false positives and 46,000 false negatives (Figure 4).

Chart, treemap chart

Description automatically generated

Figure : Model 1 Confusion Matrix

## 3.2 Model 2

The classification report for the second neural network shows the model performed with an accuracy of 83% (Figure 5). In this case the use of the dropout layers decreased the model’s performance.

Table

Description automatically generated

Figure : Model 2 Classification Report

The confusion matrix shows where misclassifications occurred. The greatest change from the first model is the significant increase in false positives–almost 77,000–an increase of almost 40,000 (Figure 6).

Chart, treemap chart

Description automatically generated

Figure : Model 2 Confusion Matrix

4 Conclusion

In conclusion, the first neural network, with its’ relatively simple structure performed the best in terms of accuracy. That said, the second neural network ran much more quickly, albeit with a notable drop in accuracy. If I were to redo the second neural network in the future, I’d explore the parameters of the dropout layers to see if an increase in accuracy could be achieved while keeping the model’s speed performance.

# Appendix

## Code

Code begins on the following page.