Name : Devdeep Shetranjiwala
Email ID : devdeep0702@gmail.com

Task 1. Electron/photon classification

Datasets:</br> https://cernbox.cern.ch/index.php/s/AtBT8y4MiQYFcgc (photons) </br> https://cernbox.cern.ch/index.php/s/FbXw3V4XNyYB3oA (electrons) </br>

> **Description: </br> 32x32 matrices (two channels - hit energy and time) for two classes of particles electrons and photons impinging on a calorimeter Please use a deep learning method of your choice to achieve the highest possible classification on this dataset.**
>
> **In this task, we will use deep learning to classify two classes of particles: electrons and photons impinging on a calorimeter. We will use two datasets, one for photons and one for electrons, which contains 32x32 matrices (two channels - hit energy and time) for each particle.</br> We will usw deep learning framework to implement our solution, Keras/TensorFlow. Our goal is to achieve the highest possible classification accuracy on this dataset with a ROC AUC score of at least 0.80. </br> First, we will load the data and preprocess it.**
> **Data Preprocessing :**
> **We will load the datasets for photons and electrons and preprocess them. We will convert the data into numpy arrays and normalize them by dividing each pixel value by the maximum pixel value.**

In [ ]:

```
import h5py
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
import tensorflow as tf
import matplotlib.pyplot as plt
import pickle
import seaborn as sns
import sys
```

In [ ]:

```
print("Num of GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
Num of GPUs Available:  1
```

In [ ]:

```
#Getting data
import requests
url = 'https://cernbox.cern.ch/remote.php/dav/public-files/AtBT8y4MiQYFcgc/SinglePhotonPt50_IMGCROPS_n249k_RHv1.hdf5'
r = requests.get(url, allow_redirects=True)
open('photons.hdf5', 'wb').write(r.content)
url = 'https://cernbox.cern.ch/remote.php/dav/public-files/FbXw3V4XNyYB3oA/SingleElectronPt50_IMGCROPS_n249k_RHv1.hdf5'
r = requests.get(url, allow_redirects=True)
open('electrons.hdf5', 'wb').write(r.content)
```

Out[ ]:

```
128927319
```

> **Model definition:</br> Keras/TensorFlow</br> We will define a simple convolutional neural network (CNN) with two convolutional layers, followed by a max pooling layer, and two fully connected layers. We will use the sigmoid activation function in the last layer as this is a binary**

> **classification problem.**

In [ ]:

```
class Net(tf.keras.Model):
    def __init__(self, skip_connection=False):
        super().__init__()
        self.skip_connection = skip_connection
        self.rflip = tf.keras.layers.RandomFlip(mode="horizontal_and_vertical")
        self.conv1 = tf.keras.layers.Conv2D(filters=64, kernel_size=3, padding='same', activation='relu')
        self.conv1p1 = tf.keras.layers.Conv2D(filters=64, kernel_size=3, padding='same')
        self.maxpool1 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))
        self.batchnorm1 = tf.keras.layers.BatchNormalization()

        self.conv2 = tf.keras.layers.Conv2D(filters=128, kernel_size=3, padding='same', activation='relu')
        self.conv2p1 = tf.keras.layers.Conv2D(filters=128, kernel_size=3, padding='same')
        self.maxpool2 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))
        self.batchnorm2 = tf.keras.layers.BatchNormalization()

        self.conv3 = tf.keras.layers.Conv2D(filters=128, kernel_size=3, padding='same', activation='relu')
        self.conv3p1 = tf.keras.layers.Conv2D(filters=128, kernel_size=3, padding='same')
        self.maxpool3 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))
        self.batchnorm3 = tf.keras.layers.BatchNormalization()

        self.conv4 = tf.keras.layers.Conv2D(filters=64, kernel_size=1, padding='same', activation='relu')

        self.gloavgpool = tf.keras.layers.GlobalAveragePooling2D()

        self.dense1 = tf.keras.layers.Dense(1, activation='sigmoid')
        self.dropout1 = tf.keras.layers.Dropout(.1)

    def build_graph(self):
        x = tf.keras.layers.Input(shape=train_x.shape[1:])
        return tf.keras.Model(inputs=[x], outputs=self.call(x))

    def call(self, inputs):
        x = self.rflip(inputs)

        x_res = self.conv1(x)
        x = self.conv1p1(x_res)
        if self.skip_connection: x = x+x_res
        x = self.maxpool1(x)
        x = self.batchnorm1(tf.keras.layers.ReLU()(x))

        x_res = self.conv2(x)
        x = self.conv2p1(x_res)
        if self.skip_connection: x = x+x_res
        x = self.maxpool2(x)
        x = self.batchnorm2(tf.keras.layers.ReLU()(x))

        x_res = self.conv3(x)
        x = self.conv3p1(x_res)
        if self.skip_connection: x = x+x_res
        x = self.maxpool3(x)
        x = self.batchnorm3(tf.keras.layers.ReLU()(x))

        x = self.conv4(x)

        x = self.dropout1(self.gloavgpool(x))

        return self.dense1(x)
```

In [ ]:

```
#read the hdf5 fies
```

```
#read the hdf5 files
e_set = h5py.File('./electrons.hdf5', 'r')
p_set = h5py.File('./photons.hdf5', 'r')

#convert to np arrays
e_x, p_x = np.asarray(e_set['X']), np.asarray(p_set['X'])

del(e_set,p_set)
#concat the electon/photon arrays
ep_x = np.concatenate([e_x, p_x])
ep_target = np.concatenate([np.ones(len(e_x)), np.zeros(len(p_x))])

del(e_x,p_x)
#remove entries with all zeros
nonzeros = np.sum(ep_x, axis=(1,2,3))!=0
ep_x, ep_target = ep_x[nonzeros], ep_target[nonzeros]

#set seed for reproducibility
seed = 42

#split into train (80%) /test (20%) set and save
X_train, X_test, y_train, y_test = train_test_split(ep_x, ep_target, test_size=0.2, stra
tify=ep_target, random_state=seed)

del(ep_x,ep_target)
#normalize data with training set mean and std to ensure no data leakage
X_train_mean, X_train_std = X_train.mean((0,1,2)), X_train.std((0,1,2))
X_train = (X_train-X_train_mean)/X_train_std
X_test = (X_test-X_train_mean)/X_train_std
del(X_train_mean,X_train_std)

#split into train (60% train , 20% validate , 20% test) -> 80 / 20
X_train, X_valid, y_train, y_valid = train_test_split(X_train,y_train, test_size=0.2, st
ratify=y_train, random_state=seed)

train_x, train_y = X_train, y_train
del(X_train,y_train)
valid_x, valid_y = X_valid, y_valid
del(X_valid,y_valid)
test_x, test_y = X_test,y_test
del(X_test,y_test)
```

In [ ]:

```
model = Net()
model.build_graph().summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 2)]       0

 random_flip (RandomFlip)    (None, 32, 32, 2)         0

 conv2d (Conv2D)             (None, 32, 32, 64)        1216

 conv2d_1 (Conv2D)           (None, 32, 32, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 16, 16, 64)       0
 )

 re_lu (ReLU)                (None, 16, 16, 64)        0

 batch_normalization (BatchN  (None, 16, 16, 64)       256
 ormalization)

 conv2d_2 (Conv2D)           (None, 16, 16, 128)       73856

 conv2d_3 (Conv2D)           (None, 16, 16, 128)       147584

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 128)        0
 2D)
```

```
2D)

 re_lu_1 (ReLU)              (None, 8, 8, 128)         0

 batch_normalization_1 (Batc  (None, 8, 8, 128)        512
 hNormalization)

 conv2d_4 (Conv2D)           (None, 8, 8, 128)         147584

 conv2d_5 (Conv2D)           (None, 8, 8, 128)         147584

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 re_lu_2 (ReLU)              (None, 4, 4, 128)         0

 batch_normalization_2 (Batc  (None, 4, 4, 128)        512
 hNormalization)

 conv2d_6 (Conv2D)           (None, 4, 4, 64)          8256

 global_average_pooling2d (G  (None, 64)               0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 564,353
Trainable params: 563,713
Non-trainable params: 640
_________________________________________________________________
```

In [ ]:

```
model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=[tf.keras.metrics.BinaryAccuracy(),
                    tf.keras.metrics.AUC(name='auc')])
```

In [ ]:

```
history = model.fit(x=train_x, y=train_y, batch_size=32, epochs=100, validation_data=(va
lid_x, valid_y))
```

```
Epoch 1/100
9949/9949 [==============================] - 106s 9ms/step - loss: 0.6017 - binary_accura
cy: 0.6800 - auc: 0.7363 - val_loss: 0.5695 - val_binary_accuracy: 0.7106 - val_auc: 0.77
39
Epoch 2/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5699 - binary_accurac
y: 0.7112 - auc: 0.7739 - val_loss: 0.5699 - val_binary_accuracy: 0.7108 - val_auc: 0.781
8
Epoch 3/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5627 - binary_accurac
y: 0.7178 - auc: 0.7813 - val_loss: 0.5579 - val_binary_accuracy: 0.7194 - val_auc: 0.788
6
Epoch 4/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5584 - binary_accurac
y: 0.7208 - auc: 0.7854 - val_loss: 0.5541 - val_binary_accuracy: 0.7260 - val_auc: 0.791
8
Epoch 5/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5544 - binary_accurac
y: 0.7239 - auc: 0.7894 - val_loss: 0.5505 - val_binary_accuracy: 0.7253 - val_auc: 0.795
9
Epoch 6/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5518 - binary_accurac
y: 0.7254 - auc: 0.7918 - val_loss: 0.5507 - val_binary_accuracy: 0.7265 - val_auc: 0.795
4
Epoch 7/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5503 - binary_accurac
```

```
y: 0.7266 - auc: 0.7934 - val_loss: 0.5554 - val_binary_accuracy: 0.7250 - val_auc: 0.795
4
Epoch 8/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5483 - binary_accurac
y: 0.7286 - auc: 0.7953 - val_loss: 0.5446 - val_binary_accuracy: 0.7317 - val_auc: 0.801
0
Epoch 9/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5463 - binary_accurac
y: 0.7302 - auc: 0.7970 - val_loss: 0.5503 - val_binary_accuracy: 0.7281 - val_auc: 0.798
3
Epoch 10/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5453 - binary_accurac
y: 0.7305 - auc: 0.7981 - val_loss: 0.5465 - val_binary_accuracy: 0.7314 - val_auc: 0.799
5
Epoch 11/100
9949/9949 [==============================] - 88s 9ms/step - loss: 0.5438 - binary_accurac
y: 0.7323 - auc: 0.7995 - val_loss: 0.5447 - val_binary_accuracy: 0.7286 - val_auc: 0.799
2
Epoch 12/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5430 - binary_accurac
y: 0.7319 - auc: 0.8001 - val_loss: 0.5440 - val_binary_accuracy: 0.7312 - val_auc: 0.801
2
Epoch 13/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5420 - binary_accurac
y: 0.7330 - auc: 0.8010 - val_loss: 0.5427 - val_binary_accuracy: 0.7310 - val_auc: 0.801
7
Epoch 14/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.5403 - binary_accurac
y: 0.7349 - auc: 0.8027 - val_loss: 0.5495 - val_binary_accuracy: 0.7276 - val_auc: 0.799
1
Epoch 15/100
9949/9949 [==============================] - 88s 9ms/step - loss: 0.5396 - binary_accurac
y: 0.7344 - auc: 0.8033 - val_loss: 0.5378 - val_binary_accuracy: 0.7353 - val_auc: 0.805
2
Epoch 16/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.5383 - binary_accurac
y: 0.7355 - auc: 0.8043 - val_loss: 0.5389 - val_binary_accuracy: 0.7352 - val_auc: 0.804
8
Epoch 17/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5374 - binary_accurac
y: 0.7359 - auc: 0.8052 - val_loss: 0.5409 - val_binary_accuracy: 0.7309 - val_auc: 0.805
2
Epoch 18/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5367 - binary_accurac
y: 0.7367 - auc: 0.8059 - val_loss: 0.5518 - val_binary_accuracy: 0.7345 - val_auc: 0.804
0
Epoch 19/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5356 - binary_accurac
y: 0.7372 - auc: 0.8069 - val_loss: 0.5405 - val_binary_accuracy: 0.7338 - val_auc: 0.804
6
Epoch 20/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5349 - binary_accurac
y: 0.7374 - auc: 0.8074 - val_loss: 0.5467 - val_binary_accuracy: 0.7281 - val_auc: 0.804
7
Epoch 21/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5340 - binary_accurac
y: 0.7378 - auc: 0.8082 - val_loss: 0.5416 - val_binary_accuracy: 0.7359 - val_auc: 0.807
1
Epoch 22/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5333 - binary_accurac
y: 0.7390 - auc: 0.8089 - val_loss: 0.5386 - val_binary_accuracy: 0.7342 - val_auc: 0.807
0
Epoch 23/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5323 - binary_accurac
y: 0.7397 - auc: 0.8097 - val_loss: 0.5405 - val_binary_accuracy: 0.7347 - val_auc: 0.806
9
Epoch 24/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5316 - binary_accurac
y: 0.7403 - auc: 0.8103 - val_loss: 0.5427 - val_binary_accuracy: 0.7347 - val_auc: 0.806
5
Epoch 25/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5305 - binary_accurac
```

```
y: 0.7405 - auc: 0.8112 - val_loss: 0.5331 - val_binary_accuracy: 0.7386 - val_auc: 0.809
4
Epoch 26/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5298 - binary_accurac
y: 0.7411 - auc: 0.8119 - val_loss: 0.5340 - val_binary_accuracy: 0.7381 - val_auc: 0.808
7
Epoch 27/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5293 - binary_accurac
y: 0.7414 - auc: 0.8123 - val_loss: 0.5322 - val_binary_accuracy: 0.7390 - val_auc: 0.809
6
Epoch 28/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5282 - binary_accurac
y: 0.7424 - auc: 0.8131 - val_loss: 0.5350 - val_binary_accuracy: 0.7368 - val_auc: 0.807
9
Epoch 29/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.5277 - binary_accurac
y: 0.7422 - auc: 0.8136 - val_loss: 0.5349 - val_binary_accuracy: 0.7373 - val_auc: 0.807
2
Epoch 30/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5269 - binary_accurac
y: 0.7430 - auc: 0.8144 - val_loss: 0.5380 - val_binary_accuracy: 0.7360 - val_auc: 0.809
3
Epoch 31/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5263 - binary_accurac
y: 0.7434 - auc: 0.8149 - val_loss: 0.5539 - val_binary_accuracy: 0.7236 - val_auc: 0.796
0
Epoch 32/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5248 - binary_accurac
y: 0.7444 - auc: 0.8161 - val_loss: 0.5409 - val_binary_accuracy: 0.7326 - val_auc: 0.804
1
Epoch 33/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.5248 - binary_accurac
y: 0.7444 - auc: 0.8161 - val_loss: 0.5331 - val_binary_accuracy: 0.7401 - val_auc: 0.809
9
Epoch 34/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5238 - binary_accurac
y: 0.7458 - auc: 0.8170 - val_loss: 0.5352 - val_binary_accuracy: 0.7382 - val_auc: 0.807
7
Epoch 35/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5230 - binary_accurac
y: 0.7457 - auc: 0.8177 - val_loss: 0.5372 - val_binary_accuracy: 0.7380 - val_auc: 0.808
2
Epoch 36/100
9949/9949 [==============================] - 93s 9ms/step - loss: 0.5223 - binary_accurac
y: 0.7461 - auc: 0.8182 - val_loss: 0.5375 - val_binary_accuracy: 0.7362 - val_auc: 0.808
1
Epoch 37/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5216 - binary_accurac
y: 0.7466 - auc: 0.8189 - val_loss: 0.5358 - val_binary_accuracy: 0.7385 - val_auc: 0.809
0
Epoch 38/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5206 - binary_accurac
y: 0.7468 - auc: 0.8196 - val_loss: 0.5360 - val_binary_accuracy: 0.7368 - val_auc: 0.808
5
Epoch 39/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5200 - binary_accurac
y: 0.7475 - auc: 0.8201 - val_loss: 0.5358 - val_binary_accuracy: 0.7358 - val_auc: 0.807
2
Epoch 40/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5186 - binary_accurac
y: 0.7491 - auc: 0.8212 - val_loss: 0.5357 - val_binary_accuracy: 0.7397 - val_auc: 0.807
6
Epoch 41/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5182 - binary_accurac
y: 0.7493 - auc: 0.8215 - val_loss: 0.5427 - val_binary_accuracy: 0.7354 - val_auc: 0.807
2
Epoch 42/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5171 - binary_accurac
y: 0.7492 - auc: 0.8224 - val_loss: 0.5392 - val_binary_accuracy: 0.7396 - val_auc: 0.808
8
Epoch 43/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5164 - binary_accurac
```

```
y: 0.7502 - auc: 0.8231 - val_loss: 0.5394 - val_binary_accuracy: 0.7357 - val_auc: 0.805
8
Epoch 44/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.5155 - binary_accurac
y: 0.7509 - auc: 0.8237 - val_loss: 0.5408 - val_binary_accuracy: 0.7351 - val_auc: 0.804
2
Epoch 45/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5143 - binary_accurac
y: 0.7514 - auc: 0.8247 - val_loss: 0.5436 - val_binary_accuracy: 0.7353 - val_auc: 0.806
2
Epoch 46/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5142 - binary_accurac
y: 0.7510 - auc: 0.8248 - val_loss: 0.5375 - val_binary_accuracy: 0.7370 - val_auc: 0.807
7
Epoch 47/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5126 - binary_accurac
y: 0.7526 - auc: 0.8262 - val_loss: 0.5410 - val_binary_accuracy: 0.7361 - val_auc: 0.804
4
Epoch 48/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5117 - binary_accurac
y: 0.7529 - auc: 0.8268 - val_loss: 0.5408 - val_binary_accuracy: 0.7364 - val_auc: 0.805
3
Epoch 49/100
9949/9949 [==============================] - 88s 9ms/step - loss: 0.5109 - binary_accurac
y: 0.7530 - auc: 0.8274 - val_loss: 0.5400 - val_binary_accuracy: 0.7356 - val_auc: 0.807
5
Epoch 50/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5104 - binary_accurac
y: 0.7534 - auc: 0.8278 - val_loss: 0.5560 - val_binary_accuracy: 0.7258 - val_auc: 0.799
3
Epoch 51/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5089 - binary_accurac
y: 0.7542 - auc: 0.8290 - val_loss: 0.5438 - val_binary_accuracy: 0.7374 - val_auc: 0.805
0
Epoch 52/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5072 - binary_accurac
y: 0.7550 - auc: 0.8303 - val_loss: 0.5387 - val_binary_accuracy: 0.7374 - val_auc: 0.806
8
Epoch 53/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5065 - binary_accurac
y: 0.7553 - auc: 0.8308 - val_loss: 0.5420 - val_binary_accuracy: 0.7362 - val_auc: 0.805
0
Epoch 54/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5057 - binary_accurac
y: 0.7562 - auc: 0.8313 - val_loss: 0.5560 - val_binary_accuracy: 0.7302 - val_auc: 0.797
4
Epoch 55/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.5045 - binary_accurac
y: 0.7577 - auc: 0.8324 - val_loss: 0.5407 - val_binary_accuracy: 0.7363 - val_auc: 0.804
5
Epoch 56/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.5036 - binary_accurac
y: 0.7577 - auc: 0.8330 - val_loss: 0.5441 - val_binary_accuracy: 0.7337 - val_auc: 0.802
9
Epoch 57/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5022 - binary_accurac
y: 0.7582 - auc: 0.8341 - val_loss: 0.5476 - val_binary_accuracy: 0.7367 - val_auc: 0.804
4
Epoch 58/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.5017 - binary_accurac
y: 0.7584 - auc: 0.8345 - val_loss: 0.5423 - val_binary_accuracy: 0.7367 - val_auc: 0.805
0
Epoch 59/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.4999 - binary_accurac
y: 0.7593 - auc: 0.8358 - val_loss: 0.5454 - val_binary_accuracy: 0.7344 - val_auc: 0.803
0
Epoch 60/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4992 - binary_accurac
y: 0.7602 - auc: 0.8362 - val_loss: 0.5473 - val_binary_accuracy: 0.7343 - val_auc: 0.803
0
Epoch 61/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.4976 - binary_accurac
```

```
y: 0.7607 - auc: 0.8375 - val_loss: 0.5568 - val_binary_accuracy: 0.7362 - val_auc: 0.802
5
Epoch 62/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4971 - binary_accurac
y: 0.7614 - auc: 0.8379 - val_loss: 0.5513 - val_binary_accuracy: 0.7349 - val_auc: 0.799
7
Epoch 63/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.4958 - binary_accurac
y: 0.7618 - auc: 0.8389 - val_loss: 0.5587 - val_binary_accuracy: 0.7348 - val_auc: 0.800
4
Epoch 64/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4946 - binary_accurac
y: 0.7629 - auc: 0.8397 - val_loss: 0.5528 - val_binary_accuracy: 0.7305 - val_auc: 0.797
8
Epoch 65/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.4935 - binary_accurac
y: 0.7637 - auc: 0.8405 - val_loss: 0.5528 - val_binary_accuracy: 0.7329 - val_auc: 0.800
8
Epoch 66/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4927 - binary_accurac
y: 0.7641 - auc: 0.8410 - val_loss: 0.5627 - val_binary_accuracy: 0.7333 - val_auc: 0.800
7
Epoch 67/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.4910 - binary_accurac
y: 0.7650 - auc: 0.8423 - val_loss: 0.5558 - val_binary_accuracy: 0.7330 - val_auc: 0.798
6
Epoch 68/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4893 - binary_accurac
y: 0.7661 - auc: 0.8434 - val_loss: 0.5549 - val_binary_accuracy: 0.7318 - val_auc: 0.800
5
Epoch 69/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4884 - binary_accurac
y: 0.7662 - auc: 0.8441 - val_loss: 0.5706 - val_binary_accuracy: 0.7299 - val_auc: 0.796
3
Epoch 70/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4872 - binary_accurac
y: 0.7669 - auc: 0.8450 - val_loss: 0.5735 - val_binary_accuracy: 0.7292 - val_auc: 0.796
3
Epoch 71/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4860 - binary_accurac
y: 0.7679 - auc: 0.8458 - val_loss: 0.5826 - val_binary_accuracy: 0.7308 - val_auc: 0.796
0
Epoch 72/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4848 - binary_accurac
y: 0.7683 - auc: 0.8467 - val_loss: 0.5704 - val_binary_accuracy: 0.7231 - val_auc: 0.795
0
Epoch 73/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4836 - binary_accurac
y: 0.7695 - auc: 0.8474 - val_loss: 0.5668 - val_binary_accuracy: 0.7327 - val_auc: 0.798
9
Epoch 74/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4814 - binary_accurac
y: 0.7704 - auc: 0.8490 - val_loss: 0.6069 - val_binary_accuracy: 0.7005 - val_auc: 0.774
3
Epoch 75/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4805 - binary_accurac
y: 0.7717 - auc: 0.8495 - val_loss: 0.5730 - val_binary_accuracy: 0.7313 - val_auc: 0.796
7
Epoch 76/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.4796 - binary_accurac
y: 0.7721 - auc: 0.8503 - val_loss: 0.5677 - val_binary_accuracy: 0.7267 - val_auc: 0.795
4
Epoch 77/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4781 - binary_accurac
y: 0.7722 - auc: 0.8511 - val_loss: 0.5634 - val_binary_accuracy: 0.7284 - val_auc: 0.793
3
Epoch 78/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.4776 - binary_accurac
y: 0.7725 - auc: 0.8515 - val_loss: 0.5704 - val_binary_accuracy: 0.7246 - val_auc: 0.791
1
Epoch 79/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4764 - binary_accurac
```

```
y: 0.7732 - auc: 0.8522 - val_loss: 0.5711 - val_binary_accuracy: 0.7294 - val_auc: 0.793
3
Epoch 80/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.4740 - binary_accurac
y: 0.7743 - auc: 0.8538 - val_loss: 0.5904 - val_binary_accuracy: 0.7306 - val_auc: 0.797
0
Epoch 81/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4732 - binary_accurac
y: 0.7751 - auc: 0.8544 - val_loss: 0.6301 - val_binary_accuracy: 0.7286 - val_auc: 0.792
5
Epoch 82/100
9949/9949 [==============================] - 89s 9ms/step - loss: 0.4719 - binary_accurac
y: 0.7759 - auc: 0.8553 - val_loss: 0.5757 - val_binary_accuracy: 0.7294 - val_auc: 0.792
9
Epoch 83/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4698 - binary_accurac
y: 0.7773 - auc: 0.8567 - val_loss: 0.5814 - val_binary_accuracy: 0.7241 - val_auc: 0.791
9
Epoch 84/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4685 - binary_accurac
y: 0.7778 - auc: 0.8575 - val_loss: 0.5784 - val_binary_accuracy: 0.7223 - val_auc: 0.787
3
Epoch 85/100
9949/9949 [==============================] - 93s 9ms/step - loss: 0.4678 - binary_accurac
y: 0.7782 - auc: 0.8580 - val_loss: 0.5797 - val_binary_accuracy: 0.7256 - val_auc: 0.790
3
Epoch 86/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4669 - binary_accurac
y: 0.7790 - auc: 0.8587 - val_loss: 0.5782 - val_binary_accuracy: 0.7268 - val_auc: 0.790
6
Epoch 87/100
9949/9949 [==============================] - 93s 9ms/step - loss: 0.4654 - binary_accurac
y: 0.7794 - auc: 0.8595 - val_loss: 0.5944 - val_binary_accuracy: 0.7265 - val_auc: 0.790
8
Epoch 88/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4640 - binary_accurac
y: 0.7809 - auc: 0.8603 - val_loss: 0.5852 - val_binary_accuracy: 0.7211 - val_auc: 0.781
3
Epoch 89/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.4621 - binary_accurac
y: 0.7816 - auc: 0.8616 - val_loss: 0.5911 - val_binary_accuracy: 0.7212 - val_auc: 0.785
1
Epoch 90/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4606 - binary_accurac
y: 0.7820 - auc: 0.8625 - val_loss: 0.6068 - val_binary_accuracy: 0.7271 - val_auc: 0.789
9
Epoch 91/100
9949/9949 [==============================] - 94s 9ms/step - loss: 0.4593 - binary_accurac
y: 0.7833 - auc: 0.8635 - val_loss: 0.5932 - val_binary_accuracy: 0.7273 - val_auc: 0.789
6
Epoch 92/100
9949/9949 [==============================] - 92s 9ms/step - loss: 0.4582 - binary_accurac
y: 0.7843 - auc: 0.8641 - val_loss: 0.5994 - val_binary_accuracy: 0.7263 - val_auc: 0.788
4
Epoch 93/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.4578 - binary_accurac
y: 0.7841 - auc: 0.8643 - val_loss: 0.6020 - val_binary_accuracy: 0.7261 - val_auc: 0.788
8
Epoch 94/100
9949/9949 [==============================] - 90s 9ms/step - loss: 0.4572 - binary_accurac
y: 0.7840 - auc: 0.8648 - val_loss: 0.5951 - val_binary_accuracy: 0.7188 - val_auc: 0.782
9
Epoch 95/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.4545 - binary_accurac
y: 0.7856 - auc: 0.8664 - val_loss: 0.6030 - val_binary_accuracy: 0.7262 - val_auc: 0.784
8
Epoch 96/100
9949/9949 [==============================] - 93s 9ms/step - loss: 0.4532 - binary_accurac
y: 0.7868 - auc: 0.8673 - val_loss: 0.6105 - val_binary_accuracy: 0.7202 - val_auc: 0.781
6
Epoch 97/100
9949/9949 [==============================] - 93s 9ms/step - loss: 0.4529 - binary_accurac
```

```
y: 0.7869 - auc: 0.8675 - val_loss: 0.6019 - val_binary_accuracy: 0.7200 - val_auc: 0.783
8
Epoch 98/100
9949/9949 [==============================] - 93s 9ms/step - loss: 0.4516 - binary_accurac
y: 0.7883 - auc: 0.8683 - val_loss: 0.6005 - val_binary_accuracy: 0.7200 - val_auc: 0.780
5
Epoch 99/100
9949/9949 [==============================] - 93s 9ms/step - loss: 0.4494 - binary_accurac
y: 0.7889 - auc: 0.8696 - val_loss: 0.6056 - val_binary_accuracy: 0.7236 - val_auc: 0.785
9
Epoch 100/100
9949/9949 [==============================] - 91s 9ms/step - loss: 0.4480 - binary_accurac
y: 0.7889 - auc: 0.8703 - val_loss: 0.6145 - val_binary_accuracy: 0.7233 - val_auc: 0.783
1
```

In [ ]:

```
# Evaluate model on test set
y_pred = model.predict(test_x)
roc_auc_score(test_y, y_pred)
```

```
3110/3110 [==============================] - 9s 3ms/step
```

Out[ ]:

```
0.7862643509662941
```

Best ROC AUC score (train) : 0.8703 </br> Best ROC AUC score (validate) : 0.8093 </br> Best ROC AUC score (test) : 0.7863 </br>