Searching for the Higgs Boson with Deep Learning

November 16, 2020

Paul Adams, Stuart Miller, and Justin Howard

1 Introduction

A Higgs boson is an elementary particle produced by quantum application of energy within the Higgs field, which can be used to explain why particles have mass. Because the process responsible for creating bosons was previously unknown to the researchers, the researchers generated sample process signals using Monte Carlo simulations to help build a model that can classify if a signal process is likely to produce a boson or if the signal process is only likely to produce noise. Monte Carlo simulations are useful for studying the property of tests when the assumptions they are derived from are not met. For this reason, the researchers applied this method. Because deep learning is useful for gaining inference from massive amounts of data when there are only very minor differences responsible for class separation and the Monte Carlo simulations generated 11 million signals, the researchers decided to model the signals using a deep neural network. Altogether, the research performs classification on 28 total features; 21 are kinetic properties and an additional seven are derived from functions of those properties, which are used to discriminate between the two classes. The two classes are signals processes that create Higgs bosons and signal processes that do not. In this project, we deconstruct the research and reconstruct the modeling performed. We then analyze the methods used and determine if a more useful approach exists given the enhancement of technology available since the original research concluded in 2014.

2 Methods

2.1 Data

The dataset used in this case study is a set of Monte Carlo simulations of signals for processes that produce Higgs bosons and background processes that do not produce Higgs bosons[1]. The dataset contains 11 million instances of 28 features: 21 kinimatic properties measured by particle deterors in the accelerator and 7 engineered features. The target variable is a binary indicator where 1 indicates a Higgs process and 0 indicates a background process. The reference paper indicates that the last 500,000 instances in this training set were used for model validation. We maintained this train-validation split in this case study, using the last 500,000 instances for validation and the prior instances for training.

[1]https://archive.ics.uci.edu/ml/datasets/HIGGS

2.2 Neural Network

2.2.1 Replication of Model

In this case study, we replicated the modeling performed by Baldi, Sadowski, & Whiteson on the Higgs boson dataset with deep neural networks (DNN). The model used in the reference study was a 5-layer multi-perceptron (MLP) with `tanh` activation, a weight decay ($L2$ regularization) coefficient of $1x10^{-5}$, and layers initialized with weights from the random normal distribution. These hyperparamters are summerized in table 1.

Table 1. Model Architecture Hyperparameters

Parameterized Object	Node Count	Activation	Weight Initialization	Weight Decay
Layer 1	300	tanh	random normal ($\mu = 0, \sigma = 0.1$)	$L2$ regularization, $1x10^{-5}$
Layer 2	300	tanh	random normal ($\mu = 0, \sigma = 0.05$)	$L2$ regularization, $1x10^{-5}$
Layer 3	300	tanh	random normal ($\mu = 0, \sigma = 0.05$)	$L2$ regularization, $1x10^{-5}$
Layer 4	300	tanh	random normal ($\mu = 0, \sigma = 0.05$)	$L2$ regularization, $1x10^{-5}$
Layer 5	300	tanh	random normal ($\mu = 0, \sigma = 0.001$)	$L2$ regularization, $1x10^{-5}$

In addition to the model architecture, we also replicated the training process. The model was trainined with stochastic gradient descent (SGD) with a batch size of 100. The learning rate was initialized at 0.05 and decreased by a factor of 1.0000002 on each batch to a minumum rate of $1x10^{-6}$. The momentum was initialized to 0.9 and increased linearly to 0.99 over 200 epochs, remaining constant after the 200th epoch. For the stopping criterion, the reference paper indicates that early stopping with minimum change in error of 0.00001 over 10 epochs was used to determine when to stop the training process (resulting in training the model over 200-1000 epochs). However, the reference paper does not indicate what error metric was monitored for early stopping. We monitored the validation with binary cross-entropy loss for early stopping as is typical practice in deep learning. The training process is summarized in Table 2. In this study, the model was implemented with `TensorFlow`[2] (version 2.2.0) because the framework used in the reference paper, `PyLearn2`[3], is no longer actively maintained.

[2]https://pypi.org/project/tensorflow/2.2.0/
[3]http://deeplearning.net/software/pylearn2/

Table 2. Model Training Parameters

Training Method	Value
Optimizer	SGD
Loss	binary cross-entropy
Initial Learning Rate	0.05
Learning Rate Decay	Decreased on each batch by a factor of 1.0000002
Minumum Learning Rate	$1x10^{-6}$
Initial Optimizer Momentum	0.9
Momentum Increase	0.00045 (epoch) + 0.9
Maximum Optimizer Monentum	0.99
Early Stopping	Minimum decrease of 0.00001 validation loss over 10 epochs

2.2.2 Suggestions for Model Improvement

Since the reference paper was published in 2014, there have been a number of notable advancements in the field of deep learning. In this section, we suggest five potential improvements for this model and discuss the expected impact each improvement would provide.

- relu activation
- dropout
- ResNets (skip connections)
- optimizers: Adam, RMSprop
- layer initialization

Rectified linear unit (ReLu) activation functions, defined as f(x) = max(0, x), are a computationally efficient means of improving gradient propagation across layers. The ReLu function simplifies neural netowrk outputs and encourages sparse activations by limiting the number of active neurons to only those with positive values. Another variant of the ReLu is the Gaussian Error Linear Unit (GELU). GELU serves as the default activation function for layers in Google's popular BERT model for Natural Language Processing. GELU applies a cumulative distribution function to network outputs. The decision to use either ReLu or GELU can be the result of further experimentation.

Dropout layers, which randomly silence a user-defined percentage of nodes in the network, serve as a means of limiting overfitting. Dropout strengthens the influence of neurons that are most effective at reducing the model's error and serve to reduce the relative distance between the validation and training error.

Residual connections, introduced in the computer vision world in the ResNet neural network, smooth the loss landscape of neural networks. As neural networks become deep, neural loss landscapes quickly transition from being nearly convex to being highly chaotic. Adding or concatentating the output of a previous layer to the output of a later layer carries forward the information learned in previous layers. This practice, combined with batch normalization, permits the deepening of neural networks.

The current network uses SGD as an optimization method, but also modifies the momentum over time. New optimizers, such as Adam and RMSProp, are capable of performing these modifications automatically to speed up the model's training. The Root Mean Square Propagation (RMSProp) algorithm applies a parameter specific learning rate decay across all parameters. The Adaptive Moment Estimation (Adam) algorithm takes RMSprop a step further by accounting for the cumulative history of gradients over time. The Adam optimizer is effective at speeding up the training of deep networks and is widely used as an optimizer of first choice in the deep learning community.

3 Results

We trained this network on the data 30 times to assess whether the performance reported in the reference paper was captured in our results. A typical training curve, shown in Fig 1, indicates that the model is overfitting to the training data and could have been stopped earlier. This overfitting is a result of the stopping criterion: momentum at maximum and loss does not improve over 10 epochs. Note that since monemtum is linearly increase over 200 epochs and the monemtum at maximum is part of the stopping criterion, the network was trained for at least 200 epochs before stopping. However, the training history in Fig 1, indicates that the training could have been stopped around 50 epochs. In this case, since the validation loss appears flat after the minimum loss, the overfitting is not highly concerning as we would expect the validation performance to be consistent on other unseen data.

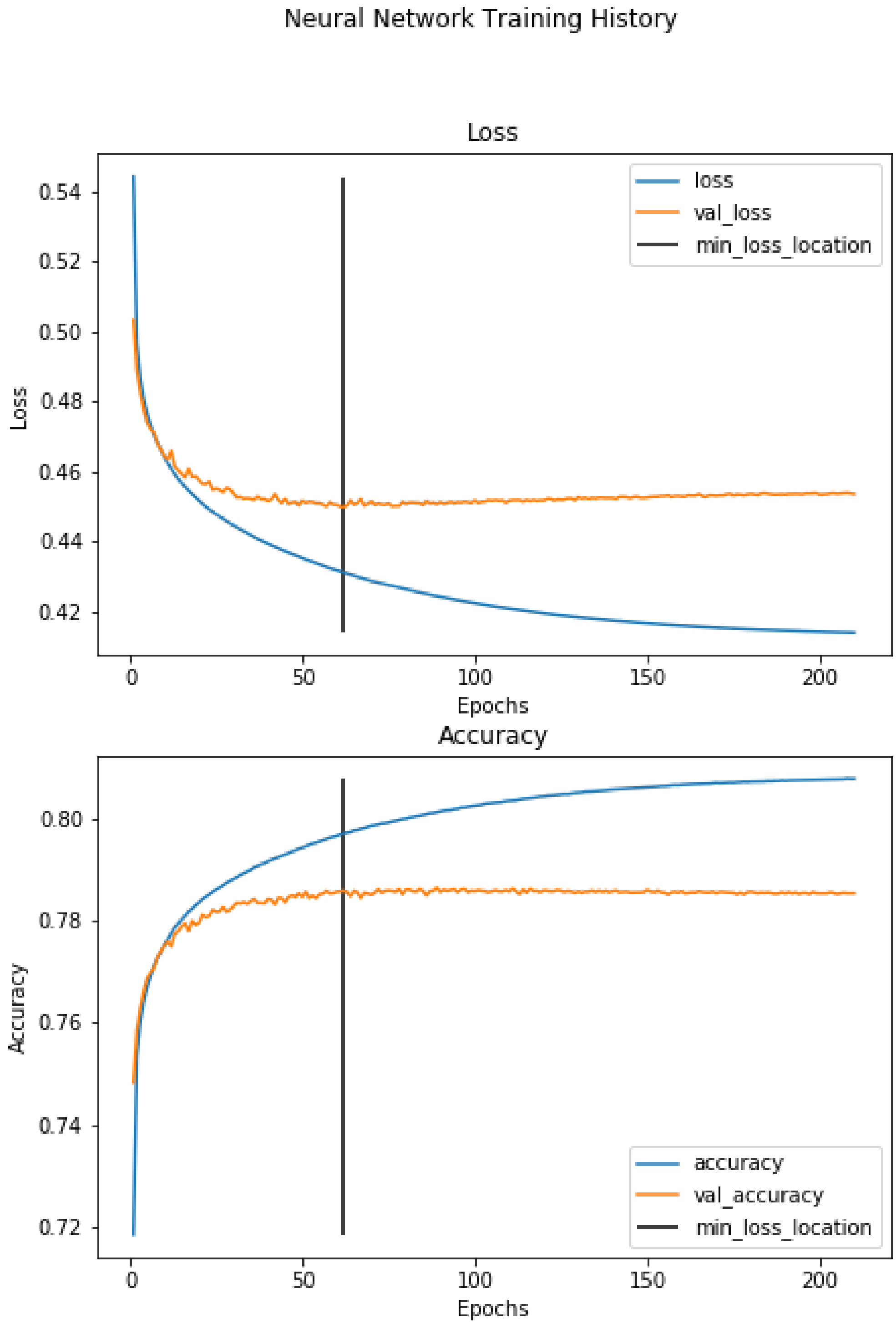

Figure 1: Training History of the neural network model.
The training curves show evidence that the network is fully trained.

The ROC curve of the model performance on the test set is shown in Fig 2. The ROC curve shows relatively balanced true positve rate performance and false positive rate performance. The area under the curve (AUC) is 0.871.

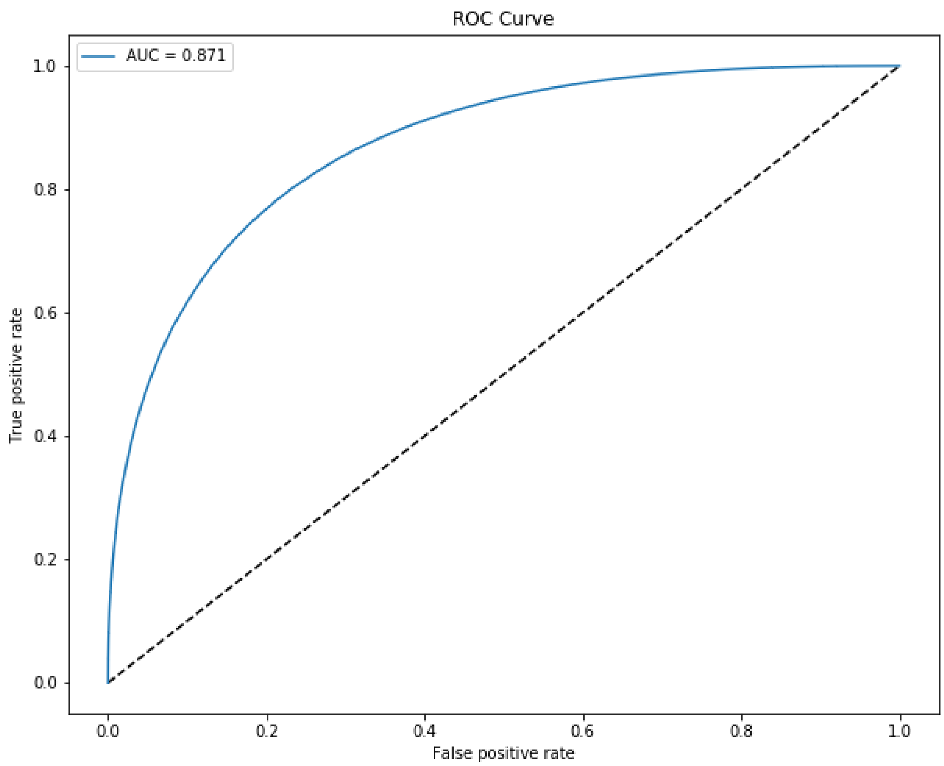

Figure 2: ROC Curve for Model Predictions on Test Dataset

In order to evaluate whether we were able to replicate the reference paper's results, we trained the model 30 times to create a distribution of results. The distribution of AUC values, shown in Fig 3, has a mean of 0.8726 with a standard deviation of 0.001. The reference paper reported a mean AUC of 0.885 with a standard deviation of 0.002. Given our distribution of results, it appears our results were inconsistent with the results of the reference paper. Additionally, a one sample t-test comparing the mean of the reference paper to our results, indicate strong evidence the results are inconsistent.

Table 3. Results of Statistical Comparison

t-Statistic	p-value
-58.80	$1.02x10^{-31}$

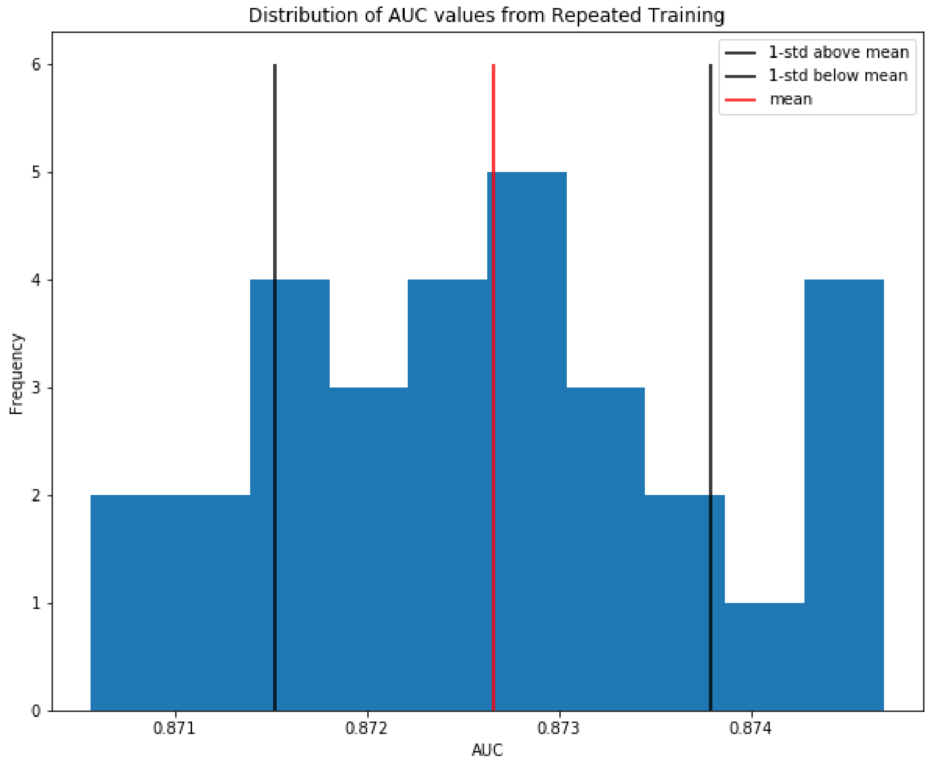

Figure 3: Histogram of AUC values from training the neural network model 30 times with different random seeds.

4 Conclusions

As evidenced through the research and findings, the original model was very useful in detecting whether or not a signal process was likely to generate a Higgs boson. However, we were unable to construct a replicate model using the researchers specifications to derive the same findings. While our findings were similar, studentized hypothesis testing across 30 epochs indicated the model we created was not able to produce the same results. As neural network technology evolves, parameterization is modified and added in the default packages. We believe this is largely the reason for the inconsistent results between our model and the novel model used in the original research. Through repeated application, we were also able to identify some over-fitting in the original model in addition to the lack of providing the metrics used, such as the loss metric used in validation, which we assumed to be binary cross-entropy. Regardless, the original research was successful in its mission to predict what inputs result in the production of Higgs bosons.

5 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). https://arxiv.org/pdf/1402.4735.pdf

A Code

[9]:
```
import pickle

import pandas as pd
import numpy as np
from scipy.stats import ttest_1samp
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve
from sklearn.metrics import auc as auc_score

seed = 42
```

[ ]:
```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras import layers
from tensorflow.keras import optimizers
from tensorflow.keras import initializers
from tensorflow.keras import callbacks
from tensorflow.keras import backend as K
from tensorflow.keras.regularizers import l2
print(tf.__version__)

auc_score = tf.keras.metrics.AUC()
```

[2]:
```
data = pd.read_csv('./data/HIGGS.csv', header = None)
# from the paper: The last 500,000 examples are used as a test set.
# The first column is the class label (1 for signal, 0 for background), followed␣
↪by the 28 features
train_test_split = data.shape[0] - 500000
X_test = data.iloc[ train_test_split : , 1: ]
y_test = data.iloc[ train_test_split : , 0 ]
X_train = data.iloc[ : train_test_split , 1: ]
y_train = data.iloc[ : train_test_split , 0 ]
```

[3]:
```
# helper functions

def create_model(hidden_size,
                 first_layer_init,
                 hidden_layer_init,
                 output_layer_init,
                 weight_decay,
                 starting_lr,
                 metrics,
                 ):
    """Create the model used in this case study
```

```
    """
    model = Sequential([
        layers.Dense(hidden_size, activation='tanh',
                     kernel_initializer=first_layer_init,
                     kernel_regularizer=l2(weight_decay),
                     dtype='float64'
                     ),
        layers.Dense(hidden_size, activation='tanh',
                     kernel_initializer=hidden_layer_init,
                     kernel_regularizer=l2(weight_decay),
                     dtype='float64'
                     ),
        layers.Dense(hidden_size, activation='tanh',
                     kernel_initializer=hidden_layer_init,
                     kernel_regularizer=l2(weight_decay),
                     dtype='float64'
                     ),
        layers.Dense(hidden_size, activation='tanh',
                     kernel_initializer=hidden_layer_init,
                     kernel_regularizer=l2(weight_decay),
                     dtype='float64'
                     ),
        layers.Dense(hidden_size, activation='tanh',
                     kernel_initializer=hidden_layer_init,
                     kernel_regularizer=l2(weight_decay),
                     dtype='float64'
                     ),
        layers.Dense(1, activation='sigmoid',
                     kernel_initializer=output_layer_init,
                     kernel_regularizer=l2(weight_decay),
                     dtype='float64'
                     )
    ])

    model.compile(optimizer=optimizers.SGD(lr=starting_lr),
                  loss='binary_crossentropy',
                  metrics=metrics)

    return model
```

[4]:

```
#We selected a five-layer neural
#network with 300 hidden units in each layer, a learning
#rate of 0.05, and a weight decay coefficient of 1 ÃŮ 10âĹŠ5

hidden_size = 300
starting_lr = 0.05
weight_decay = 1e-6
```

```
#Hidden units all used the tanh activation function.
#Weights were initialized from a normal distribution with
#zero mean and standard deviation 0.1 in the first layer,
#0.001 in the output layer, and 0.05 all other hidden layers.
#Gradient computations were made on mini-batches
#of size 100.

first_layer_init = initializers.RandomNormal(
    mean=0.0, stddev=0.1, seed=seed
)
hidden_layer_init = initializers.RandomNormal(
    mean=0.0, stddev=0.05, seed=seed
)
output_layer_init = initializers.RandomNormal(
    mean=0.0, stddev=0.001, seed=seed
)
```

[109]:

```
model = create_model(hidden_size,
                     first_layer_init,
                     hidden_layer_init,
                     output_layer_init,
                     weight_decay,
                     starting_lr,
                     metrics=['accuracy',
                           auc_score])
```

[110]:

```
# The learning rate decayed by a factor
#of 1.0000002 every batch update until it reached a minimum of 10âĹŠ6
class LRSchedule(callbacks.Callback):
    """Lower the learning rate by a factor of 1.0000002 until the learning
    rate is at 1e-6 at which point it should remain constant
    """
    def on_batch_end(self, batch, logs):
        current_lr = K.get_value(model.optimizer.lr)
        if current_lr > 1e-6:
            lr = current_lr / 1.0000002 #1.00002
            K.set_value(self.model.optimizer.lr, lr)
        else:
            K.set_value(self.model.optimizer.lr, 1e-6)

lr_scheduler = LRSchedule()
```

[111]:

```
#A momentum term increased linearly over
#the first 200 epochs from 0.9 to 0.99, at which point it
#remained constant.
```

```python
class MomentumSchedule(callbacks.Callback):
    """Update the momentum linearly from 0.9 to 0.99 between epochs 1 and 200
    """
    def on_epoch_end(self, epoch, logs=None):
        starting_value = 0.9
        ending_value = 0.99
        number_epochs = 200
        step_increase = (ending_value-starting_value) / number_epochs
        if epoch > number_epochs:
            K.set_value(self.model.optimizer.momentum, ending_value)
        else:
            current_momentum = K.get_value(self.model.optimizer.momentum)
            current_momentum += step_increase
            K.set_value(self.model.optimizer.momentum, current_momentum)

momentum_scheduler = MomentumSchedule()
```

[112]:

```python
#Training ended when the momentum had reached its maximum value
#i.e. need to run for 200 epochs at least before stopping
#and the minimum error on
#the validation set (500,000 examples) had not decreased
#by more than a factor of 0.00001 over 10 epochs

class EarlyStoppingAfterMinEpoch(callbacks.Callback):
    """Stop training when a monitored metric has stopped improving and minimum␣
 ↪number epochs has been reached.
    Primarily borrowed from:
    https://github.com/tensorflow/tensorflow/blob/v2.3.1/tensorflow/python/keras/
 ↪callbacks.py#L1559-L1690
    """
    def __init__(self,
                 monitor='val_loss',
                 min_delta=0,
                 patience=0,
                 min_epoch=200,
                 verbose=0,
                 mode='auto',
                 baseline=None,
                 restore_best_weights=False):
        super(EarlyStoppingAfterMinEpoch, self).__init__()

        self.monitor = monitor
        self.patience = patience
        self.min_epoch = min_epoch
        self.verbose = verbose
        self.baseline = baseline
        self.min_delta = abs(min_delta)
```

```
        self.wait = 0
        self.stopped_epoch = 0
        self.restore_best_weights = restore_best_weights
        self.best_weights = None

        if mode not in ['auto', 'min', 'max']:
            logging.warning('EarlyStopping mode %s is unknown, '
                            'fallback to auto mode.', mode)
        mode = 'auto'

        if mode == 'min':
            self.monitor_op = np.less
        elif mode == 'max':
            self.monitor_op = np.greater
        else:
            if 'acc' in self.monitor:
                self.monitor_op = np.greater
            else:
                self.monitor_op = np.less

        if self.monitor_op == np.greater:
            self.min_delta *= 1
        else:
            self.min_delta *= -1

    def on_train_begin(self, logs=None):
    # Allow instances to be re-used
        self.wait = 0
        self.stopped_epoch = 0
        if self.baseline is not None:
            self.best = self.baseline
        else:
            self.best = np.Inf if self.monitor_op == np.less else -np.Inf
        self.best_weights = None

    def on_epoch_end(self, epoch, logs=None):
        current = self.get_monitor_value(logs)
        if current is None:
            return
        if (epoch - self.min_epoch + self.patience) > 0:
            if self.verbose > 0:
                print('Started Monitor for Earing Stopping')
            if self.monitor_op(current - self.min_delta, self.best):
                self.best = current
                self.wait = 0
                if self.restore_best_weights:
                    self.best_weights = self.model.get_weights()
```

```
            else:
                self.wait += 1
                if self.wait >= self.patience:
                    self.stopped_epoch = epoch
                    self.model.stop_training = True
                if self.restore_best_weights:
                    if self.verbose > 0:
                        print('Restoring model weights from the end of the best␣
 ↪epoch.')
                    self.model.set_weights(self.best_weights)

    def on_train_end(self, logs=None):
        if self.stopped_epoch > 0 and self.verbose > 0:
            print('Epoch %05d: early stopping' % (self.stopped_epoch + 1))

    def get_monitor_value(self, logs):
        logs = logs or {}
        monitor_value = logs.get(self.monitor)
        if monitor_value is None:
            logging.warning('Early stopping conditioned on metric `%s` '
                          'which is not available. Available metrics are: %s',
                          self.monitor, ','.join(list(logs.keys())))
        return monitor_value

early_stopping_criterion = EarlyStoppingAfterMinEpoch(
    monitor='val_loss',
    min_delta=0.00001,
    patience=10,
    verbose=1
)
```

[113]:
```
hist = model.fit(X_train, y_train,
                 batch_size=100,
                 epochs=400,
                 validation_data=(X_test, y_test),
                 callbacks=[
                     lr_scheduler, momentum_scheduler, early_stopping_criterion
                 ]
                )
```

```
Epoch 1/400
WARNING:tensorflow:Layer dense_6 is casting an input tensor from dtype float64
to the layer's dtype of float32, which is new behavior in TensorFlow 2.  The
layer has dtype float32 because it's dtype defaults to floatx.

If you intended to run this layer in float32, you can safely ignore this
warning. If in doubt, this warning is likely only an issue if you are porting a
```

```
TensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call
`tf.keras.backend.set_floatx('float64')`. To change just this layer, pass
dtype='float64' to the layer constructor. If you are the author of this layer,
you can disable autocasting by passing autocast=False to the base Layer
constructor.

105000/105000 [==============================] - 546s 5ms/step - loss: 0.5442 -
accuracy: 0.7185 - auc_2: 0.7957 - val_loss: 0.5033 - val_accuracy: 0.7483 -
val_auc_2: 0.8313
Epoch 2/400
105000/105000 [==============================] - 541s 5ms/step - loss: 0.4969 -
accuracy: 0.7533 - auc_2: 0.8360 - val_loss: 0.4888 - val_accuracy: 0.7585 -
val_auc_2: 0.8422
Epoch 3/400
105000/105000 [==============================] - 539s 5ms/step - loss: 0.4868 -
accuracy: 0.7601 - auc_2: 0.8437 - val_loss: 0.4824 - val_accuracy: 0.7628 -
val_auc_2: 0.8470
Epoch 4/400
105000/105000 [==============================] - 517s 5ms/step - loss: 0.4808 -
accuracy: 0.7640 - auc_2: 0.8481 - val_loss: 0.4775 - val_accuracy: 0.7662 -
val_auc_2: 0.8506
Epoch 5/400
105000/105000 [==============================] - 546s 5ms/step - loss: 0.4766 -
accuracy: 0.7669 - auc_2: 0.8512 - val_loss: 0.4738 - val_accuracy: 0.7687 -
val_auc_2: 0.8533
Epoch 6/400
105000/105000 [==============================] - 547s 5ms/step - loss: 0.4733 -
accuracy: 0.7690 - auc_2: 0.8535 - val_loss: 0.4721 - val_accuracy: 0.7698 -
val_auc_2: 0.8547
Epoch 7/400
105000/105000 [==============================] - 547s 5ms/step - loss: 0.4706 -
accuracy: 0.7710 - auc_2: 0.8555 - val_loss: 0.4712 - val_accuracy: 0.7706 -
val_auc_2: 0.8563
Epoch 8/400
105000/105000 [==============================] - 546s 5ms/step - loss: 0.4683 -
accuracy: 0.7726 - auc_2: 0.8572 - val_loss: 0.4677 - val_accuracy: 0.7728 -
val_auc_2: 0.8578
Epoch 9/400
105000/105000 [==============================] - 545s 5ms/step - loss: 0.4661 -
accuracy: 0.7738 - auc_2: 0.8587 - val_loss: 0.4663 - val_accuracy: 0.7739 -
val_auc_2: 0.8588
Epoch 10/400
105000/105000 [==============================] - 544s 5ms/step - loss: 0.4641 -
accuracy: 0.7752 - auc_2: 0.8602 - val_loss: 0.4642 - val_accuracy: 0.7751 -
val_auc_2: 0.8605
Epoch 11/400
```

```
105000/105000 [==============================] - 544s 5ms/step - loss: 0.4623 -
accuracy: 0.7764 - auc_2: 0.8615 - val_loss: 0.4633 - val_accuracy: 0.7758 -
val_auc_2: 0.8609
Epoch 12/400
105000/105000 [==============================] - 544s 5ms/step - loss: 0.4606 -
accuracy: 0.7775 - auc_2: 0.8626 - val_loss: 0.4659 - val_accuracy: 0.7749 -
val_auc_2: 0.8613
Epoch 13/400
105000/105000 [==============================] - 537s 5ms/step - loss: 0.4592 -
accuracy: 0.7786 - auc_2: 0.8636 - val_loss: 0.4613 - val_accuracy: 0.7774 -
val_auc_2: 0.8623
Epoch 14/400
105000/105000 [==============================] - 531s 5ms/step - loss: 0.4578 -
accuracy: 0.7794 - auc_2: 0.8646 - val_loss: 0.4603 - val_accuracy: 0.7778 -
val_auc_2: 0.8634
Epoch 15/400
105000/105000 [==============================] - 530s 5ms/step - loss: 0.4566 -
accuracy: 0.7802 - auc_2: 0.8654 - val_loss: 0.4592 - val_accuracy: 0.7788 -
val_auc_2: 0.8639
Epoch 16/400
105000/105000 [==============================] - 530s 5ms/step - loss: 0.4555 -
accuracy: 0.7809 - auc_2: 0.8662 - val_loss: 0.4582 - val_accuracy: 0.7794 -
val_auc_2: 0.8648
Epoch 17/400
105000/105000 [==============================] - 526s 5ms/step - loss: 0.4544 -
accuracy: 0.7817 - auc_2: 0.8670 - val_loss: 0.4607 - val_accuracy: 0.7779 -
val_auc_2: 0.8633
Epoch 18/400
105000/105000 [==============================] - 528s 5ms/step - loss: 0.4534 -
accuracy: 0.7822 - auc_2: 0.8677 - val_loss: 0.4584 - val_accuracy: 0.7799 -
val_auc_2: 0.8650
Epoch 19/400
105000/105000 [==============================] - 539s 5ms/step - loss: 0.4524 -
accuracy: 0.7830 - auc_2: 0.8683 - val_loss: 0.4586 - val_accuracy: 0.7791 -
val_auc_2: 0.8648
Epoch 20/400
105000/105000 [==============================] - 541s 5ms/step - loss: 0.4515 -
accuracy: 0.7835 - auc_2: 0.8690 - val_loss: 0.4575 - val_accuracy: 0.7795 -
val_auc_2: 0.8652
Epoch 21/400
105000/105000 [==============================] - 548s 5ms/step - loss: 0.4507 -
accuracy: 0.7842 - auc_2: 0.8695 - val_loss: 0.4564 - val_accuracy: 0.7811 -
val_auc_2: 0.8667
Epoch 22/400
105000/105000 [==============================] - 564s 5ms/step - loss: 0.4498 -
accuracy: 0.7846 - auc_2: 0.8701 - val_loss: 0.4563 - val_accuracy: 0.7806 -
val_auc_2: 0.8662
Epoch 23/400
```

```
105000/105000 [==============================] - 551s 5ms/step - loss: 0.4490 -
accuracy: 0.7851 - auc_2: 0.8706 - val_loss: 0.4570 - val_accuracy: 0.7809 -
val_auc_2: 0.8663
Epoch 24/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4484 -
accuracy: 0.7856 - auc_2: 0.8711 - val_loss: 0.4547 - val_accuracy: 0.7820 -
val_auc_2: 0.8675
Epoch 25/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4476 -
accuracy: 0.7861 - auc_2: 0.8716 - val_loss: 0.4550 - val_accuracy: 0.7816 -
val_auc_2: 0.8673
Epoch 26/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4470 -
accuracy: 0.7864 - auc_2: 0.8720 - val_loss: 0.4547 - val_accuracy: 0.7817 -
val_auc_2: 0.8673
Epoch 27/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4464 -
accuracy: 0.7869 - auc_2: 0.8725 - val_loss: 0.4540 - val_accuracy: 0.7826 -
val_auc_2: 0.8682
Epoch 28/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4457 -
accuracy: 0.7874 - auc_2: 0.8729 - val_loss: 0.4552 - val_accuracy: 0.7817 -
val_auc_2: 0.8677
Epoch 29/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4451 -
accuracy: 0.7877 - auc_2: 0.8733 - val_loss: 0.4549 - val_accuracy: 0.7820 -
val_auc_2: 0.8674
Epoch 30/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4445 -
accuracy: 0.7881 - auc_2: 0.8737 - val_loss: 0.4537 - val_accuracy: 0.7827 -
val_auc_2: 0.8682
Epoch 31/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4439 -
accuracy: 0.7885 - auc_2: 0.8741 - val_loss: 0.4524 - val_accuracy: 0.7834 -
val_auc_2: 0.8690
Epoch 32/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4434 -
accuracy: 0.7889 - auc_2: 0.8745 - val_loss: 0.4526 - val_accuracy: 0.7833 -
val_auc_2: 0.8689
Epoch 33/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4428 -
accuracy: 0.7893 - auc_2: 0.8749 - val_loss: 0.4521 - val_accuracy: 0.7836 -
val_auc_2: 0.8693
Epoch 34/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4422 -
accuracy: 0.7896 - auc_2: 0.8752 - val_loss: 0.4522 - val_accuracy: 0.7832 -
val_auc_2: 0.8691
Epoch 35/400
```

```
105000/105000 [==============================] - 578s 6ms/step - loss: 0.4417 -
accuracy: 0.7899 - auc_2: 0.8756 - val_loss: 0.4520 - val_accuracy: 0.7838 -
val_auc_2: 0.8695
Epoch 36/400
105000/105000 [==============================] - 578s 6ms/step - loss: 0.4411 -
accuracy: 0.7904 - auc_2: 0.8760 - val_loss: 0.4518 - val_accuracy: 0.7842 -
val_auc_2: 0.8697
Epoch 37/400
105000/105000 [==============================] - 581s 6ms/step - loss: 0.4406 -
accuracy: 0.7907 - auc_2: 0.8763 - val_loss: 0.4527 - val_accuracy: 0.7834 -
val_auc_2: 0.8691
Epoch 38/400
105000/105000 [==============================] - 575s 5ms/step - loss: 0.4402 -
accuracy: 0.7910 - auc_2: 0.8766 - val_loss: 0.4520 - val_accuracy: 0.7834 -
val_auc_2: 0.8697
Epoch 39/400
105000/105000 [==============================] - 522s 5ms/step - loss: 0.4397 -
accuracy: 0.7912 - auc_2: 0.8769 - val_loss: 0.4520 - val_accuracy: 0.7833 -
val_auc_2: 0.8695
Epoch 40/400
105000/105000 [==============================] - 513s 5ms/step - loss: 0.4393 -
accuracy: 0.7916 - auc_2: 0.8772 - val_loss: 0.4514 - val_accuracy: 0.7842 -
val_auc_2: 0.8701
Epoch 41/400
105000/105000 [==============================] - 510s 5ms/step - loss: 0.4388 -
accuracy: 0.7919 - auc_2: 0.8775 - val_loss: 0.4519 - val_accuracy: 0.7840 -
val_auc_2: 0.8699
Epoch 42/400
105000/105000 [==============================] - 509s 5ms/step - loss: 0.4384 -
accuracy: 0.7921 - auc_2: 0.8778 - val_loss: 0.4534 - val_accuracy: 0.7831 -
val_auc_2: 0.8688
Epoch 43/400
105000/105000 [==============================] - 527s 5ms/step - loss: 0.4380 -
accuracy: 0.7923 - auc_2: 0.8781 - val_loss: 0.4517 - val_accuracy: 0.7844 -
val_auc_2: 0.8702
Epoch 44/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4375 -
accuracy: 0.7926 - auc_2: 0.8784 - val_loss: 0.4509 - val_accuracy: 0.7844 -
val_auc_2: 0.8703
Epoch 45/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4371 -
accuracy: 0.7929 - auc_2: 0.8786 - val_loss: 0.4521 - val_accuracy: 0.7836 -
val_auc_2: 0.8700
Epoch 46/400
105000/105000 [==============================] - 568s 5ms/step - loss: 0.4367 -
accuracy: 0.7932 - auc_2: 0.8789 - val_loss: 0.4506 - val_accuracy: 0.7846 -
val_auc_2: 0.8704
Epoch 47/400
```

```
105000/105000 [==============================] - 565s 5ms/step - loss: 0.4363 -
accuracy: 0.7935 - auc_2: 0.8792 - val_loss: 0.4510 - val_accuracy: 0.7849 -
val_auc_2: 0.8705
Epoch 48/400
105000/105000 [==============================] - 380s 4ms/step - loss: 0.4359 -
accuracy: 0.7937 - auc_2: 0.8794 - val_loss: 0.4512 - val_accuracy: 0.7846 -
val_auc_2: 0.8704
Epoch 49/400
105000/105000 [==============================] - 471s 4ms/step - loss: 0.4355 -
accuracy: 0.7941 - auc_2: 0.8797 - val_loss: 0.4502 - val_accuracy: 0.7853 -
val_auc_2: 0.8710
Epoch 50/400
105000/105000 [==============================] - 565s 5ms/step - loss: 0.4351 -
accuracy: 0.7942 - auc_2: 0.8799 - val_loss: 0.4515 - val_accuracy: 0.7846 -
val_auc_2: 0.8704
Epoch 51/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4347 -
accuracy: 0.7945 - auc_2: 0.8802 - val_loss: 0.4509 - val_accuracy: 0.7856 -
val_auc_2: 0.8707
Epoch 52/400
105000/105000 [==============================] - 570s 5ms/step - loss: 0.4344 -
accuracy: 0.7948 - auc_2: 0.8804 - val_loss: 0.4512 - val_accuracy: 0.7844 -
val_auc_2: 0.8707
Epoch 53/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4339 -
accuracy: 0.7951 - auc_2: 0.8807 - val_loss: 0.4511 - val_accuracy: 0.7849 -
val_auc_2: 0.8709
Epoch 54/400
105000/105000 [==============================] - 569s 5ms/step - loss: 0.4336 -
accuracy: 0.7952 - auc_2: 0.8809 - val_loss: 0.4508 - val_accuracy: 0.7844 -
val_auc_2: 0.8706
Epoch 55/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4332 -
accuracy: 0.7955 - auc_2: 0.8812 - val_loss: 0.4508 - val_accuracy: 0.7851 -
val_auc_2: 0.8711
Epoch 56/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4328 -
accuracy: 0.7957 - auc_2: 0.8814 - val_loss: 0.4504 - val_accuracy: 0.7853 -
val_auc_2: 0.8711
Epoch 57/400
105000/105000 [==============================] - 566s 5ms/step - loss: 0.4325 -
accuracy: 0.7959 - auc_2: 0.8816 - val_loss: 0.4499 - val_accuracy: 0.7858 -
val_auc_2: 0.8715
Epoch 58/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4322 -
accuracy: 0.7962 - auc_2: 0.8818 - val_loss: 0.4510 - val_accuracy: 0.7853 -
val_auc_2: 0.8708
Epoch 59/400
```

```
105000/105000 [==============================] - 557s 5ms/step - loss: 0.4319 -
accuracy: 0.7963 - auc_2: 0.8820 - val_loss: 0.4504 - val_accuracy: 0.7853 -
val_auc_2: 0.8712
Epoch 60/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4316 -
accuracy: 0.7965 - auc_2: 0.8822 - val_loss: 0.4504 - val_accuracy: 0.7856 -
val_auc_2: 0.8712
Epoch 61/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4313 -
accuracy: 0.7968 - auc_2: 0.8824 - val_loss: 0.4498 - val_accuracy: 0.7857 -
val_auc_2: 0.8716
Epoch 62/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4310 -
accuracy: 0.7970 - auc_2: 0.8826 - val_loss: 0.4497 - val_accuracy: 0.7858 -
val_auc_2: 0.8715
Epoch 63/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4306 -
accuracy: 0.7972 - auc_2: 0.8828 - val_loss: 0.4502 - val_accuracy: 0.7855 -
val_auc_2: 0.8715
Epoch 64/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4303 -
accuracy: 0.7973 - auc_2: 0.8830 - val_loss: 0.4516 - val_accuracy: 0.7847 -
val_auc_2: 0.8715
Epoch 65/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4300 -
accuracy: 0.7976 - auc_2: 0.8832 - val_loss: 0.4504 - val_accuracy: 0.7858 -
val_auc_2: 0.8714
Epoch 66/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4297 -
accuracy: 0.7977 - auc_2: 0.8834 - val_loss: 0.4509 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 67/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4295 -
accuracy: 0.7979 - auc_2: 0.8836 - val_loss: 0.4521 - val_accuracy: 0.7844 -
val_auc_2: 0.8712
Epoch 68/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4291 -
accuracy: 0.7981 - auc_2: 0.8838 - val_loss: 0.4501 - val_accuracy: 0.7854 -
val_auc_2: 0.8717
Epoch 69/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4288 -
accuracy: 0.7983 - auc_2: 0.8839 - val_loss: 0.4506 - val_accuracy: 0.7853 -
val_auc_2: 0.8714
Epoch 70/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4286 -
accuracy: 0.7985 - auc_2: 0.8841 - val_loss: 0.4514 - val_accuracy: 0.7850 -
val_auc_2: 0.8709
Epoch 71/400
```

```
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4283 -
accuracy: 0.7986 - auc_2: 0.8843 - val_loss: 0.4503 - val_accuracy: 0.7855 -
val_auc_2: 0.8713
Epoch 72/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4281 -
accuracy: 0.7988 - auc_2: 0.8844 - val_loss: 0.4508 - val_accuracy: 0.7861 -
val_auc_2: 0.8716
Epoch 73/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4279 -
accuracy: 0.7990 - auc_2: 0.8846 - val_loss: 0.4504 - val_accuracy: 0.7855 -
val_auc_2: 0.8715
Epoch 74/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4276 -
accuracy: 0.7990 - auc_2: 0.8847 - val_loss: 0.4504 - val_accuracy: 0.7858 -
val_auc_2: 0.8715
Epoch 75/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4274 -
accuracy: 0.7991 - auc_2: 0.8849 - val_loss: 0.4508 - val_accuracy: 0.7859 -
val_auc_2: 0.8715
Epoch 76/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4272 -
accuracy: 0.7994 - auc_2: 0.8850 - val_loss: 0.4499 - val_accuracy: 0.7858 -
val_auc_2: 0.8716
Epoch 77/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4269 -
accuracy: 0.7995 - auc_2: 0.8852 - val_loss: 0.4501 - val_accuracy: 0.7856 -
val_auc_2: 0.8716
Epoch 78/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4267 -
accuracy: 0.7997 - auc_2: 0.8853 - val_loss: 0.4500 - val_accuracy: 0.7861 -
val_auc_2: 0.8718
Epoch 79/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4265 -
accuracy: 0.7998 - auc_2: 0.8854 - val_loss: 0.4511 - val_accuracy: 0.7851 -
val_auc_2: 0.8710
Epoch 80/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4263 -
accuracy: 0.7999 - auc_2: 0.8856 - val_loss: 0.4506 - val_accuracy: 0.7861 -
val_auc_2: 0.8714
Epoch 81/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4260 -
accuracy: 0.8001 - auc_2: 0.8857 - val_loss: 0.4509 - val_accuracy: 0.7861 -
val_auc_2: 0.8716
Epoch 82/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4258 -
accuracy: 0.8002 - auc_2: 0.8859 - val_loss: 0.4510 - val_accuracy: 0.7857 -
val_auc_2: 0.8714
Epoch 83/400
```

```
105000/105000 [==============================] - 570s 5ms/step - loss: 0.4255 -
accuracy: 0.8004 - auc_2: 0.8860 - val_loss: 0.4509 - val_accuracy: 0.7856 -
val_auc_2: 0.8715
Epoch 84/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4253 -
accuracy: 0.8005 - auc_2: 0.8861 - val_loss: 0.4508 - val_accuracy: 0.7862 -
val_auc_2: 0.8717
Epoch 85/400
105000/105000 [==============================] - 570s 5ms/step - loss: 0.4251 -
accuracy: 0.8006 - auc_2: 0.8863 - val_loss: 0.4503 - val_accuracy: 0.7858 -
val_auc_2: 0.8717
Epoch 86/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4249 -
accuracy: 0.8008 - auc_2: 0.8864 - val_loss: 0.4514 - val_accuracy: 0.7849 -
val_auc_2: 0.8712
Epoch 87/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4247 -
accuracy: 0.8010 - auc_2: 0.8866 - val_loss: 0.4507 - val_accuracy: 0.7857 -
val_auc_2: 0.8716
Epoch 88/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4245 -
accuracy: 0.8010 - auc_2: 0.8867 - val_loss: 0.4509 - val_accuracy: 0.7858 -
val_auc_2: 0.8713
Epoch 89/400
105000/105000 [==============================] - 570s 5ms/step - loss: 0.4243 -
accuracy: 0.8012 - auc_2: 0.8868 - val_loss: 0.4505 - val_accuracy: 0.7864 -
val_auc_2: 0.8717
Epoch 90/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4241 -
accuracy: 0.8013 - auc_2: 0.8869 - val_loss: 0.4511 - val_accuracy: 0.7856 -
val_auc_2: 0.8716
Epoch 91/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4239 -
accuracy: 0.8015 - auc_2: 0.8871 - val_loss: 0.4506 - val_accuracy: 0.7860 -
val_auc_2: 0.8718
Epoch 92/400
105000/105000 [==============================] - 570s 5ms/step - loss: 0.4237 -
accuracy: 0.8016 - auc_2: 0.8872 - val_loss: 0.4509 - val_accuracy: 0.7858 -
val_auc_2: 0.8714
Epoch 93/400
105000/105000 [==============================] - 570s 5ms/step - loss: 0.4235 -
accuracy: 0.8016 - auc_2: 0.8873 - val_loss: 0.4512 - val_accuracy: 0.7860 -
val_auc_2: 0.8714
Epoch 94/400
105000/105000 [==============================] - 566s 5ms/step - loss: 0.4233 -
accuracy: 0.8018 - auc_2: 0.8874 - val_loss: 0.4508 - val_accuracy: 0.7861 -
val_auc_2: 0.8716
Epoch 95/400
```

```
105000/105000 [==============================] - 552s 5ms/step - loss: 0.4231 -
accuracy: 0.8019 - auc_2: 0.8875 - val_loss: 0.4511 - val_accuracy: 0.7856 -
val_auc_2: 0.8715
Epoch 96/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4229 -
accuracy: 0.8021 - auc_2: 0.8876 - val_loss: 0.4510 - val_accuracy: 0.7860 -
val_auc_2: 0.8717
Epoch 97/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4228 -
accuracy: 0.8021 - auc_2: 0.8877 - val_loss: 0.4512 - val_accuracy: 0.7862 -
val_auc_2: 0.8716
Epoch 98/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4226 -
accuracy: 0.8023 - auc_2: 0.8878 - val_loss: 0.4511 - val_accuracy: 0.7855 -
val_auc_2: 0.8713
Epoch 99/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4224 -
accuracy: 0.8024 - auc_2: 0.8880 - val_loss: 0.4509 - val_accuracy: 0.7858 -
val_auc_2: 0.8715
Epoch 100/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4223 -
accuracy: 0.8025 - auc_2: 0.8880 - val_loss: 0.4516 - val_accuracy: 0.7857 -
val_auc_2: 0.8713
Epoch 101/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4221 -
accuracy: 0.8026 - auc_2: 0.8882 - val_loss: 0.4509 - val_accuracy: 0.7859 -
val_auc_2: 0.8715
Epoch 102/400
105000/105000 [==============================] - 570s 5ms/step - loss: 0.4219 -
accuracy: 0.8026 - auc_2: 0.8883 - val_loss: 0.4512 - val_accuracy: 0.7858 -
val_auc_2: 0.8717
Epoch 103/400
105000/105000 [==============================] - 572s 5ms/step - loss: 0.4217 -
accuracy: 0.8029 - auc_2: 0.8884 - val_loss: 0.4513 - val_accuracy: 0.7857 -
val_auc_2: 0.8716
Epoch 104/400
105000/105000 [==============================] - 505s 5ms/step - loss: 0.4215 -
accuracy: 0.8029 - auc_2: 0.8885 - val_loss: 0.4517 - val_accuracy: 0.7855 -
val_auc_2: 0.8715
Epoch 105/400
105000/105000 [==============================] - 476s 5ms/step - loss: 0.4214 -
accuracy: 0.8030 - auc_2: 0.8886 - val_loss: 0.4515 - val_accuracy: 0.7857 -
val_auc_2: 0.8716
Epoch 106/400
105000/105000 [==============================] - 480s 5ms/step - loss: 0.4213 -
accuracy: 0.8031 - auc_2: 0.8887 - val_loss: 0.4517 - val_accuracy: 0.7859 -
val_auc_2: 0.8713
Epoch 107/400
```

```
105000/105000 [==============================] - 481s 5ms/step - loss: 0.4211 -
accuracy: 0.8031 - auc_2: 0.8887 - val_loss: 0.4515 - val_accuracy: 0.7857 -
val_auc_2: 0.8714
Epoch 108/400
105000/105000 [==============================] - 481s 5ms/step - loss: 0.4210 -
accuracy: 0.8032 - auc_2: 0.8888 - val_loss: 0.4516 - val_accuracy: 0.7857 -
val_auc_2: 0.8716
Epoch 109/400
105000/105000 [==============================] - 475s 5ms/step - loss: 0.4208 -
accuracy: 0.8034 - auc_2: 0.8889 - val_loss: 0.4510 - val_accuracy: 0.7860 -
val_auc_2: 0.8716
Epoch 110/400
105000/105000 [==============================] - 471s 4ms/step - loss: 0.4207 -
accuracy: 0.8034 - auc_2: 0.8890 - val_loss: 0.4515 - val_accuracy: 0.7857 -
val_auc_2: 0.8713
Epoch 111/400
105000/105000 [==============================] - 469s 4ms/step - loss: 0.4206 -
accuracy: 0.8035 - auc_2: 0.8891 - val_loss: 0.4517 - val_accuracy: 0.7863 -
val_auc_2: 0.8718
Epoch 112/400
105000/105000 [==============================] - 470s 4ms/step - loss: 0.4204 -
accuracy: 0.8036 - auc_2: 0.8891 - val_loss: 0.4517 - val_accuracy: 0.7857 -
val_auc_2: 0.8715
Epoch 113/400
105000/105000 [==============================] - 469s 4ms/step - loss: 0.4203 -
accuracy: 0.8037 - auc_2: 0.8892 - val_loss: 0.4517 - val_accuracy: 0.7851 -
val_auc_2: 0.8711
Epoch 114/400
105000/105000 [==============================] - 478s 5ms/step - loss: 0.4202 -
accuracy: 0.8038 - auc_2: 0.8893 - val_loss: 0.4513 - val_accuracy: 0.7860 -
val_auc_2: 0.8716
Epoch 115/400
105000/105000 [==============================] - 477s 5ms/step - loss: 0.4200 -
accuracy: 0.8039 - auc_2: 0.8894 - val_loss: 0.4517 - val_accuracy: 0.7854 -
val_auc_2: 0.8715
Epoch 116/400
105000/105000 [==============================] - 497s 5ms/step - loss: 0.4199 -
accuracy: 0.8040 - auc_2: 0.8895 - val_loss: 0.4514 - val_accuracy: 0.7862 -
val_auc_2: 0.8717
Epoch 117/400
105000/105000 [==============================] - 475s 5ms/step - loss: 0.4197 -
accuracy: 0.8040 - auc_2: 0.8896 - val_loss: 0.4517 - val_accuracy: 0.7858 -
val_auc_2: 0.8715
Epoch 118/400
105000/105000 [==============================] - 475s 5ms/step - loss: 0.4196 -
accuracy: 0.8041 - auc_2: 0.8896 - val_loss: 0.4515 - val_accuracy: 0.7858 -
val_auc_2: 0.8716
Epoch 119/400
```

```
105000/105000 [==============================] - 473s 5ms/step - loss: 0.4195 -
accuracy: 0.8043 - auc_2: 0.8897 - val_loss: 0.4516 - val_accuracy: 0.7855 -
val_auc_2: 0.8714
Epoch 120/400
105000/105000 [==============================] - 474s 5ms/step - loss: 0.4194 -
accuracy: 0.8043 - auc_2: 0.8898 - val_loss: 0.4516 - val_accuracy: 0.7859 -
val_auc_2: 0.8715
Epoch 121/400
105000/105000 [==============================] - 471s 4ms/step - loss: 0.4192 -
accuracy: 0.8044 - auc_2: 0.8899 - val_loss: 0.4520 - val_accuracy: 0.7858 -
val_auc_2: 0.8717
Epoch 122/400
105000/105000 [==============================] - 472s 4ms/step - loss: 0.4191 -
accuracy: 0.8045 - auc_2: 0.8900 - val_loss: 0.4521 - val_accuracy: 0.7857 -
val_auc_2: 0.8715
Epoch 123/400
105000/105000 [==============================] - 472s 4ms/step - loss: 0.4190 -
accuracy: 0.8045 - auc_2: 0.8900 - val_loss: 0.4516 - val_accuracy: 0.7858 -
val_auc_2: 0.8715
Epoch 124/400
105000/105000 [==============================] - 470s 4ms/step - loss: 0.4189 -
accuracy: 0.8046 - auc_2: 0.8901 - val_loss: 0.4519 - val_accuracy: 0.7860 -
val_auc_2: 0.8716
Epoch 125/400
105000/105000 [==============================] - 469s 4ms/step - loss: 0.4187 -
accuracy: 0.8047 - auc_2: 0.8902 - val_loss: 0.4518 - val_accuracy: 0.7858 -
val_auc_2: 0.8715
Epoch 126/400
105000/105000 [==============================] - 494s 5ms/step - loss: 0.4187 -
accuracy: 0.8047 - auc_2: 0.8902 - val_loss: 0.4518 - val_accuracy: 0.7856 -
val_auc_2: 0.8715
Epoch 127/400
105000/105000 [==============================] - 576s 5ms/step - loss: 0.4185 -
accuracy: 0.8047 - auc_2: 0.8903 - val_loss: 0.4522 - val_accuracy: 0.7858 -
val_auc_2: 0.8715
Epoch 128/400
105000/105000 [==============================] - 559s 5ms/step - loss: 0.4184 -
accuracy: 0.8049 - auc_2: 0.8904 - val_loss: 0.4522 - val_accuracy: 0.7856 -
val_auc_2: 0.8712
Epoch 129/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4183 -
accuracy: 0.8050 - auc_2: 0.8904 - val_loss: 0.4517 - val_accuracy: 0.7854 -
val_auc_2: 0.8714
Epoch 130/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4182 -
accuracy: 0.8050 - auc_2: 0.8905 - val_loss: 0.4526 - val_accuracy: 0.7857 -
val_auc_2: 0.8714
Epoch 131/400
```

```
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4181 -
accuracy: 0.8050 - auc_2: 0.8905 - val_loss: 0.4521 - val_accuracy: 0.7854 -
val_auc_2: 0.8714
Epoch 132/400
105000/105000 [==============================] - 576s 5ms/step - loss: 0.4180 -
accuracy: 0.8052 - auc_2: 0.8906 - val_loss: 0.4524 - val_accuracy: 0.7856 -
val_auc_2: 0.8714
Epoch 133/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4179 -
accuracy: 0.8052 - auc_2: 0.8907 - val_loss: 0.4520 - val_accuracy: 0.7856 -
val_auc_2: 0.8715
Epoch 134/400
105000/105000 [==============================] - 575s 5ms/step - loss: 0.4178 -
accuracy: 0.8052 - auc_2: 0.8907 - val_loss: 0.4523 - val_accuracy: 0.7859 -
val_auc_2: 0.8716
Epoch 135/400
105000/105000 [==============================] - 575s 5ms/step - loss: 0.4177 -
accuracy: 0.8053 - auc_2: 0.8908 - val_loss: 0.4521 - val_accuracy: 0.7855 -
val_auc_2: 0.8713
Epoch 136/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4176 -
accuracy: 0.8054 - auc_2: 0.8909 - val_loss: 0.4524 - val_accuracy: 0.7859 -
val_auc_2: 0.8714
Epoch 137/400
105000/105000 [==============================] - 578s 6ms/step - loss: 0.4175 -
accuracy: 0.8054 - auc_2: 0.8909 - val_loss: 0.4524 - val_accuracy: 0.7855 -
val_auc_2: 0.8716
Epoch 138/400
105000/105000 [==============================] - 559s 5ms/step - loss: 0.4174 -
accuracy: 0.8055 - auc_2: 0.8910 - val_loss: 0.4523 - val_accuracy: 0.7858 -
val_auc_2: 0.8714
Epoch 139/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4173 -
accuracy: 0.8056 - auc_2: 0.8910 - val_loss: 0.4523 - val_accuracy: 0.7855 -
val_auc_2: 0.8714
Epoch 140/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4173 -
accuracy: 0.8056 - auc_2: 0.8910 - val_loss: 0.4523 - val_accuracy: 0.7857 -
val_auc_2: 0.8715
Epoch 141/400
105000/105000 [==============================] - 578s 6ms/step - loss: 0.4172 -
accuracy: 0.8056 - auc_2: 0.8911 - val_loss: 0.4521 - val_accuracy: 0.7859 -
val_auc_2: 0.8716
Epoch 142/400
105000/105000 [==============================] - 564s 5ms/step - loss: 0.4171 -
accuracy: 0.8057 - auc_2: 0.8911 - val_loss: 0.4526 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 143/400
```

```
105000/105000 [==============================] - 566s 5ms/step - loss: 0.4170 -
accuracy: 0.8057 - auc_2: 0.8912 - val_loss: 0.4526 - val_accuracy: 0.7856 -
val_auc_2: 0.8713
Epoch 144/400
105000/105000 [==============================] - 575s 5ms/step - loss: 0.4170 -
accuracy: 0.8058 - auc_2: 0.8912 - val_loss: 0.4526 - val_accuracy: 0.7856 -
val_auc_2: 0.8714
Epoch 145/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4168 -
accuracy: 0.8058 - auc_2: 0.8913 - val_loss: 0.4522 - val_accuracy: 0.7854 -
val_auc_2: 0.8713
Epoch 146/400
105000/105000 [==============================] - 575s 5ms/step - loss: 0.4168 -
accuracy: 0.8059 - auc_2: 0.8913 - val_loss: 0.4526 - val_accuracy: 0.7855 -
val_auc_2: 0.8715
Epoch 147/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4167 -
accuracy: 0.8059 - auc_2: 0.8914 - val_loss: 0.4527 - val_accuracy: 0.7855 -
val_auc_2: 0.8714
Epoch 148/400
105000/105000 [==============================] - 576s 5ms/step - loss: 0.4166 -
accuracy: 0.8059 - auc_2: 0.8914 - val_loss: 0.4524 - val_accuracy: 0.7855 -
val_auc_2: 0.8713
Epoch 149/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4166 -
accuracy: 0.8060 - auc_2: 0.8915 - val_loss: 0.4523 - val_accuracy: 0.7855 -
val_auc_2: 0.8714
Epoch 150/400
105000/105000 [==============================] - 564s 5ms/step - loss: 0.4165 -
accuracy: 0.8061 - auc_2: 0.8915 - val_loss: 0.4525 - val_accuracy: 0.7860 -
val_auc_2: 0.8715
Epoch 151/400
105000/105000 [==============================] - 573s 5ms/step - loss: 0.4164 -
accuracy: 0.8061 - auc_2: 0.8916 - val_loss: 0.4526 - val_accuracy: 0.7857 -
val_auc_2: 0.8714
Epoch 152/400
105000/105000 [==============================] - 569s 5ms/step - loss: 0.4163 -
accuracy: 0.8062 - auc_2: 0.8916 - val_loss: 0.4527 - val_accuracy: 0.7855 -
val_auc_2: 0.8713
Epoch 153/400
105000/105000 [==============================] - 568s 5ms/step - loss: 0.4162 -
accuracy: 0.8062 - auc_2: 0.8916 - val_loss: 0.4527 - val_accuracy: 0.7858 -
val_auc_2: 0.8714
Epoch 154/400
105000/105000 [==============================] - 558s 5ms/step - loss: 0.4162 -
accuracy: 0.8063 - auc_2: 0.8917 - val_loss: 0.4529 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 155/400
```

```
105000/105000 [==============================] - 543s 5ms/step - loss: 0.4161 -
accuracy: 0.8063 - auc_2: 0.8917 - val_loss: 0.4528 - val_accuracy: 0.7854 -
val_auc_2: 0.8712
Epoch 156/400
105000/105000 [==============================] - 564s 5ms/step - loss: 0.4160 -
accuracy: 0.8063 - auc_2: 0.8918 - val_loss: 0.4531 - val_accuracy: 0.7851 -
val_auc_2: 0.8712
Epoch 157/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4160 -
accuracy: 0.8064 - auc_2: 0.8918 - val_loss: 0.4526 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 158/400
105000/105000 [==============================] - 571s 5ms/step - loss: 0.4159 -
accuracy: 0.8065 - auc_2: 0.8919 - val_loss: 0.4528 - val_accuracy: 0.7856 -
val_auc_2: 0.8712
Epoch 159/400
105000/105000 [==============================] - 575s 5ms/step - loss: 0.4158 -
accuracy: 0.8064 - auc_2: 0.8919 - val_loss: 0.4529 - val_accuracy: 0.7854 -
val_auc_2: 0.8711
Epoch 160/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4158 -
accuracy: 0.8065 - auc_2: 0.8919 - val_loss: 0.4527 - val_accuracy: 0.7857 -
val_auc_2: 0.8715
Epoch 161/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4157 -
accuracy: 0.8065 - auc_2: 0.8920 - val_loss: 0.4527 - val_accuracy: 0.7853 -
val_auc_2: 0.8711
Epoch 162/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4157 -
accuracy: 0.8066 - auc_2: 0.8920 - val_loss: 0.4531 - val_accuracy: 0.7854 -
val_auc_2: 0.8711
Epoch 163/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4156 -
accuracy: 0.8066 - auc_2: 0.8920 - val_loss: 0.4529 - val_accuracy: 0.7855 -
val_auc_2: 0.8711
Epoch 164/400
105000/105000 [==============================] - 578s 6ms/step - loss: 0.4155 -
accuracy: 0.8067 - auc_2: 0.8921 - val_loss: 0.4530 - val_accuracy: 0.7854 -
val_auc_2: 0.8711
Epoch 165/400
105000/105000 [==============================] - 576s 5ms/step - loss: 0.4155 -
accuracy: 0.8067 - auc_2: 0.8921 - val_loss: 0.4534 - val_accuracy: 0.7857 -
val_auc_2: 0.8712
Epoch 166/400
105000/105000 [==============================] - 574s 5ms/step - loss: 0.4154 -
accuracy: 0.8068 - auc_2: 0.8921 - val_loss: 0.4528 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 167/400
```

```
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4154 -
accuracy: 0.8067 - auc_2: 0.8922 - val_loss: 0.4526 - val_accuracy: 0.7855 -
val_auc_2: 0.8713
Epoch 168/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4153 -
accuracy: 0.8068 - auc_2: 0.8922 - val_loss: 0.4531 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 169/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4152 -
accuracy: 0.8068 - auc_2: 0.8922 - val_loss: 0.4530 - val_accuracy: 0.7856 -
val_auc_2: 0.8714
Epoch 170/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4152 -
accuracy: 0.8069 - auc_2: 0.8923 - val_loss: 0.4529 - val_accuracy: 0.7853 -
val_auc_2: 0.8712
Epoch 171/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4152 -
accuracy: 0.8069 - auc_2: 0.8923 - val_loss: 0.4531 - val_accuracy: 0.7856 -
val_auc_2: 0.8714
Epoch 172/400
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4151 -
accuracy: 0.8069 - auc_2: 0.8923 - val_loss: 0.4530 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 173/400
105000/105000 [==============================] - 577s 5ms/step - loss: 0.4151 -
accuracy: 0.8069 - auc_2: 0.8924 - val_loss: 0.4534 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 174/400
105000/105000 [==============================] - 449s 4ms/step - loss: 0.4150 -
accuracy: 0.8070 - auc_2: 0.8924 - val_loss: 0.4535 - val_accuracy: 0.7854 -
val_auc_2: 0.8712
Epoch 175/400
105000/105000 [==============================] - 452s 4ms/step - loss: 0.4150 -
accuracy: 0.8069 - auc_2: 0.8924 - val_loss: 0.4530 - val_accuracy: 0.7854 -
val_auc_2: 0.8712
Epoch 176/400
105000/105000 [==============================] - 436s 4ms/step - loss: 0.4149 -
accuracy: 0.8070 - auc_2: 0.8924 - val_loss: 0.4533 - val_accuracy: 0.7851 -
val_auc_2: 0.8710
Epoch 177/400
105000/105000 [==============================] - 506s 5ms/step - loss: 0.4149 -
accuracy: 0.8070 - auc_2: 0.8924 - val_loss: 0.4532 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 178/400
105000/105000 [==============================] - 588s 6ms/step - loss: 0.4148 -
accuracy: 0.8071 - auc_2: 0.8925 - val_loss: 0.4530 - val_accuracy: 0.7851 -
val_auc_2: 0.8710
Epoch 179/400
```

```
105000/105000 [==============================] - 579s 6ms/step - loss: 0.4148 -
accuracy: 0.8071 - auc_2: 0.8925 - val_loss: 0.4533 - val_accuracy: 0.7855 -
val_auc_2: 0.8710
Epoch 180/400
105000/105000 [==============================] - 584s 6ms/step - loss: 0.4148 -
accuracy: 0.8071 - auc_2: 0.8925 - val_loss: 0.4535 - val_accuracy: 0.7856 -
val_auc_2: 0.8713
Epoch 181/400
105000/105000 [==============================] - 496s 5ms/step - loss: 0.4147 -
accuracy: 0.8071 - auc_2: 0.8926 - val_loss: 0.4533 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 182/400
105000/105000 [==============================] - 464s 4ms/step - loss: 0.4147 -
accuracy: 0.8071 - auc_2: 0.8926 - val_loss: 0.4531 - val_accuracy: 0.7853 -
val_auc_2: 0.8712
Epoch 183/400
105000/105000 [==============================] - 457s 4ms/step - loss: 0.4146 -
accuracy: 0.8072 - auc_2: 0.8926 - val_loss: 0.4536 - val_accuracy: 0.7855 -
val_auc_2: 0.8711
Epoch 184/400
105000/105000 [==============================] - 464s 4ms/step - loss: 0.4146 -
accuracy: 0.8072 - auc_2: 0.8926 - val_loss: 0.4537 - val_accuracy: 0.7854 -
val_auc_2: 0.8711
Epoch 185/400
105000/105000 [==============================] - 451s 4ms/step - loss: 0.4145 -
accuracy: 0.8073 - auc_2: 0.8926 - val_loss: 0.4535 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 186/400
105000/105000 [==============================] - 458s 4ms/step - loss: 0.4145 -
accuracy: 0.8072 - auc_2: 0.8927 - val_loss: 0.4533 - val_accuracy: 0.7853 -
val_auc_2: 0.8713
Epoch 187/400
105000/105000 [==============================] - 589s 6ms/step - loss: 0.4145 -
accuracy: 0.8073 - auc_2: 0.8927 - val_loss: 0.4532 - val_accuracy: 0.7852 -
val_auc_2: 0.8712
Epoch 188/400
105000/105000 [==============================] - 570s 5ms/step - loss: 0.4144 -
accuracy: 0.8073 - auc_2: 0.8927 - val_loss: 0.4533 - val_accuracy: 0.7853 -
val_auc_2: 0.8711
Epoch 189/400
105000/105000 [==============================] - 454s 4ms/step - loss: 0.4144 -
accuracy: 0.8074 - auc_2: 0.8927 - val_loss: 0.4534 - val_accuracy: 0.7855 -
val_auc_2: 0.8712
Epoch 190/400
105000/105000 [==============================] - 446s 4ms/step - loss: 0.4144 -
accuracy: 0.8073 - auc_2: 0.8928 - val_loss: 0.4533 - val_accuracy: 0.7852 -
val_auc_2: 0.8712
Epoch 191/400
```

```
105000/105000 [==============================] - 446s 4ms/step - loss: 0.4143 -
accuracy: 0.8074 - auc_2: 0.8928 - val_loss: 0.4535 - val_accuracy: 0.7855 -
val_auc_2: 0.8711
Epoch 192/400
104996/105000 [============================>.] - ETA: 0s - loss: 0.4143 -
accuracy: 0.8074 - auc_2: 0.8928Started Monitor for Earing Stopping
105000/105000 [==============================] - 446s 4ms/step - loss: 0.4143 -
accuracy: 0.8074 - auc_2: 0.8928 - val_loss: 0.4534 - val_accuracy: 0.7853 -
val_auc_2: 0.8712
Epoch 193/400
104997/105000 [============================>.] - ETA: 0s - loss: 0.4143 -
accuracy: 0.8074 - auc_2: 0.8928Started Monitor for Earing Stopping
105000/105000 [==============================] - 445s 4ms/step - loss: 0.4143 -
accuracy: 0.8074 - auc_2: 0.8928 - val_loss: 0.4535 - val_accuracy: 0.7855 -
val_auc_2: 0.8711
Epoch 194/400
104988/105000 [============================>.] - ETA: 0s - loss: 0.4142 -
accuracy: 0.8075 - auc_2: 0.8928Started Monitor for Earing Stopping
105000/105000 [==============================] - 445s 4ms/step - loss: 0.4142 -
accuracy: 0.8075 - auc_2: 0.8928 - val_loss: 0.4535 - val_accuracy: 0.7852 -
val_auc_2: 0.8711
Epoch 195/400
104990/105000 [============================>.] - ETA: 0s - loss: 0.4142 -
accuracy: 0.8075 - auc_2: 0.8929Started Monitor for Earing Stopping
105000/105000 [==============================] - 444s 4ms/step - loss: 0.4142 -
accuracy: 0.8075 - auc_2: 0.8929 - val_loss: 0.4534 - val_accuracy: 0.7854 -
val_auc_2: 0.8711
Epoch 196/400
104989/105000 [============================>.] - ETA: 0s - loss: 0.4142 -
accuracy: 0.8075 - auc_2: 0.8929Started Monitor for Earing Stopping
105000/105000 [==============================] - 444s 4ms/step - loss: 0.4142 -
accuracy: 0.8075 - auc_2: 0.8929 - val_loss: 0.4536 - val_accuracy: 0.7854 -
val_auc_2: 0.8712
Epoch 197/400
104998/105000 [============================>.] - ETA: 0s - loss: 0.4141 -
accuracy: 0.8075 - auc_2: 0.8929Started Monitor for Earing Stopping
105000/105000 [==============================] - 444s 4ms/step - loss: 0.4141 -
accuracy: 0.8075 - auc_2: 0.8929 - val_loss: 0.4536 - val_accuracy: 0.7852 -
val_auc_2: 0.8710
Epoch 198/400
104994/105000 [============================>.] - ETA: 0s - loss: 0.4141 -
accuracy: 0.8075 - auc_2: 0.8929Started Monitor for Earing Stopping
105000/105000 [==============================] - 442s 4ms/step - loss: 0.4141 -
accuracy: 0.8075 - auc_2: 0.8929 - val_loss: 0.4536 - val_accuracy: 0.7856 -
val_auc_2: 0.8712
Epoch 199/400
105000/105000 [==============================] - ETA: 0s - loss: 0.4140 -
accuracy: 0.8075 - auc_2: 0.8929Started Monitor for Earing Stopping
```

```
105000/105000 [==============================] - 442s 4ms/step - loss: 0.4140 -
accuracy: 0.8075 - auc_2: 0.8929 - val_loss: 0.4537 - val_accuracy: 0.7853 -
val_auc_2: 0.8711
Epoch 200/400
104997/105000 [============================>.] - ETA: 0s - loss: 0.4140 -
accuracy: 0.8075 - auc_2: 0.8930Started Monitor for Earing Stopping
105000/105000 [==============================] - 439s 4ms/step - loss: 0.4140 -
accuracy: 0.8075 - auc_2: 0.8930 - val_loss: 0.4533 - val_accuracy: 0.7853 -
val_auc_2: 0.8712
Epoch 201/400
104991/105000 [============================>.] - ETA: 0s - loss: 0.4140 -
accuracy: 0.8076 - auc_2: 0.8930Started Monitor for Earing Stopping
105000/105000 [==============================] - 439s 4ms/step - loss: 0.4140 -
accuracy: 0.8076 - auc_2: 0.8930 - val_loss: 0.4533 - val_accuracy: 0.7853 -
val_auc_2: 0.8711
Epoch 202/400
104988/105000 [============================>.] - ETA: 0s - loss: 0.4140 -
accuracy: 0.8076 - auc_2: 0.8930Started Monitor for Earing Stopping
105000/105000 [==============================] - 440s 4ms/step - loss: 0.4140 -
accuracy: 0.8076 - auc_2: 0.8930 - val_loss: 0.4534 - val_accuracy: 0.7853 -
val_auc_2: 0.8710
Epoch 203/400
104997/105000 [============================>.] - ETA: 0s - loss: 0.4139 -
accuracy: 0.8076 - auc_2: 0.8930Started Monitor for Earing Stopping
105000/105000 [==============================] - 437s 4ms/step - loss: 0.4139 -
accuracy: 0.8076 - auc_2: 0.8930 - val_loss: 0.4536 - val_accuracy: 0.7855 -
val_auc_2: 0.8711
Epoch 204/400
104990/105000 [============================>.] - ETA: 0s - loss: 0.4139 -
accuracy: 0.8076 - auc_2: 0.8930Started Monitor for Earing Stopping
105000/105000 [==============================] - 438s 4ms/step - loss: 0.4139 -
accuracy: 0.8076 - auc_2: 0.8930 - val_loss: 0.4537 - val_accuracy: 0.7854 -
val_auc_2: 0.8712
Epoch 205/400
104996/105000 [============================>.] - ETA: 0s - loss: 0.4139 -
accuracy: 0.8076 - auc_2: 0.8930Started Monitor for Earing Stopping
105000/105000 [==============================] - 437s 4ms/step - loss: 0.4139 -
accuracy: 0.8076 - auc_2: 0.8930 - val_loss: 0.4535 - val_accuracy: 0.7852 -
val_auc_2: 0.8710
Epoch 206/400
104987/105000 [============================>.] - ETA: 0s - loss: 0.4139 -
accuracy: 0.8077 - auc_2: 0.8930Started Monitor for Earing Stopping
105000/105000 [==============================] - 485s 5ms/step - loss: 0.4139 -
accuracy: 0.8077 - auc_2: 0.8930 - val_loss: 0.4536 - val_accuracy: 0.7853 -
val_auc_2: 0.8710
Epoch 207/400
104991/105000 [============================>.] - ETA: 0s - loss: 0.4138 -
accuracy: 0.8076 - auc_2: 0.8931Started Monitor for Earing Stopping
```

```
105000/105000 [==============================] - 476s 5ms/step - loss: 0.4138 -
accuracy: 0.8076 - auc_2: 0.8931 - val_loss: 0.4536 - val_accuracy: 0.7854 -
val_auc_2: 0.8711
Epoch 208/400
104997/105000 [============================>.] - ETA: 0s - loss: 0.4138 -
accuracy: 0.8077 - auc_2: 0.8931Started Monitor for Earing Stopping
105000/105000 [==============================] - 545s 5ms/step - loss: 0.4138 -
accuracy: 0.8077 - auc_2: 0.8931 - val_loss: 0.4538 - val_accuracy: 0.7853 -
val_auc_2: 0.8711
Epoch 209/400
104989/105000 [============================>.] - ETA: 0s - loss: 0.4138 -
accuracy: 0.8077 - auc_2: 0.8931Started Monitor for Earing Stopping
105000/105000 [==============================] - 520s 5ms/step - loss: 0.4138 -
accuracy: 0.8077 - auc_2: 0.8931 - val_loss: 0.4536 - val_accuracy: 0.7853 -
val_auc_2: 0.8711
Epoch 210/400
104990/105000 [============================>.] - ETA: 0s - loss: 0.4138 -
accuracy: 0.8077 - auc_2: 0.8931Started Monitor for Earing Stopping
105000/105000 [==============================] - 565s 5ms/step - loss: 0.4138 -
accuracy: 0.8077 - auc_2: 0.8931 - val_loss: 0.4534 - val_accuracy: 0.7853 -
val_auc_2: 0.8710
Epoch 00210: early stopping
```

[142]:
```python
# save current state of model
model.save('./data/model.h5')
```

[115]:
```python
min_loss_loc = np.argmin(hist.history['val_loss'])

print('Best loss:', min(hist.history['val_loss']))
print('AUC:', hist.history['val_auc_2'][min_loss_loc])
print('Accuracy:', hist.history['val_accuracy'][min_loss_loc])
```

```
Best loss: 0.44971632957458496
AUC: 0.871549129486084
Accuracy: 0.7858020067214966
```

[75]:
```python
def plot_training_curves(history, title=None, caption=None, save_path=None):
    ''' Plot the training curves for loss and accuracy given a model history
    '''
    # find the minimum loss epoch
    minimum = np.min(history['val_loss'])
    min_loc = np.where(minimum == history['val_loss'])[0]
    # get the vline y-min and y-max
    loss_min, loss_max = (min(history['val_loss'] + history['loss']),
                          max(history['val_loss'] + history['loss']))
    acc_min, acc_max = (min(history['val_accuracy'] + history['accuracy']),
                        max(history['val_accuracy'] + history['accuracy']))
```

```python
    # create figure
    fig, ax = plt.subplots(nrows=2, figsize = (7,10))
    fig.suptitle(title)
    index = np.arange(1, len(history['accuracy']) + 1)
    # plot the loss and validation loss
    ax[0].plot(index, history['loss'], label = 'loss')
    ax[0].plot(index, history['val_loss'], label = 'val_loss')
    ax[0].vlines(min_loc + 1, loss_min, loss_max, label = 'min_loss_location')
    ax[0].set_title('Loss')
    ax[0].set_ylabel('Loss')
    ax[0].set_xlabel('Epochs')
    ax[0].legend()
    # plot the accuracy and validation accuracy
    ax[1].plot(index, history['accuracy'], label = 'accuracy')
    ax[1].plot(index, history['val_accuracy'], label = 'val_accuracy')
    ax[1].vlines(min_loc + 1, acc_min, acc_max, label = 'min_loss_location')
    ax[1].set_title('Accuracy')
    ax[1].set_ylabel('Accuracy')
    ax[1].set_xlabel('Epochs')
    ax[1].legend()
    if caption is not None:
        plt.figtext(0.5, 0.01, caption, wrap=True, horizontalalignment='center',␣
↪fontsize=14);
    plt.show()
    if save_path is not None:
        fig.savefig(save_path)
```

[140]:
```python
#with open('example_hist.pickle', 'wb') as f:
#    pickle.dump(hist.history, f)

# reload save training history
with open('example_hist.pickle', 'rb') as f:
    history = pickle.load(f)

caption = 'Figure 1: Training History of the neural network model.\n\
The training curves show evidence that the network is fully trained.'
plot_training_curves(history, title="Neural Network Training History",
                     caption=caption,
                     #save_path='./images/training_history.png'
                    )
```

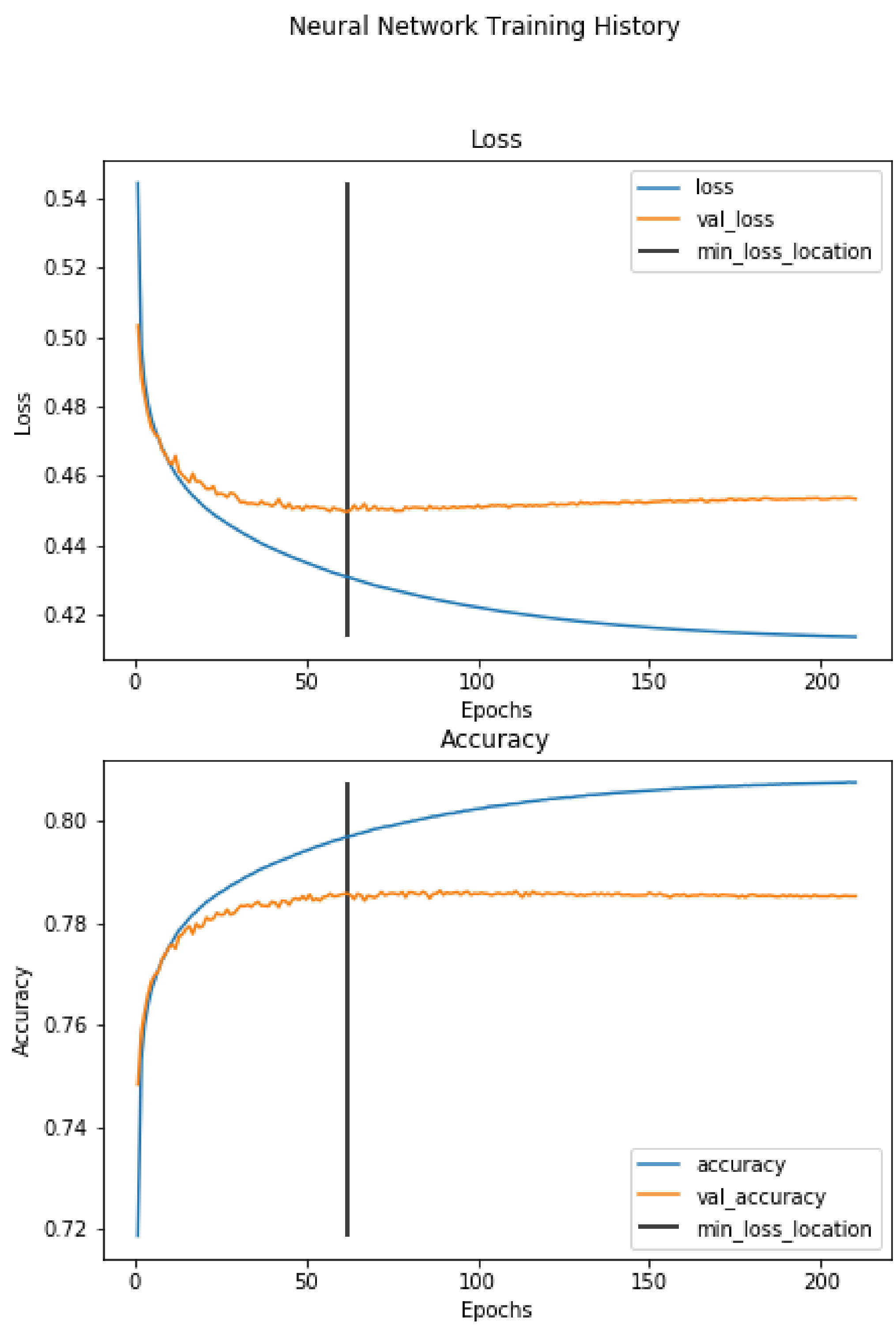

Figure 1: Training History of the neural network model.
The training curves show evidence that the network is fully trained.

[135]:
```
# uncomment to overwrite data
# generate ROC curve
#y_pred = model.predict(X_test).ravel()
#fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_test, y_pred)
#roc_metrics = pd.DataFrame(
#    zip(fpr_keras, tpr_keras, thresholds_keras),
#    columns=['fpr', 'tpr', 'thres'])
#roc_metrics.to_csv('./data/roc_metrics.csv', index=False)
```

[10]:
```
# read back stored data for reproducibility offline
roc_metrics = pd.read_csv('./data/roc_metrics.csv')
auc = auc_score(roc_metrics.fpr, roc_metrics.tpr)

plt.figure(figsize=(10,8))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(roc_metrics.fpr, roc_metrics.tpr, label='AUC = {:.3f}'.format(auc))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
caption = 'Figure 2: ROC Curve for Model Predictions on Test Dataset'
plt.figtext(0.5, 0.01, caption, wrap=True, horizontalalignment='center',␣
 ↪fontsize=14);
#plt.savefig('./images/roc_curve.png')
```

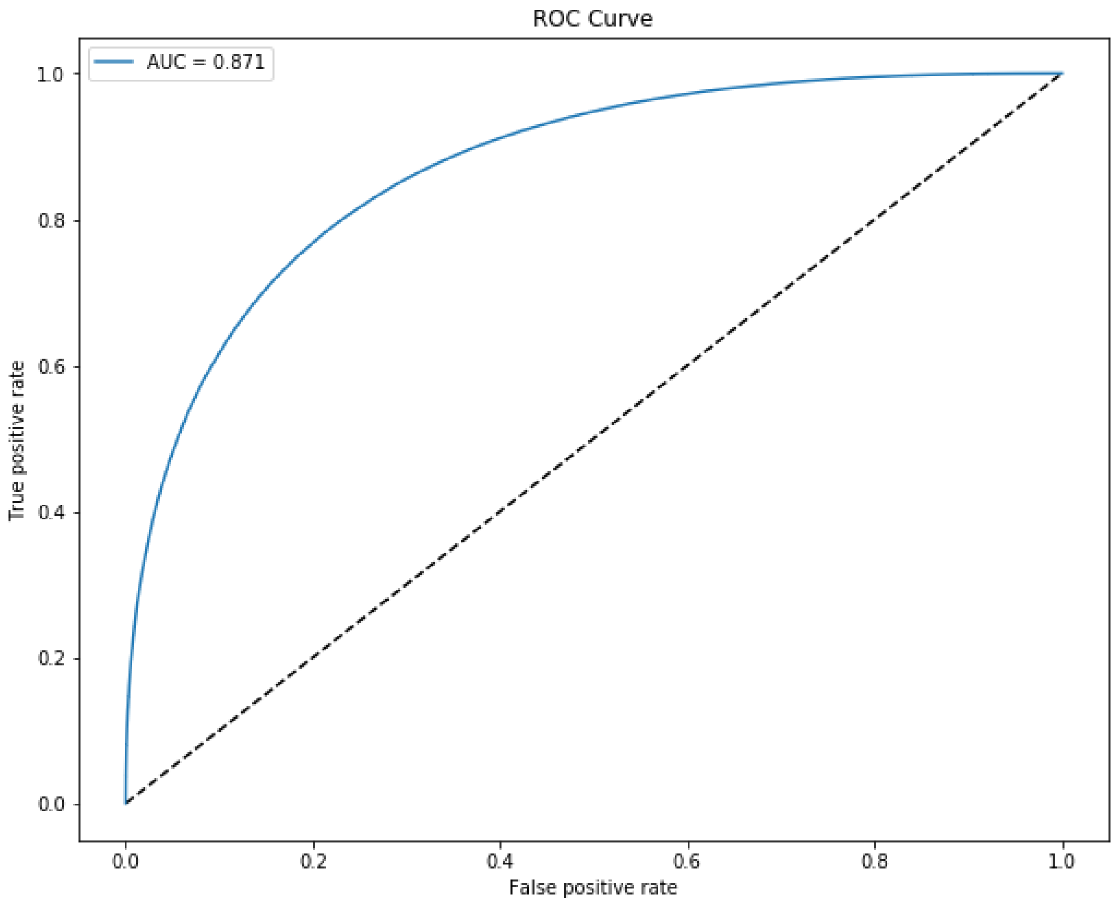

Figure 2: ROC Curve for Model Predictions on Test Dataset

[4]:
```
# repeated runs of model with different seeds

repeated_runs_auc = [
    0.8732744455337524,
    0.871549129486084,
    0.8727940320968628,
    0.8728284239768982,
    0.8732682466506958,
    0.8734915256500244,
    0.8746642470359802,
    0.8726969957351685,
    0.8716903924942017,
    0.8746868371963501,
    0.8711093068122864,
    0.8746849298477173,
    0.8720808029174805,
    0.8712741136550903,
```

```
    0.8736899495124817,
    0.8725572228431702,
    0.8705684542655945,
    0.8717679977416992,
    0.8705768585205078,
    0.8725365996360779,
    0.8724434971809387,
    0.8725489377975464,
    0.8739815354347229,
    0.8731337785720825,
    0.8729691505432129,
    0.8722008466720581,
    0.8719663619995117,
    0.872745156288147,
    0.8715148568153381,
    0.874409019947052

]

repeated_runs_loss = [
    0.4477136433124542,
    0.44971632957458496,
    0.4489780068397522,
    0.4481181800365448,
    0.44732430577278137,
    0.4475749135017395,
    0.4462486207485199,
    0.44879305362701416,
    0.4501331150531769,
    0.4456951916217804,
    0.45096442103385925,
    0.44551539421081543,
    0.44927680492401123,
    0.4507071375846863,
    0.446825236082077,
    0.44892680644989014,
    0.45166826248168945,
    0.4501870572566986,
    0.4518648386001587,
    0.448852002620697,
    0.4488407075405121,
    0.44896048307418823,
    0.4467504322528839,
    0.44824519753456116,
    0.44829028844833374,
    0.4492127597332001,
    0.4502909183502197,
```

```
    0.44889262318611145,
    0.4506274461746216,
    0.446505069732666

]

repeated_runs_acc = [
    0.7878699898719788,
    0.7858020067214966,
    0.7876880168914795,
    0.7876840233802795,
    0.788345992565155,
    0.7885339856147766,
    0.7901520133018494,
    0.787339985370636,
    0.7864779829978943,
    0.7891579866409302,
    0.7860220074653625,
    0.7893180251121521,
    0.787056028842926,
    0.7858679890632629,
    0.7884140014648438,
    0.7871099710464478,
    0.7856720089912415,
    0.786545991897583,
    0.7852619886398315,
    0.7871519923210144,
    0.7873740196228027,
    0.7871580123901367,
    0.7885199785232544,
    0.7875980138778687,
    0.7877079844474792,
    0.7870500087738037,
    0.7871119976043701,
    0.7873460054397583,
    0.7863839864730835,
    0.789322018623352

]
```

[11]:
```
#just use this to up sample
runs = list()
#for _ in range(100):
#    runs.append(np.random.choice(repeated_runs_auc, 100).mean())
runs = runs + repeated_runs_auc

# plot the distributions of AUC
```

```python
mean = np.mean(runs)
std = np.std(runs)

plt.figure(figsize=(10,8))
canvas = plt.hist(runs)
max_height = max(canvas[0])
plt.vlines(mean + std, 0, max_height+1, label = '1-std above mean')
plt.vlines(mean - std, 0, max_height+1, label = '1-std below mean')
plt.vlines(mean, 0, max_height+1, label = 'mean', color='r')
plt.title('Distribution of AUC values from Repeated Training')
plt.xlabel('AUC')
plt.ylabel('Frequency')

caption = 'Figure 3: Histogram of AUC values from training the neural network\n\
model ' + str(len(runs)) + ' times with different random seeds.'
plt.figtext(0.5, 0.01, caption, wrap=True, horizontalalignment='center',␣
 ↪fontsize=14);
# plot the reference from the paper
#plt.vlines(0.885, 0, max_height+1, label = 'reference mean', color='b')
plt.legend();
plt.savefig('./images/auc_distribution.png')
```

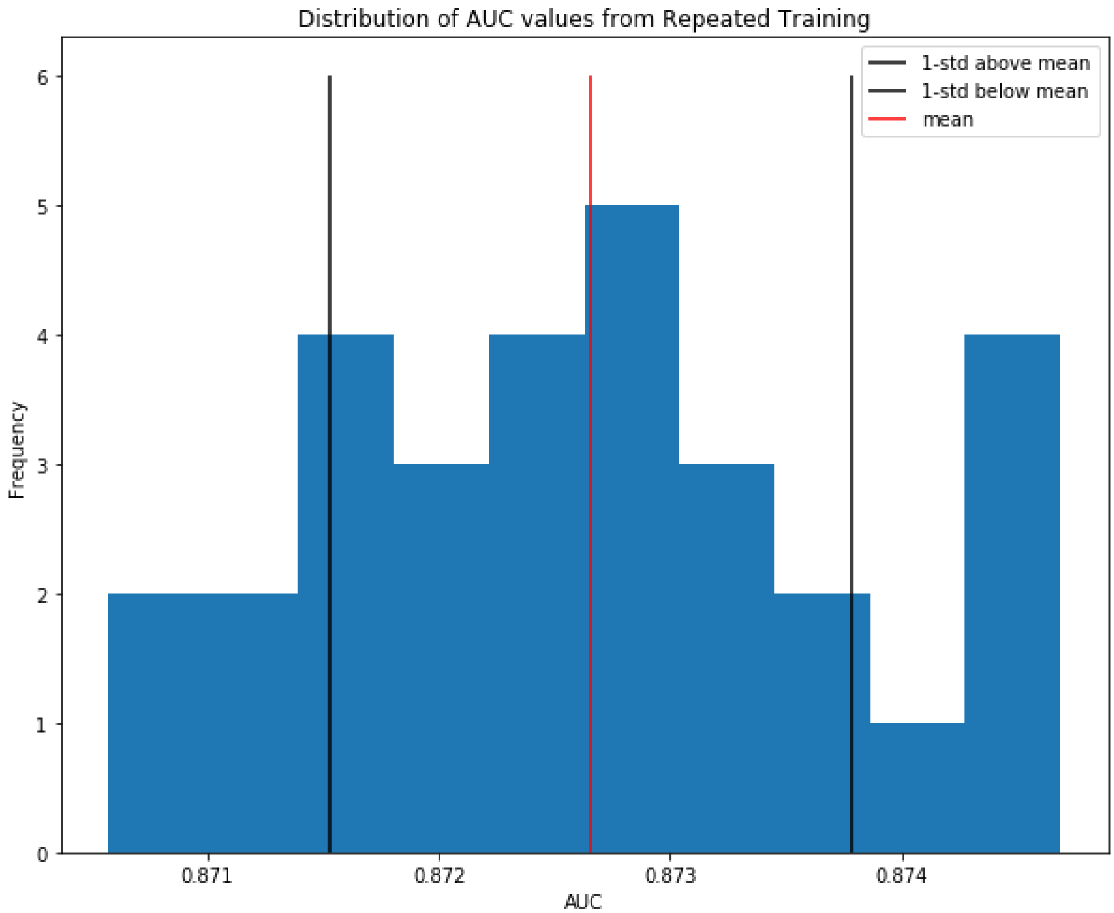

Figure 3: Histogram of AUC values from training the neural network model 30 times with different random seeds.

[6]:
```
# test auc scores against mean from paper
ttest_1samp(runs, 0.885)
```

[6]:
```
Ttest_1sampResult(statistic=-58.79915399037024, pvalue=1.0211818996222144e-31)
```