DS7333 Quantifying the World: Case Study 6­­

Rick Fontenot, Feby Thomas Cheruvathoor

1. **Introduction**

­

The goal for this case study is to build a classification model using a dense neural network to distinguish between a signal process which produces Higgs bosons and a background process which does not.

The goal is to maximize the accuracy and describe the decisions made while designing and building the network as well as determining when the model has sufficiently finished training.

1. **Methods**

**Data description**

This HIGGS dataset originates from the UCI Machine Learning repository (<https://archive.ics.uci.edu/ml/datasets/HIGGS>) which included 11,000,000

observations. This study specifically uses a similar dataset provided to the SMU Data Science DS7333 class and contains 7,000,000 observations.

There are 28 features that can be used to make predictions that are labeled anonymously as f0-f26 as well as “mass”. The UCI resources describe these features as 21 low level features that are kinematic properties measured by the particle detectors in the accelerator and 7 high level features that are functions of the first 21 features derived by physicists to help discriminate between the two classes. For the purposes of this case study, specific domain knowledge and understanding of the physics will not be part of the analysis. The focus is on construction of the neural network and classification performance metrics.

The target variable “label” has 2 distinct categories:

0 (for background process): 3,499,121 observations

1 (for signal which produces Higgs bosons): 3,500,879 observations

**Processing data & Feature Creation**

Class Imbalance:

The two classes of the classification target are balanced on the raw dataset, so no resampling or other actions were taken regarding class weights, other than stratified splitting to maintain balance between modeling sets.

Duplicates:

There were no duplicate observations in the dataset that required removal

Missing Values:

There were no missing values in the dataset that required imputing or removal

Categorical Encoding:

All available features are numeric, so no features were encoded.

Standardization and Scaling:

The features of the raw dataset are generally centered at a mean of zero with the exception of mass which ranges from 500 to 1500. All features including mass were scaled using the sklearn StandardScaler tool prior to modeling.

**Training Split & Cross Validation Strategy**

One key approach to training neural networks is determining when a model has reached optimal training. Continuing to train based on decreasing loss for the training set itself could result in overfitting.

In order to train, tune, and determine training stop points, 10% of the data was split into a validation set (stratified shuffled data to get random observations with similar proportion of records in the target classes).

An additional 10% of the dataset was split into a test set for an unbiased measure of accuracy after model tuning. The remaining 80% of the dataset is used for tuning. (both of these splits also utilized stratified shuffled data to get random observations with similar proportion of records in the target classes).

Due to the large size of the training set (5.6 million observations) we strategized that the cost (training time) would outweigh the benefits to performance. Rather than using cross validation to address risk of overfitting, drop layers and L2 regularization were evaluated.

**Random Forest modeling (Model 1)**

A random forest model using primarily default parameters for the sklearn RandomForestClassifier was created as an initial benchmark for further model comparisons. Due to the size of the dataset, the maximum depth was reduced from none to 5, and the number of estimators was reduced from 100 to 50 so that training time was feasible in the timeline available for this study. No grid searches or hyperparameter tuning was performed. The intent of this model was to check accuracy and training time on a simple training model that the neural network models can be compared against.

**Simple Base Neural Network (Model 2)**

As a starting point for comparison to the random forest model, a first neural network was built using a single hidden layer with 300 neurons, the relu activation function, and adam optimizer with a batch size of 5,000 and 5 epochs. All other parameters such as learning rate were left to defaults, and no regularization such as L2 or drop layers were included. No grid searches or hyperparameter tuning was performed. The intent of this model was to check accuracy and training time on a quick neural network to compare to the random forest as well as set a benchmark for further neural nets to compare against.

**Initial Neural Network based on Research (Model 3)**

To further develop the neural network model, we studied previous research on higgs boson deep learning including “Searching for Exotic Particles in High-energy Physics with Deep Learning.” (Baldi et al., 2014). Utilizing insights from this paper, the next neural net was built with 5 hidden layers, with 300 neurons per layer, the hyperbolic tangent activation function along with the SGD optimizer and learning rate of 0.05. No regularization such as L2 or drop layers were included. Moving beyond the simple base neural network, this model serves as the starting point for tuning hyperparameters on the final model.

**Tuned Neural Network (Model 4)**

Expanding beyond model 3, a dropout layer was added following each of the 5 dense layers to evaluate effects of regularization. For feasibility with time available for this study, searches for hyperparameter tuning were performed in stages. Training is based on monitoring validation set metrics, all accuracies reported in following searches are validation accuracy (not training or test accuracy):

Stage-1 Hyperparameter Search:

The initial search utilized the HParams dashboard tool with TensorFlow’s TensorBoard. A batch size of 5000 with 5 epochs was chosen for it’s relatively fast training time to search for the optimal combination of activation function and optimizer. All comparisons in this initial search were built with 300 neurons per hidden layer and a dropout rate of zero. The table below summarizes the specific optimizer and activations evaluated, and the gelu activation combined with the adam optimizer was found to produce the highest accuracy.



Figure 1: Performance Summary for each model in stage-1 hyperparameter search

Stage-2 Hyperparameter Search:

Focusing on the gelu activation and adam optimizer, a search for the optimal learning rate was performed. This search also used the same batch size of 5000 with 5 epochs built with 300 neurons per hidden layer and a dropout rate of zero. The table below summarizes the search range and a learning rate of 0.001 was found to produce the highest accuracy.



Figure 2: Performance Summary for each model in stage-2 hyperparameter search

Stage-3 Hyperparameter Search:

Based on the first two hyperparameter searches, tuning for regularization via dropout rate was performed utilizing the gelu activation and adam optimizer with a learning rate of 0.001. This search moved beyond the 5 epochs and allowed each model to train until an early stopping criteria of validation accuracy with a minimum delta of 0.01% and patience of 3 epochs. For this search the batch size was constant at 5,000 similar to prior models. The table below summarizes the search range and a dropout rate of 0.2 was found to produce the highest accuracy.



Figure 3: Performance Summary for each model in stage-3 hyperparameter search

Stage-4 Hyperparameter Search:

The first three searches were all performed with a batch size of 5,000 so that many models across parameter searches could be trained and evaluated within a reasonable time. With all other hyperparameters set to optimal values based on searches 1-3, a final search was performed for the optimal batch size where training times may take significantly longer. Similar to the last stage each model was allowed to train until an early stopping criteria of validation accuracy with a minimum delta of 0.01% and patience of 3 epochs.

Reducing Batch size to less than 250 drastically increased training time. While the theory was smaller batch sizes would increase accuracy, that was not observed in this search. The table below summarizes the search range and a batch size of 5,000 was found to produce the highest accuracy.



Figure 4: Performance Summary for each model in stage-4 hyperparameter search

Final Model design:

Based on the hyperparameter tuning searches above, the final tuned model was built with the following design:

callback = tf.keras.callbacks.EarlyStopping(

monitor='val\_accuracy',

min\_delta=0.0001,

patience=3,

verbose=0,

mode='auto',

baseline=None,

restore\_best\_weights=True)

gelu\_model = tf.keras.Sequential([

tf.keras.layers.Dense(300, activation='gelu', input\_shape=(28,)),

tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(300, activation='gelu'),

tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(300, activation='gelu'),

tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(300, activation='gelu'),

tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(300, activation='gelu'),

tf.keras.layers.Dropout(0.2),

tf.keras.layers.Dense(2, activation='softmax')])

gelu\_model.compile(optimizer=tf.optimizers.Adam(learning\_rate=0.001),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=['accuracy'])

gelu\_model.fit(train\_features\_scaled, train\_labels, batch\_size=5000, epochs=100, callbacks=callback, validation\_data=[validation\_features\_scaled, validation\_labels])

1. **Results**

The random forest benchmark model took 101 minutes to train with an accuracy of 82.4%. Even the simple 1-layer neural net was able to achieve similar accuracy with just a 1 minute training time. Increasing the neural network to 5 layers based on the reference paper insights increased accuracy to 86.45% with just 2 minutes of training time.

The final tuned neural network provided the best accuracy of 88.7% and while training time of 29 minutes is more than the initial neural network models, it’s still very reasonable training time. This model was chosen as the preferred model to study results further.



Figure 5: Performance Summary for each model considered

**Tuned Neural Network results**

The confusion matrix and classification reports show model predictions perform well for both classes with similar f1 scores in each class:

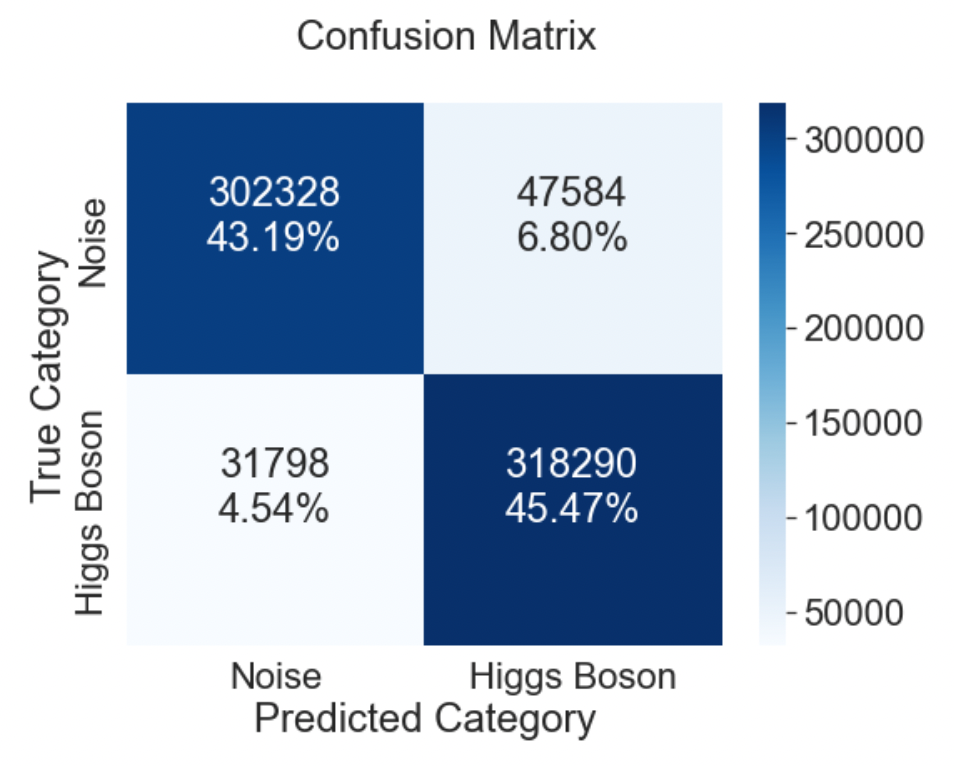


Figure 6: Confusion Matrix for the Tuned NN predictions

Classification Report

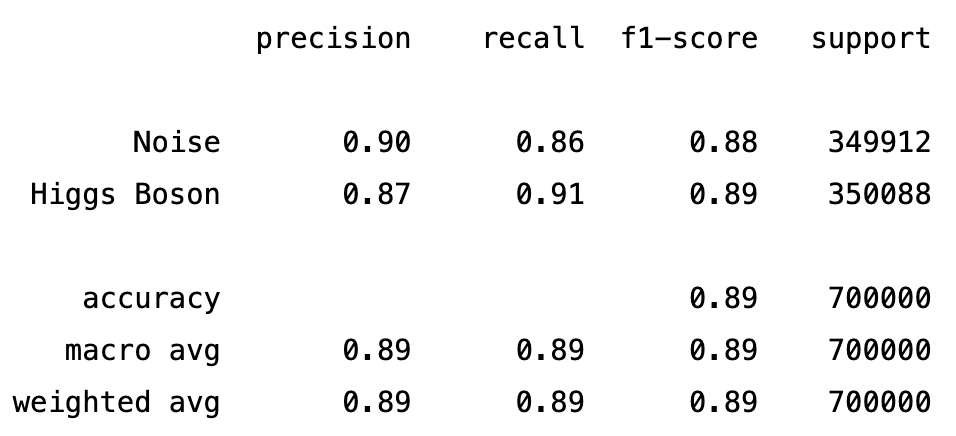


Figure 7: Classification Report for the Tuned NN predictions

Probability discriminant thresholds were evaluated and the standard 0.5 cutoff produced the maximum accuracy.

1. **Conclusions**

With 88.7% accuracy this deep learning model provides a useful tool in searches for exotic particles. Neural network models appear to be an appropriate model for this challenge compared to a random forest.

This study also demonstrates the importance of the neural network design and hyperparameters and the significant effect they can have on prediction accuracy.

The results of the optimal batch size for training were not in-line with expectations. While initial searches for hyperparameters were performed with a consistent batch size of 5,000 for the time constraints, further study would prove interesting if more time allows. It’s possible that starting with a small batch size of 32 or 64 for all parameter searches could identify a different network design that outperforms the currently tuned model.

**References**

1. Baldi, P., P. Sadowski, and D. Whiteson. “Searching for Exotic Particles in High-energy Physics with Deep Learning.” Nature Communications 5 (July 2, 2014)

**Appendix**

1. **Code**

A rendered notebook containing code for the base analysis can be accessed at:

<https://nbviewer.org/github/rickfontenot/QTW/blob/main/Case%20Study%206/case6_rick.ipynb>