

Data_preparation_BHSig260Bengali

November 14, 2024

```
[ ]: import h5py
import numpy as np
from numpy import zeros, ones, asarray, concatenate, array, asarray, pad
import os, sys
from PIL import Image, ImageOps
import cv2
from h5py import File
from math import ceil, floor
import seaborn as sns
```

```
[ ]: from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[ ]: # BASE_DIR = os.pardir
file = os.path.join(
    ~content, 'drive', 'MyDrive', 'BHSig260', 'BHSig260', 'Bengali', 'list.genuine')
print(file)
with open(file) as fil:
    genuine_raw = list(fil.read().split('\n'))
genuine = list()
for lin in genuine_raw:
    genuine.append(lin[lin.find('/') + 1:])#appeding name of the image

file = os.path.join(
    ~content, 'drive', 'MyDrive', 'BHSig260', 'BHSig260', 'Bengali', 'list.forgery')
with open(file) as fil:
    forged_raw = list(fil.read().split('\n'))
forged = list()
for lin in forged_raw:
    forged.append(lin[lin.find('/') + 1:])
```

/content/drive/MyDrive/BHSig260/BHSig260/Bengali/list.genuine

```
[ ]: # %%timeit -n1 -r1
# BASE_DIR = os.pardir
i = 0
x, y = 224,224
```

```

X = np.zeros((54 * 160, y, x, 3), dtype=np.int8)
Y = np.zeros((54 * 160, 1), dtype = np.uint8)
S = np.zeros((54 * 160, 1), dtype = np.uint8)

for indir in os.listdir(os.path.join('/content','drive','MyDrive','BHSig260','BHSig260','Bengali')):
    PATH = os.path.join('/content','drive','MyDrive','BHSig260','BHSig260','Bengali', indir)
    try:
        for infile in os.listdir(PATH):
            im = cv2.imread((os.path.join(PATH,infile)),0)
#           with cv.imread(os.path.join(PATH, infile)) as im:
            X[i] = (cv2.resize(im,(x,y), interpolation = cv2.INTER_AREA)).reshape(y,x,1)
            S[i] = int(indir)
            if infile in genuine:
                Y[i] = 1
            # elif infile in forged:
            #     Y[i] = 0
            i = i + 1
        print(i/54, end = '\t')
    except:
        print("{} might not be a valid directory".format(indir))

```

```

1.0      2.0      3.0      4.0      5.0      6.0      019 might not be a valid
directory
059 might not be a valid directory
7.407407407407407      8.407407407407407      9.407407407407407
10.407407407407407      11.407407407407407      12.407407407407407
13.407407407407407      14.407407407407407      15.407407407407407
16.40740740740741      17.40740740740741      18.40740740740741
19.40740740740741      20.40740740740741      21.40740740740741
22.40740740740741      23.40740740740741      24.40740740740741
25.40740740740741      26.40740740740741      27.40740740740741
28.40740740740741      29.40740740740741      083 might not be a valid
directory
028 might not be a valid directory

```

```
[ ]: # im = cv2.imread(os.path.join(PATH,infile),0)
# im.shape
```

```
[ ]: file = '/content/drive/MyDrive/bhsig260bengali_224x224.h5'
with h5py.File(file, 'w') as hdf:
    hdf.create_dataset('X', data=X)
    hdf.create_dataset('Y', data=Y)
    hdf.create_dataset('S', data=S)
```

```
[ ]: # print(sum(Y[0:54]))
```

```
[ ]: # x, y = 224,224
def xor(x, y):
    z = x ^ y
    return z
```

```
[ ]: # BASE_DIR = os.getcwd()
file = '/content/drive/MyDrive/bhsig260bengali_224x224x1.h5'
# file = os.path.join(BASE_DIR, infile)
with File(file, 'r') as hdf:
    X = array(hdf.get('X'))
    Y = array(hdf.get('Y'))
    S = array(hdf.get('S'))
writers = list()
for i in S.squeeze():
    if i not in writers:
        writers.append(i)
print(writers)
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 0]

```
[ ]: X.shape
```

```
[ ]: (8640, 224, 224, 1)
```

```
[ ]: # Taking only data of 30 writers bcoz of memory limitations
number_of_writers = 30
writers = writers[0:number_of_writers]
X = X[0:number_of_writers*54]
Y = Y[0:number_of_writers*54]
S = S[0:number_of_writers*54]

print(len(writers), len(X), len(Y), len(S))
```

30 1620 1620 1620

```
[ ]: # n_original * n_forged
```

```
[ ]: n_writers = len(writers)
print(n_writers)
n_original = int(Y.sum() // n_writers)
print(n_original)
n_forged = int(len(Y.squeeze()) // n_writers - n_original)
```

```

print(n_forged)
m = int(n_writers * (n_original * n_forged + n_original * (n_original - 1) // 2))
print(m)

```

30
24
30
29880

```

[ ]: S1 = np.zeros((m, X.shape[1], X.shape[2], X.shape[3]), dtype = np.uint8)
S2 = np.zeros((m, X.shape[1], X.shape[2], X.shape[3]), dtype = np.uint8)
y = np.zeros((m, 1), dtype = np.uint8)
w = np.zeros((m, 1), dtype = np.uint8)
l = 0
for writer in writers:
    ind = np.where(S.squeeze() == writer)
    if len(ind) != 1:
        print('Error')
    for i in ind[0]:
        for j in ind[0]:
            if i <= j:
                continue
            if Y.squeeze()[i] == 0 and Y.squeeze()[j] == 0:
                continue
            S1[l] = X[i]
            S2[l] = X[j]
            y[l] = xor(Y[i], Y[j])
            w[l][0] = writer
            l = l + 1
# 'bhsig260hindi_224x224_Preprocessed_drive.h5'
outfile = '/content/drive/MyDrive/'+'bigsig260bengali'+str(X.
    shape[1])+'x'+str(X.shape[2])+'x'+str(X.shape[3])+'_siamese_preprocessed.h5'
print(outfile)
# file = os.path.join(BASE_DIR, outfile)
with File(outfile, 'w') as hdf:
    hdf.create_dataset('S1', data = S1)
    hdf.create_dataset('S2', data = S2)
    hdf.create_dataset('Y', data = y)
    hdf.create_dataset('L', data = w)

```

/content/drive/MyDrive/bigsig260bengali224x224x1_siamese_preprocessed.h5

Data_preparation_BHSig260Hindi

November 14, 2024

```
[ ]: import h5py
import numpy as np
from numpy import zeros, ones, asarray, concatenate, array, asarray, pad
import os, sys
from PIL import Image, ImageOps
import cv2
from h5py import File
from math import ceil, floor
import seaborn as sns
```

```
[ ]: from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).

```
[ ]: # BASE_DIR = os.pardir
file = os.path.join('/  
↳content','drive','MyDrive','BHSig260','BHSig260','Hindi','list.genuine')
print(file)
with open(file) as fil:
    genuine_raw = list(fil.read().split('\n'))
genuine = list()
for lin in genuine_raw:
    genuine.append(lin[lin.find('/') + 1:])#appeding name of the image

file = os.path.join('/content','drive','MyDrive','BHSig260','BHSig260','Hindi',  
↳'list.forgery')
with open(file) as fil:
    forged_raw = list(fil.read().split('\n'))
forged = list()
for lin in forged_raw:
    forged.append(lin[lin.find('/') + 1:])
```

/content/drive/MyDrive/BHSig260/BHSig260/Hindi/list.genuine

```
[ ]: # %%timeit -n1 -r1
# BASE_DIR = os.pardir
i = 0
```

```

x, y = 224,224
X = np.zeros((54 * 160, y, x, 3), dtype=np.int8)
Y = np.zeros((54 * 160, 1), dtype = np.uint8)
S = np.zeros((54 * 160, 1), dtype = np.uint8)

for indir in os.listdir(os.path.join('/content','drive','MyDrive','BHSig260','BHSig260','Bengali')):
    PATH = os.path.join('/content','drive','MyDrive','BHSig260','BHSig260','Bengali', indir)
    try:
        for infile in os.listdir(PATH):
            im = cv2.imread((os.path.join(PATH,infile)),0)
#           with cv.imread((os.path.join(PATH, infile)),0) as im:
            X[i] = (cv2.resize(im,(x,y), interpolation = cv2.INTER_AREA)).reshape(y,x,1)
            S[i] = int(indir)
            if infile in genuine:
                Y[i] = 1
            # elif infile in forged:
            #     Y[i] = 0
            i = i + 1
        print(i/54, end = '\t')
    except:
        print("{} might not be a valid directory".format(indir))

```

```

1.0      2.0      3.0      4.0      5.0      6.0      7.0      8.0      9.0      10.0
086 might not be a valid directory
11.203703703704        12.203703703704        13.203703703704
14.203703703704        15.203703703704

```

```
[ ]: # im = cv2.imread(os.path.join(PATH,infile),0)
# im.shape
```

```
[ ]: file = '/content/drive/MyDrive/bhsig260bengali_224x224.h5'
with h5py.File(file, 'w') as hdf:
    hdf.create_dataset('X', data=X)
    hdf.create_dataset('Y', data=Y)
    hdf.create_dataset('S', data=S)
```

```
[ ]: # print(sum(Y[0:54]))
```

```
[ ]: # x, y = 224,224
def xor(x, y):
    z = x ^ y
    return z
```

```
[ ]: # BASE_DIR = os.getcwd()
file = '/content/drive/MyDrive/bhsig260hindi_224x224x3.h5'
# file = os.path.join(BASE_DIR, infile)
with File(file, 'r') as hdf:
    X = array(hdf.get('X'))
    Y = array(hdf.get('Y'))
    S = array(hdf.get('S'))
writers = list()
for i in S.squeeze():
    if i not in writers:
        writers.append(i)
print(writers)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101,
102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133,
134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149,
150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160]
```

```
[ ]: X.shape
```

```
[ ]: (8640, 224, 224, 3)
```

```
[ ]: # Taking only data of 30 writers bcoz of memory limitations
number_of_writers = 15
writers = writers[0:number_of_writers]
X = X[0:number_of_writers*54]
Y = Y[0:number_of_writers*54]
S = S[0:number_of_writers*54]

print(len(writers), len(X), len(Y), len(S))
```

```
15 810 810 810
```

```
[ ]: # n_original * n_forged
```

```
[ ]: n_writers = len(writers)
print(n_writers)
n_original = int(Y.sum() // n_writers)
print(n_original)
n_forged = int(len(Y.squeeze()) // n_writers - n_original)
print(n_forged)
m = int(n_writers * (n_original * n_forged + n_original * (n_original - 1) // 2))
```

```
print(m)
```

```
15  
24  
30  
14940
```

```
[ ]: S1 = np.zeros((m, X.shape[1], X.shape[2], X.shape[3]), dtype = np.uint8)  
S2 = np.zeros((m, X.shape[1], X.shape[2], X.shape[3]), dtype = np.uint8)  
y = np.zeros((m, 1), dtype = np.uint8)  
w = np.zeros((m, 1), dtype = np.uint8)  
l = 0  
for writer in writers:  
    ind = np.where(S.squeeze() == writer)  
    if len(ind) != 1:  
        print('Error')  
    for i in ind[0]:  
        for j in ind[0]:  
            if i <= j:  
                continue  
            if Y.squeeze()[i] == 0 and Y.squeeze()[j] == 0:  
                continue  
            S1[1] = X[i]  
            S2[1] = X[j]  
            y[1] = xor(Y[i], Y[j])  
            w[1][0] = writer  
            l = l + 1  
            # 'bhsig260hindi_224x224_Preprocessed_drive.h5'  
outfile = '/content/drive/MyDrive/'+'bigsig260'+str(X.shape[1])+'x'+str(X.  
    ↪shape[2])+ 'x' + str(X.shape[3])+'_siamese_preprocessed.h5'  
print(outfile)  
# file = os.path.join(BASE_DIR, outfile)  
with File(outfile, 'w') as hdf:  
    hdf.create_dataset('S1', data = S1)  
    hdf.create_dataset('S2', data = S2)  
    hdf.create_dataset('Y', data = y)  
    hdf.create_dataset('L', data = w)
```

```
/content/drive/MyDrive/bigsig260224x224x3_siamese_preprocessed.h5
```

```
[ ]:
```

Data_preparation_Cedar

November 14, 2024

```
[ ]: import h5py
import numpy as np
from numpy import zeros, ones, asarray, concatenate, array, asarray, pad
import os, sys
from PIL import Image, ImageOps
import cv2
import seaborn as sns
from h5py import File
from math import ceil, floor
```

```
[ ]: from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).

```
[ ]: def process_image(im, img_dim):
    w_new, h_new = img_dim
    im = im.convert('L')
    im = ImageOps.invert(im)
    w, h = im.size
    w_prime = min(w_new, w * h_new // h)
    h_prime = h * w_prime // w
    im = im.resize((w_prime, h_prime))
    img = asarray(im)
    top = floor((h_new - h_prime) / 2)
    bottom = ceil((h_new - h_prime) / 2)
    left = floor((w_new - w_prime) / 2)
    right = ceil((w_new - w_prime) / 2)
    img = pad(img, ((top, bottom), (left, right)))
    img = img.reshape(img.shape[0], img.shape[1], 1)
    return img
```

```
[ ]: def process_batch(X, Y, L, img_dim, Dir, i, value):
    for infile in os.listdir(Dir):
        file = os.path.join(Dir, infile)
        try:
            with Image.open(file) as im:
```

```

        X[i] = process_image(im, img_dim)
        Y[i] = value
        L[i] = infile.split('_')[1]
        i = i + 1
    except Exception as ex:
        template = "An exception of type {0} occurred. Arguments:\n{1!r}"
        message = template.format(type(ex).__name__, ex.args)
        print(message)

```

```
[ ]: img_dim = (224,224)
m = 2640
```

```

[ ]: X = zeros((m,224,224, 1), dtype = np.uint8)
Y = zeros((m, 1), dtype = np.uint8)
L = zeros((m, 1), dtype = np.uint8)
BASE_DIR = os.getcwd()
i = 0
Dir = '/content/drive/MyDrive/CedarDataset/full_org'
process_batch(X, Y, L, img_dim, Dir, i, 1)
i = 1320
Dir = '/content/drive/MyDrive/CedarDataset/full_forg'
process_batch(X, Y, L, img_dim, Dir, i, 0)
i = 2640
outfile = 'cedar_'+str(img_dim[0])+x+str(img_dim[1])+'.h5'
print(outfile)
file = os.path.join(BASE_DIR, outfile)
with File(outfile, 'w') as hdf:
    hdf.create_dataset('X', data = X)
    hdf.create_dataset('Y', data = Y)
    hdf.create_dataset('L', data = L)

```

cedar_224x224.h5

```
[ ]: def xor(x, y):
    z = x ^ y
    return z
```

```

[ ]: img_dim = (224,224)
m = 2640
BASE_DIR = os.getcwd()
infile = 'cedar_'+str(img_dim[0])+x+str(img_dim[1])+'.h5'
file = os.path.join(BASE_DIR, infile)
with File(file, 'r') as hdf:
    X = array(hdf.get('X'))
    Y = array(hdf.get('Y'))
    L = array(hdf.get('L'))
writers = list()
for i in L.squeeze():

```

```

if i not in writers:
    writers.append(i)
print(writers)

```

```
[30, 25, 23, 24, 19, 1, 27, 2, 29, 28, 18, 21, 26, 20, 22, 40, 36, 37, 3, 35,
39, 38, 33, 32, 31, 34, 47, 48, 41, 45, 49, 44, 4, 43, 42, 46, 50, 5, 55, 6, 53,
52, 51, 54, 7, 8, 9, 17, 14, 15, 13, 12, 11, 10, 16]
```

```

[ ]: n_writers = len(writers)
print(n_writers)
n_original = int(Y.sum() // n_writers)
print(n_original)
n_forged = int(len(Y.squeeze()) // n_writers - n_original)
print(n_forged)
m = int(n_writers * (n_original * n_forged + n_original * (n_original - 1) // 2))
print(m)

```

```
55
24
24
46860
```

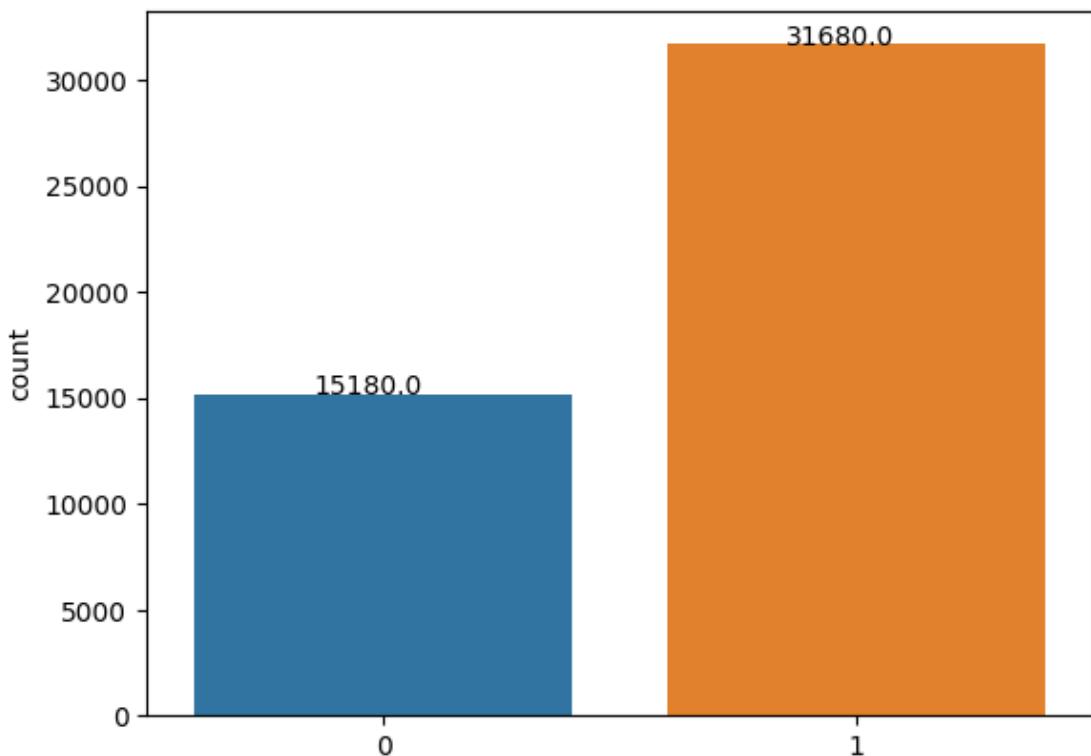
```

[ ]: S1 = np.zeros((m, X.shape[1], X.shape[2], X.shape[3]), dtype = np.uint8)
S2 = np.zeros((m, X.shape[1], X.shape[2], X.shape[3]), dtype = np.uint8)
y = np.zeros((m, 1), dtype = np.uint8)
w = np.zeros((m, 1), dtype = np.uint8)
l = 0
for writer in writers:
    ind = np.where(L.squeeze() == writer)
    if len(ind) != 1:
        print('Error')
    for i in ind[0]:
        for j in ind[0]:
            if i <= j:
                continue
            if Y.squeeze()[i] == 0 and Y.squeeze()[j] == 0:
                continue
            S1[l] = X[i]
            S2[l] = X[j]
            y[l] = xor(Y[i], Y[j])
            w[l][0] = writer
            l = l + 1
outfile = '/content/drive/MyDrive/CedarDataset/' +
        'cedar_'+str(img_dim[0])+'x'+str(img_dim[1])+'_siamese.h5'
print(outfile)
# file =
with File(outfile, 'w') as hdf:
```

```
hdf.create_dataset('S1', data = S1)
hdf.create_dataset('S2', data = S2)
hdf.create_dataset('Y', data = y)
hdf.create_dataset('L', data = w)
```

```
/content/drive/MyDrive/CedarDataset/cedar_224x224_siamese.h5
```

```
[ ]: ax = sns.countplot(x = y.reshape(y.shape[0]))
for p in ax.patches:
    ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x() + 0.25, p.get_height() + 0.01))
```



Model1_SCNN_BHSig260BengaliTraining

November 14, 2024

1 Shallow Convolution Model

This model has only 3 convolution layers with filters of size 7, 5, 3 and 3 max pooling layers.

```
[ ]: from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import LeakyReLU, Softmax
from keras.layers import Conv2D, Activation, Input, Dropout, Lambda, Flatten, Dense
from keras.layers import Dense, Flatten, Reshape, Activation, MaxPool2D
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model

[ ]: import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

```
[ ]: from google.colab import drive  
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
There are 3 databases available with me. * BhSig100Bengali * BhSig160Hindi * Cedar * Sig-comp2011
```

```
We can select any of the three available.
```

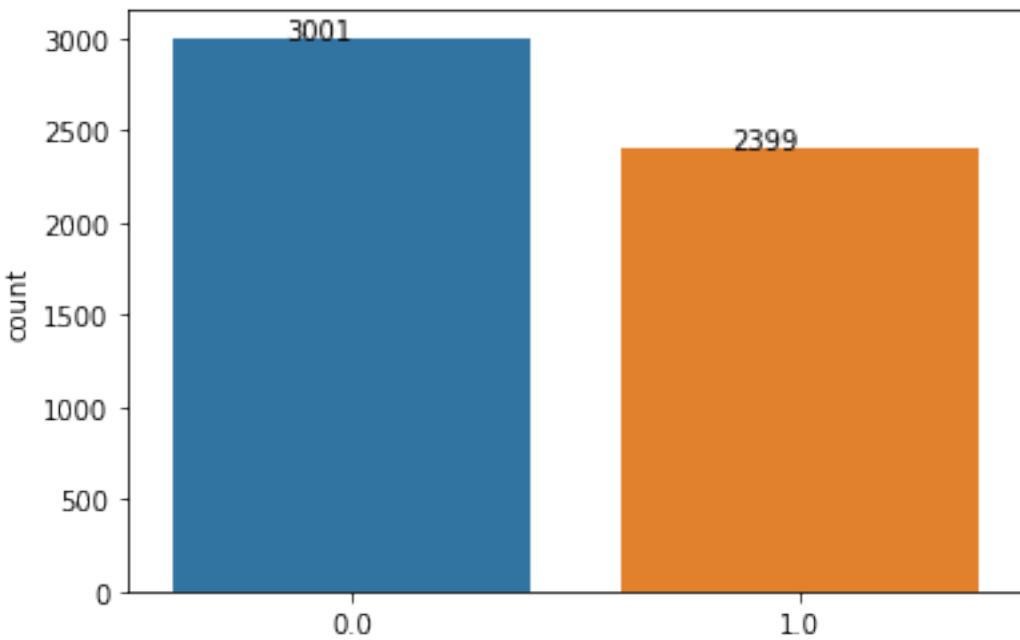
```
[ ]: # database = input('Database Name: ')  
# Models\model1_scnn\database\bhsig100bengali_128x64.h5  
# database = database.lower() + '_128x64.h5'  
# BASE_DIR = os.path.join(os.pardir, os.pardir)  
metadata = {}  
file = "/content/drive/MyDrive/bhsig260bengali_128x643.h5"  
print(file)  
try:  
    with File(file, 'r') as hdf:  
        X = np.array(hdf.get('X'))  
        Y = np.array(hdf.get('Y'))  
        S = np.array(hdf.get('S'))  
except Exception as ex:  
    template = "An exception of type {0} occurred. Arguments:\n{1!r}"  
    message = template.format(type(ex).__name__, ex.args)  
    print(message)  
X = X / 255.0  
Y = Y * 1.0  
# Y = one_hot(Y * 1.0, depth=2)  
# Y = reshape(Y, (-1, 2))  
print("Feature shape =", X.shape)  
print("Label shape =", Y.shape)
```

```
/content/drive/MyDrive/bhsig260bengali_128x643.h5
```

```
Feature shape = (5400, 64, 128, 3)
```

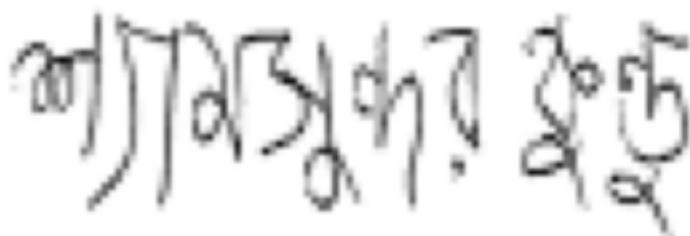
```
Label shape = (5400, 1)
```

```
[ ]: ax = sns.countplot(x = Y.reshape(Y.shape[0]))  
for p in ax.patches:  
    ax.annotate('{:}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.  
    ↵01))
```

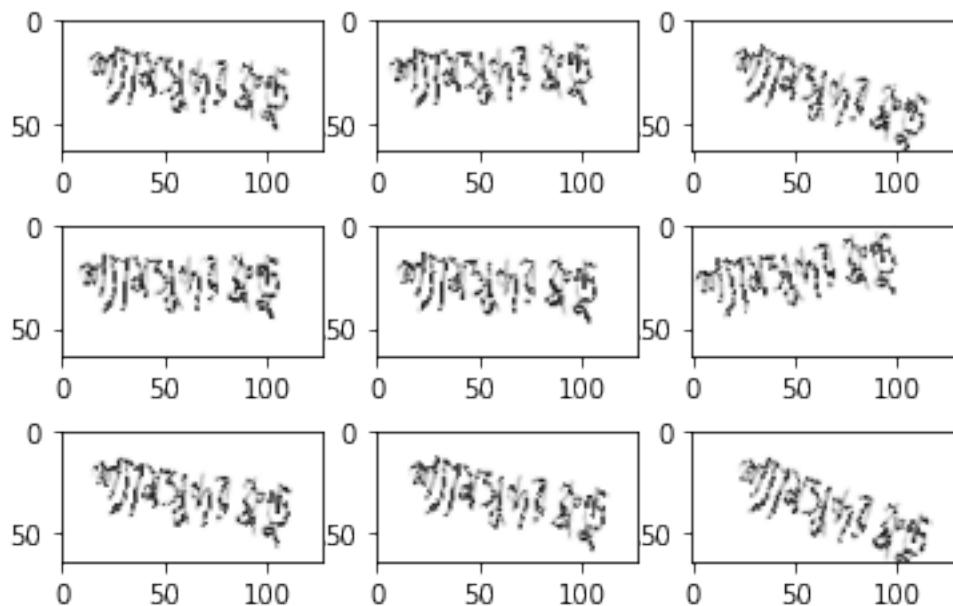


```
[ ]: x = np.random.randint(0,X.shape[0])
img = X[x]
print(x)
print(img.shape)
plt.imshow((img * 255).astype(np.uint8))
plt.axis("off")
plt.show()
```

3837
(64, 128, 3)



```
[ ]: # load the image
# img = load_img('bird.jpg')
# convert to numpy array
data = img_to_array(img *255)
# expand dimension to one sample
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(rotation_range=20)
# prepare iterator
it = datagen.flow(samples, batch_size=1)
# generate samples and plot
for i in range(9):
    # define subplot
    plt.subplot(330 + 1 + i)
    # generate batch of images
    batch = it.next()
    # convert to unsigned integers for viewing
    image = batch[0].astype('uint8')
    # plot raw pixel data
    plt.imshow(image)
# pyplot.axis("off")
# show the figure
plt.show()
```



```
[ ]: import sklearn
class_weights = sklearn.utils.class_weight.compute_class_weight(class_weight = "balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict
```

```
[ ]: {0: 0.8997000999666778, 1: 1.1254689453939142}
```

```
[ ]: seed=randint(10)
metadata['split_seed']=seed
print('seed =', seed)
indices = permutation(X.shape[0])
# print(X)
same = True
if same:
    m = int(0.70 * X.shape[0])
    n = int(0.15 * X.shape[0])
    training_id, validation_id, test_id = indices[:m], indices[m: m + n], indices[m+n:]
    X_train, X_test, X_validate = X[training_id], X[test_id], X[validation_id]
    Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]
else:
    m = int(0.70 * X.shape[0])
    training_id, validation_id = indices[:m], indices[m:]
    X_train, X_validate = X[training_id], X[validation_id]
    Y_train, Y_validate = Y[training_id], Y[validation_id]
print("Shape of Features in training set =", X_train.shape)
print("Shape of Labels in training set =", Y_train.shape)
print("Shape of Features in testing set =", X_test.shape)
print("Shape of Labels in testing set =", Y_test.shape)

del Y,X
```

```
seed = 3
Shape of Features in training set = (3779, 64, 128, 3)
Shape of Labels in training set = (3779, 1)
Shape of Features in testing set = (811, 64, 128, 3)
Shape of Labels in testing set = (811, 1)
```

```
[ ]: #One hot Encoding
Y_train = one_hot(Y_train, depth=2)
Y_train = reshape(Y_train, (-1, 2))

Y_test = one_hot(Y_test, depth=2)
Y_test = reshape(Y_test, (-1, 2))

Y_validate = one_hot(Y_validate, depth=2)
Y_validate = reshape(Y_validate, (-1, 2))
```

```
[ ]: print("Shape of Labels in training set =", Y_train.shape)

Shape of Labels in training set = (3779, 2)

[ ]: # X_train, X_test, Y_train, Y_test = train_test_split(X, Y,train_size = 0.7,random_state = 42)

[ ]: # X_test, X_validate, Y_test, Y_validate = train_test_split(X_test, Y_test,train_size = 0.5, random_state = 42)

[ ]: # del Y,X

[ ]: seed =randint(10)
metadata['init_seed']=seed
print('seed='+str(seed))
weights1 = np.random.rand(7,7,3,40)
bias1 = np.random.rand(40)
weights2 = np.random.rand(5,5,40,30)
bias2 = np.random.rand(30)
weights3 = np.random.rand(3,3,30,20)
bias3 = np.random.rand(20)

model = Sequential()
model.add(Input(shape = (64, 128, 3),name='input'))
conv1 = Conv2D(40, 7, strides = 1,activation='relu',name='conv1',use_bias =True)
model.add(conv1)
model.add(BatchNormalization(epsilon = 1e-08, axis = 1,name = 'batchnorm1',momentum = 0.9))
model.add(MaxPool2D(strides=2,name = "Pool1"))

conv2 = Conv2D(30, 5,strides = 1, activation='relu', name='conv2',use_bias =True)
model.add(conv2)
model.add(BatchNormalization(epsilon = 1e-08, axis = 1,name = 'batchnorm2',momentum = 0.9))
model.add(MaxPool2D(strides=3,name = "Pool2"))

conv3 = Conv2D(20, 3, strides = 1,activation='relu',name='conv3',use_bias =True)
model.add(conv3)
model.add(BatchNormalization(epsilon = 1e-08, axis = 1,name = 'batchnorm3',momentum = 0.9))
model.add(MaxPool2D(strides=3,name = "Pool3"))

model.add(Flatten())
```

```
model.add(Dense(32, activation='relu',  
    ↪kernel_initializer=glorot_uniform(seed=seed), kernel_regularizer='l2'))  
model.add(Dense(2, activation='softmax',  
    ↪kernel_initializer=glorot_uniform(seed=seed), kernel_regularizer='l2'))
```

```
seed=6
```

```
[ ]: model.layers
```

```
[ ]: [<keras.layers.convolutional.conv2d.Conv2D at 0x7f49409cd280>,  
       <keras.layers.normalization.batch_normalization.BatchNormalization at  
       0x7f49403b2c70>,  
       <keras.layers.pooling.max_pooling2d.MaxPooling2D at 0x7f49403b2f70>,  
       <keras.layers.convolutional.conv2d.Conv2D at 0x7f49403b2ca0>,  
       <keras.layers.normalization.batch_normalization.BatchNormalization at  
       0x7f49403bee20>,  
       <keras.layers.pooling.max_pooling2d.MaxPooling2D at 0x7f49403b2d90>,  
       <keras.layers.convolutional.conv2d.Conv2D at 0x7f49403cb4f0>,  
       <keras.layers.normalization.batch_normalization.BatchNormalization at  
       0x7f4940332580>,  
       <keras.layers.pooling.max_pooling2d.MaxPooling2D at 0x7f49403bed30>,  
       <keras.layers.reshape.flatten.Flatten at 0x7f49403b2f10>,  
       <keras.layers.core.dense.Dense at 0x7f4940315d30>,  
       <keras.layers.core.dense.Dense at 0x7f4940357130>]
```

```
[ ]: model.layers[0].set_weights([weights1,bias1])  
model.layers[3].set_weights([weights2,bias2])  
model.layers[6].set_weights([weights3,bias3])
```

```
[ ]: model.layers[6].get_weights()[0].shape
```

```
[ ]: (3, 3, 30, 20)
```

```
[ ]: # len(conv1.get_weights())  
# [len(a) for a in conv1.get_weights()]
```

The summary of the model is given below:

```
[ ]: print(model.summary())
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv1 (Conv2D)	(None, 58, 122, 40)	5920
batchnorm1 (BatchNormalizat ion)	(None, 58, 122, 40)	232

Pool1 (MaxPooling2D)	(None, 29, 61, 40)	0
conv2 (Conv2D)	(None, 25, 57, 30)	30030
batchnorm2 (BatchNormalizat ion)	(None, 25, 57, 30)	100
Pool2 (MaxPooling2D)	(None, 8, 19, 30)	0
conv3 (Conv2D)	(None, 6, 17, 20)	5420
batchnorm3 (BatchNormalizat ion)	(None, 6, 17, 20)	24
Pool3 (MaxPooling2D)	(None, 2, 6, 20)	0
flatten (Flatten)	(None, 240)	0
dense (Dense)	(None, 32)	7712
dense_1 (Dense)	(None, 2)	66
=====		
Total params:	49,504	
Trainable params:	49,326	
Non-trainable params:	178	

None		

In comile process * Optimizer is *Adam* * Loss is *Binary Cross Entropy Loss* * Metrics involve *Binary Accuracy*

```
[ ]: with tf.device('/device:GPU:0'):
    SGD = tf.keras.optimizers.SGD(learning_rate = 1e-04,momentum = 0.9,clipvalue= 7)
    model.compile(optimizer = SGD, loss='binary_crossentropy',metrics=['accuracy','AUC'])
    #, 'TruePositives', 'TrueNegatives', 'FalsePositives', 'FalseNegatives'])
```

1.1 Batch Size Testing

```
[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),  
epochs = 3,  
batch_size=8)
```

Epoch 1/3
648/648 [=====] - 6s 7ms/step - loss: 1.2955 -
binary_accuracy: 0.4379 - val_loss: 1.2547 - val_binary_accuracy: 0.4433

```
Epoch 2/3
648/648 [=====] - 5s 7ms/step - loss: 1.2257 -
binary_accuracy: 0.4379 - val_loss: 1.2153 - val_binary_accuracy: 0.4433
Epoch 3/3
648/648 [=====] - 4s 7ms/step - loss: 1.1953 -
binary_accuracy: 0.4379 - val_loss: 1.1880 - val_binary_accuracy: 0.4433
[ ]: <keras.callbacks.History at 0x7fe7b203df50>
[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
batch_size=16)
378/378 [=====] - 6s 13ms/step - loss: 1.5875 -
binary_accuracy: 0.4383 - auc: 0.5000 - val_loss: 1.6461 - val_binary_accuracy:
0.4406 - val_auc: 0.5000
[ ]: <keras.callbacks.History at 0x7f02b93e4c90>
[ ]: model.fit(X_train, Y_train, epochs = 10, batch_size = 32,validation_data=(X_validate, Y_validate),
shuffle=True,class_weight = class_weight_dict)
Epoch 1/10
189/189 [=====] - 4s 19ms/step - loss: 1.2786 -
accuracy: 0.5982 - auc: 0.6303 - val_loss: 1.2695 - val_accuracy: 0.6088 -
val_auc: 0.6411
Epoch 2/10
189/189 [=====] - 3s 18ms/step - loss: 1.2592 -
accuracy: 0.6068 - auc: 0.6442 - val_loss: 1.2550 - val_accuracy: 0.6096 -
val_auc: 0.6480
Epoch 3/10
189/189 [=====] - 3s 18ms/step - loss: 1.2427 -
accuracy: 0.6197 - auc: 0.6619 - val_loss: 1.2426 - val_accuracy: 0.6235 -
val_auc: 0.6662
Epoch 4/10
189/189 [=====] - 3s 18ms/step - loss: 1.2303 -
accuracy: 0.6204 - auc: 0.6707 - val_loss: 1.2340 - val_accuracy: 0.6157 -
val_auc: 0.6640
Epoch 5/10
189/189 [=====] - 3s 18ms/step - loss: 1.2223 -
accuracy: 0.6217 - auc: 0.6740 - val_loss: 1.2245 - val_accuracy: 0.6235 -
val_auc: 0.6730
Epoch 6/10
189/189 [=====] - 3s 18ms/step - loss: 1.2153 -
accuracy: 0.6195 - auc: 0.6763 - val_loss: 1.2156 - val_accuracy: 0.6296 -
val_auc: 0.6782
Epoch 7/10
189/189 [=====] - 3s 18ms/step - loss: 1.2071 -
accuracy: 0.6285 - auc: 0.6808 - val_loss: 1.2139 - val_accuracy: 0.6250 -
```

```
val_auc: 0.6685
Epoch 8/10
189/189 [=====] - 4s 19ms/step - loss: 1.1999 -
accuracy: 0.6348 - auc: 0.6831 - val_loss: 1.2049 - val_accuracy: 0.6273 -
val_auc: 0.6782
Epoch 9/10
189/189 [=====] - 4s 19ms/step - loss: 1.1924 -
accuracy: 0.6326 - auc: 0.6876 - val_loss: 1.2009 - val_accuracy: 0.6304 -
val_auc: 0.6769
Epoch 10/10
189/189 [=====] - 4s 20ms/step - loss: 1.1878 -
accuracy: 0.6348 - auc: 0.6875 - val_loss: 1.1969 - val_accuracy: 0.6289 -
val_auc: 0.6759
```

```
[ ]: <keras.callbacks.History at 0x7f72905e3390>
```

```
[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
batch_size=64)
```

```
95/95 [=====] - 4s 28ms/step - loss: 9644.4453 -
binary_accuracy: 0.4395 - val_loss: 2332.6816 - val_binary_accuracy: 0.4591
```

```
[ ]: <keras.callbacks.History at 0x7f2f404ee050>
```

```
[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
batch_size=128)
```

```
48/48 [=====] - 4s 51ms/step - loss: 10394.5078 -
binary_accuracy: 0.4395 - val_loss: 4305.1074 - val_binary_accuracy: 0.4591
```

```
[ ]: <keras.callbacks.History at 0x7f2f403fa810>
```

1.2 In fit process

- epochs = 20
- batch size = 32

```
[ ]: with tf.device('/device:GPU:0'):
    SGD = tf.keras.optimizers.SGD(learning_rate = 1e-04,momentum = 0.9,clipvalue_
    ↵= 7)
    model.compile(optimizer = SGD, loss='binary_crossentropy',_
    ↵metrics=['accuracy','AUC'])
    #, 'TruePositives', 'TrueNegatives', 'FalsePositives', 'FalseNegatives'])
```

```
[ ]: history = model.fit(X_train, Y_train, epochs = 1000, batch_size = 32,_
    ↵validation_data=(X_validate, Y_validate),shuffle=True
                ,class_weight = class_weight_dict)
print('training done')
```

```
Epoch 1/1000
119/119 [=====] - 11s 27ms/step - loss: 1.5365 -
accuracy: 0.4917 - auc: 0.5003 - val_loss: 1.3989 - val_accuracy: 0.5346 -
val_auc: 0.5359
Epoch 2/1000
119/119 [=====] - 3s 24ms/step - loss: 1.3527 -
accuracy: 0.5464 - auc: 0.5822 - val_loss: 1.3474 - val_accuracy: 0.5444 -
val_auc: 0.5865
Epoch 3/1000
119/119 [=====] - 3s 26ms/step - loss: 1.3073 -
accuracy: 0.5673 - auc: 0.6254 - val_loss: 1.3284 - val_accuracy: 0.5654 -
val_auc: 0.6140
Epoch 4/1000
119/119 [=====] - 3s 23ms/step - loss: 1.2846 -
accuracy: 0.5827 - auc: 0.6437 - val_loss: 1.3051 - val_accuracy: 0.5728 -
val_auc: 0.6331
Epoch 5/1000
119/119 [=====] - 3s 22ms/step - loss: 1.2679 -
accuracy: 0.6134 - auc: 0.6585 - val_loss: 1.2938 - val_accuracy: 0.6235 -
val_auc: 0.6472
Epoch 6/1000
119/119 [=====] - 2s 19ms/step - loss: 1.2586 -
accuracy: 0.6298 - auc: 0.6698 - val_loss: 1.2773 - val_accuracy: 0.6420 -
val_auc: 0.6590
Epoch 7/1000
119/119 [=====] - 2s 19ms/step - loss: 1.2465 -
accuracy: 0.6335 - auc: 0.6756 - val_loss: 1.2664 - val_accuracy: 0.6370 -
val_auc: 0.6584
Epoch 8/1000
119/119 [=====] - 3s 23ms/step - loss: 1.2364 -
accuracy: 0.6420 - auc: 0.6812 - val_loss: 1.2599 - val_accuracy: 0.6432 -
val_auc: 0.6714
Epoch 9/1000
119/119 [=====] - 2s 20ms/step - loss: 1.2313 -
accuracy: 0.6396 - auc: 0.6839 - val_loss: 1.2517 - val_accuracy: 0.6469 -
val_auc: 0.6702
Epoch 10/1000
119/119 [=====] - 3s 21ms/step - loss: 1.2183 -
accuracy: 0.6451 - auc: 0.6937 - val_loss: 1.2504 - val_accuracy: 0.6432 -
val_auc: 0.6679
Epoch 11/1000
119/119 [=====] - 3s 22ms/step - loss: 1.2157 -
accuracy: 0.6491 - auc: 0.6930 - val_loss: 1.2392 - val_accuracy: 0.6444 -
val_auc: 0.6729
Epoch 12/1000
119/119 [=====] - 2s 20ms/step - loss: 1.2108 -
accuracy: 0.6483 - auc: 0.6969 - val_loss: 1.2296 - val_accuracy: 0.6432 -
val_auc: 0.6737
```

```
Epoch 13/1000
119/119 [=====] - 3s 22ms/step - loss: 1.2066 -
accuracy: 0.6433 - auc: 0.6948 - val_loss: 1.2315 - val_accuracy: 0.6296 -
val_auc: 0.6685
Epoch 14/1000
119/119 [=====] - 3s 23ms/step - loss: 1.1986 -
accuracy: 0.6502 - auc: 0.7032 - val_loss: 1.2229 - val_accuracy: 0.6407 -
val_auc: 0.6852
Epoch 15/1000
119/119 [=====] - 2s 20ms/step - loss: 1.1936 -
accuracy: 0.6512 - auc: 0.7063 - val_loss: 1.2245 - val_accuracy: 0.6210 -
val_auc: 0.6696
Epoch 16/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1904 -
accuracy: 0.6510 - auc: 0.7031 - val_loss: 1.2193 - val_accuracy: 0.6235 -
val_auc: 0.6729
Epoch 17/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1848 -
accuracy: 0.6494 - auc: 0.7068 - val_loss: 1.2078 - val_accuracy: 0.6444 -
val_auc: 0.6887
Epoch 18/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1792 -
accuracy: 0.6552 - auc: 0.7117 - val_loss: 1.2055 - val_accuracy: 0.6395 -
val_auc: 0.6891
Epoch 19/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1771 -
accuracy: 0.6491 - auc: 0.7089 - val_loss: 1.2015 - val_accuracy: 0.6296 -
val_auc: 0.6849
Epoch 20/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1723 -
accuracy: 0.6518 - auc: 0.7101 - val_loss: 1.1975 - val_accuracy: 0.6309 -
val_auc: 0.6865
Epoch 21/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1688 -
accuracy: 0.6552 - auc: 0.7126 - val_loss: 1.1904 - val_accuracy: 0.6469 -
val_auc: 0.6971
Epoch 22/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1665 -
accuracy: 0.6597 - auc: 0.7096 - val_loss: 1.1859 - val_accuracy: 0.6444 -
val_auc: 0.6940
Epoch 23/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1583 -
accuracy: 0.6520 - auc: 0.7166 - val_loss: 1.1813 - val_accuracy: 0.6481 -
val_auc: 0.7002
Epoch 24/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1569 -
accuracy: 0.6547 - auc: 0.7161 - val_loss: 1.1774 - val_accuracy: 0.6383 -
val_auc: 0.6977
```

```
Epoch 25/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1554 -
accuracy: 0.6557 - auc: 0.7141 - val_loss: 1.1735 - val_accuracy: 0.6432 -
val_auc: 0.7002
Epoch 26/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1497 -
accuracy: 0.6571 - auc: 0.7188 - val_loss: 1.1717 - val_accuracy: 0.6358 -
val_auc: 0.6979
Epoch 27/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1500 -
accuracy: 0.6565 - auc: 0.7123 - val_loss: 1.1682 - val_accuracy: 0.6383 -
val_auc: 0.6965
Epoch 28/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1415 -
accuracy: 0.6578 - auc: 0.7212 - val_loss: 1.1682 - val_accuracy: 0.6296 -
val_auc: 0.6931
Epoch 29/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1408 -
accuracy: 0.6602 - auc: 0.7186 - val_loss: 1.1620 - val_accuracy: 0.6457 -
val_auc: 0.7029
Epoch 30/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1386 -
accuracy: 0.6536 - auc: 0.7196 - val_loss: 1.1582 - val_accuracy: 0.6395 -
val_auc: 0.7020
Epoch 31/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1357 -
accuracy: 0.6618 - auc: 0.7204 - val_loss: 1.1597 - val_accuracy: 0.6370 -
val_auc: 0.7004
Epoch 32/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1310 -
accuracy: 0.6539 - auc: 0.7201 - val_loss: 1.1543 - val_accuracy: 0.6346 -
val_auc: 0.6997
Epoch 33/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1324 -
accuracy: 0.6645 - auc: 0.7178 - val_loss: 1.1476 - val_accuracy: 0.6444 -
val_auc: 0.7068
Epoch 34/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1250 -
accuracy: 0.6642 - auc: 0.7234 - val_loss: 1.1513 - val_accuracy: 0.6284 -
val_auc: 0.6928
Epoch 35/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1226 -
accuracy: 0.6600 - auc: 0.7240 - val_loss: 1.1393 - val_accuracy: 0.6531 -
val_auc: 0.7102
Epoch 36/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1194 -
accuracy: 0.6663 - auc: 0.7247 - val_loss: 1.1386 - val_accuracy: 0.6432 -
val_auc: 0.7072
```

```
Epoch 37/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1144 -
accuracy: 0.6626 - auc: 0.7285 - val_loss: 1.1376 - val_accuracy: 0.6420 -
val_auc: 0.7040
Epoch 38/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1158 -
accuracy: 0.6589 - auc: 0.7208 - val_loss: 1.1343 - val_accuracy: 0.6420 -
val_auc: 0.7061
Epoch 39/1000
119/119 [=====] - 2s 20ms/step - loss: 1.1121 -
accuracy: 0.6602 - auc: 0.7226 - val_loss: 1.1323 - val_accuracy: 0.6444 -
val_auc: 0.7060
Epoch 40/1000
119/119 [=====] - 3s 22ms/step - loss: 1.1089 -
accuracy: 0.6687 - auc: 0.7280 - val_loss: 1.1369 - val_accuracy: 0.6333 -
val_auc: 0.6957
Epoch 41/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1032 -
accuracy: 0.6692 - auc: 0.7295 - val_loss: 1.1312 - val_accuracy: 0.6321 -
val_auc: 0.6977
Epoch 42/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1026 -
accuracy: 0.6679 - auc: 0.7284 - val_loss: 1.1241 - val_accuracy: 0.6444 -
val_auc: 0.7056
Epoch 43/1000
119/119 [=====] - 2s 19ms/step - loss: 1.1006 -
accuracy: 0.6634 - auc: 0.7274 - val_loss: 1.1225 - val_accuracy: 0.6444 -
val_auc: 0.7058
Epoch 44/1000
119/119 [=====] - 2s 20ms/step - loss: 1.0977 -
accuracy: 0.6663 - auc: 0.7278 - val_loss: 1.1257 - val_accuracy: 0.6321 -
val_auc: 0.6995
Epoch 45/1000
119/119 [=====] - 3s 22ms/step - loss: 1.0948 -
accuracy: 0.6621 - auc: 0.7284 - val_loss: 1.1121 - val_accuracy: 0.6568 -
val_auc: 0.7136
Epoch 46/1000
119/119 [=====] - 2s 20ms/step - loss: 1.0911 -
accuracy: 0.6690 - auc: 0.7305 - val_loss: 1.1168 - val_accuracy: 0.6407 -
val_auc: 0.7024
Epoch 47/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0884 -
accuracy: 0.6631 - auc: 0.7296 - val_loss: 1.1126 - val_accuracy: 0.6444 -
val_auc: 0.7065
Epoch 48/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0878 -
accuracy: 0.6658 - auc: 0.7296 - val_loss: 1.1162 - val_accuracy: 0.6370 -
val_auc: 0.6979
```

```
Epoch 49/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0842 -
accuracy: 0.6679 - auc: 0.7314 - val_loss: 1.1076 - val_accuracy: 0.6469 -
val_auc: 0.7072
Epoch 50/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0815 -
accuracy: 0.6687 - auc: 0.7315 - val_loss: 1.1076 - val_accuracy: 0.6432 -
val_auc: 0.7045
Epoch 51/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0813 -
accuracy: 0.6695 - auc: 0.7302 - val_loss: 1.0974 - val_accuracy: 0.6531 -
val_auc: 0.7123
Epoch 52/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0752 -
accuracy: 0.6703 - auc: 0.7354 - val_loss: 1.0980 - val_accuracy: 0.6531 -
val_auc: 0.7121
Epoch 53/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0749 -
accuracy: 0.6610 - auc: 0.7296 - val_loss: 1.0972 - val_accuracy: 0.6519 -
val_auc: 0.7101
Epoch 54/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0733 -
accuracy: 0.6676 - auc: 0.7303 - val_loss: 1.0923 - val_accuracy: 0.6519 -
val_auc: 0.7098
Epoch 55/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0687 -
accuracy: 0.6666 - auc: 0.7342 - val_loss: 1.0926 - val_accuracy: 0.6481 -
val_auc: 0.7072
Epoch 56/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0673 -
accuracy: 0.6719 - auc: 0.7330 - val_loss: 1.0836 - val_accuracy: 0.6679 -
val_auc: 0.7189
Epoch 57/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0633 -
accuracy: 0.6724 - auc: 0.7332 - val_loss: 1.0845 - val_accuracy: 0.6617 -
val_auc: 0.7152
Epoch 58/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0600 -
accuracy: 0.6758 - auc: 0.7356 - val_loss: 1.0847 - val_accuracy: 0.6469 -
val_auc: 0.7087
Epoch 59/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0603 -
accuracy: 0.6743 - auc: 0.7353 - val_loss: 1.0804 - val_accuracy: 0.6457 -
val_auc: 0.7115
Epoch 60/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0589 -
accuracy: 0.6698 - auc: 0.7315 - val_loss: 1.0891 - val_accuracy: 0.6321 -
val_auc: 0.6973
```

```
Epoch 61/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0540 -
accuracy: 0.6668 - auc: 0.7338 - val_loss: 1.0858 - val_accuracy: 0.6333 -
val_auc: 0.6998
Epoch 62/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0508 -
accuracy: 0.6735 - auc: 0.7367 - val_loss: 1.0744 - val_accuracy: 0.6481 -
val_auc: 0.7104
Epoch 63/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0488 -
accuracy: 0.6711 - auc: 0.7377 - val_loss: 1.0718 - val_accuracy: 0.6519 -
val_auc: 0.7125
Epoch 64/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0471 -
accuracy: 0.6727 - auc: 0.7372 - val_loss: 1.0646 - val_accuracy: 0.6630 -
val_auc: 0.7203
Epoch 65/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0454 -
accuracy: 0.6716 - auc: 0.7355 - val_loss: 1.0711 - val_accuracy: 0.6457 -
val_auc: 0.7084
Epoch 66/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0442 -
accuracy: 0.6716 - auc: 0.7368 - val_loss: 1.0675 - val_accuracy: 0.6469 -
val_auc: 0.7095
Epoch 67/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0418 -
accuracy: 0.6658 - auc: 0.7350 - val_loss: 1.0635 - val_accuracy: 0.6519 -
val_auc: 0.7137
Epoch 68/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0396 -
accuracy: 0.6692 - auc: 0.7345 - val_loss: 1.0664 - val_accuracy: 0.6383 -
val_auc: 0.7068
Epoch 69/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0363 -
accuracy: 0.6790 - auc: 0.7389 - val_loss: 1.0599 - val_accuracy: 0.6593 -
val_auc: 0.7152
Epoch 70/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0341 -
accuracy: 0.6737 - auc: 0.7386 - val_loss: 1.0570 - val_accuracy: 0.6519 -
val_auc: 0.7138
Epoch 71/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0326 -
accuracy: 0.6806 - auc: 0.7392 - val_loss: 1.0526 - val_accuracy: 0.6605 -
val_auc: 0.7179
Epoch 72/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0306 -
accuracy: 0.6748 - auc: 0.7381 - val_loss: 1.0573 - val_accuracy: 0.6469 -
val_auc: 0.7097
```

```
Epoch 73/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0274 -
accuracy: 0.6716 - auc: 0.7393 - val_loss: 1.0544 - val_accuracy: 0.6407 -
val_auc: 0.7075
Epoch 74/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0232 -
accuracy: 0.6748 - auc: 0.7411 - val_loss: 1.0452 - val_accuracy: 0.6630 -
val_auc: 0.7217
Epoch 75/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0229 -
accuracy: 0.6761 - auc: 0.7390 - val_loss: 1.0500 - val_accuracy: 0.6519 -
val_auc: 0.7109
Epoch 76/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0181 -
accuracy: 0.6756 - auc: 0.7449 - val_loss: 1.0451 - val_accuracy: 0.6506 -
val_auc: 0.7149
Epoch 77/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0183 -
accuracy: 0.6766 - auc: 0.7396 - val_loss: 1.0415 - val_accuracy: 0.6580 -
val_auc: 0.7153
Epoch 78/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0182 -
accuracy: 0.6756 - auc: 0.7385 - val_loss: 1.0451 - val_accuracy: 0.6519 -
val_auc: 0.7073
Epoch 79/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0172 -
accuracy: 0.6748 - auc: 0.7372 - val_loss: 1.0386 - val_accuracy: 0.6519 -
val_auc: 0.7145
Epoch 80/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0139 -
accuracy: 0.6769 - auc: 0.7384 - val_loss: 1.0438 - val_accuracy: 0.6383 -
val_auc: 0.7045
Epoch 81/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0106 -
accuracy: 0.6748 - auc: 0.7410 - val_loss: 1.0356 - val_accuracy: 0.6593 -
val_auc: 0.7140
Epoch 82/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0072 -
accuracy: 0.6772 - auc: 0.7432 - val_loss: 1.0287 - val_accuracy: 0.6593 -
val_auc: 0.7205
Epoch 83/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0063 -
accuracy: 0.6719 - auc: 0.7428 - val_loss: 1.0319 - val_accuracy: 0.6481 -
val_auc: 0.7116
Epoch 84/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0026 -
accuracy: 0.6795 - auc: 0.7431 - val_loss: 1.0267 - val_accuracy: 0.6568 -
val_auc: 0.7186
```

```
Epoch 85/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0027 -
accuracy: 0.6756 - auc: 0.7418 - val_loss: 1.0266 - val_accuracy: 0.6580 -
val_auc: 0.7161
Epoch 86/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9999 -
accuracy: 0.6753 - auc: 0.7431 - val_loss: 1.0235 - val_accuracy: 0.6568 -
val_auc: 0.7185
Epoch 87/1000
119/119 [=====] - 2s 19ms/step - loss: 1.0006 -
accuracy: 0.6745 - auc: 0.7393 - val_loss: 1.0206 - val_accuracy: 0.6543 -
val_auc: 0.7199
Epoch 88/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9946 -
accuracy: 0.6809 - auc: 0.7464 - val_loss: 1.0179 - val_accuracy: 0.6605 -
val_auc: 0.7231
Epoch 89/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9958 -
accuracy: 0.6788 - auc: 0.7418 - val_loss: 1.0182 - val_accuracy: 0.6531 -
val_auc: 0.7169
Epoch 90/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9896 -
accuracy: 0.6819 - auc: 0.7459 - val_loss: 1.0138 - val_accuracy: 0.6580 -
val_auc: 0.7213
Epoch 91/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9894 -
accuracy: 0.6806 - auc: 0.7451 - val_loss: 1.0124 - val_accuracy: 0.6531 -
val_auc: 0.7193
Epoch 92/1000
119/119 [=====] - 2s 20ms/step - loss: 0.9897 -
accuracy: 0.6806 - auc: 0.7413 - val_loss: 1.0145 - val_accuracy: 0.6593 -
val_auc: 0.7156
Epoch 93/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9850 -
accuracy: 0.6817 - auc: 0.7457 - val_loss: 1.0111 - val_accuracy: 0.6531 -
val_auc: 0.7150
Epoch 94/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9859 -
accuracy: 0.6801 - auc: 0.7423 - val_loss: 1.0123 - val_accuracy: 0.6481 -
val_auc: 0.7109
Epoch 95/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9825 -
accuracy: 0.6814 - auc: 0.7453 - val_loss: 1.0080 - val_accuracy: 0.6543 -
val_auc: 0.7158
Epoch 96/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9791 -
accuracy: 0.6819 - auc: 0.7455 - val_loss: 1.0046 - val_accuracy: 0.6556 -
val_auc: 0.7188
```

```
Epoch 97/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9760 -
accuracy: 0.6827 - auc: 0.7471 - val_loss: 1.0049 - val_accuracy: 0.6531 -
val_auc: 0.7150
Epoch 98/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9752 -
accuracy: 0.6793 - auc: 0.7482 - val_loss: 1.0038 - val_accuracy: 0.6519 -
val_auc: 0.7149
Epoch 99/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9736 -
accuracy: 0.6772 - auc: 0.7455 - val_loss: 1.0002 - val_accuracy: 0.6506 -
val_auc: 0.7172
Epoch 100/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9686 -
accuracy: 0.6795 - auc: 0.7503 - val_loss: 0.9968 - val_accuracy: 0.6556 -
val_auc: 0.7215
Epoch 101/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9699 -
accuracy: 0.6780 - auc: 0.7454 - val_loss: 0.9960 - val_accuracy: 0.6617 -
val_auc: 0.7217
Epoch 102/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9694 -
accuracy: 0.6769 - auc: 0.7448 - val_loss: 1.0021 - val_accuracy: 0.6346 -
val_auc: 0.7057
Epoch 103/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9673 -
accuracy: 0.6758 - auc: 0.7446 - val_loss: 0.9937 - val_accuracy: 0.6543 -
val_auc: 0.7183
Epoch 104/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9639 -
accuracy: 0.6811 - auc: 0.7480 - val_loss: 0.9905 - val_accuracy: 0.6568 -
val_auc: 0.7201
Epoch 105/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9618 -
accuracy: 0.6830 - auc: 0.7484 - val_loss: 0.9954 - val_accuracy: 0.6457 -
val_auc: 0.7090
Epoch 106/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9644 -
accuracy: 0.6777 - auc: 0.7418 - val_loss: 0.9847 - val_accuracy: 0.6556 -
val_auc: 0.7240
Epoch 107/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9599 -
accuracy: 0.6766 - auc: 0.7451 - val_loss: 0.9852 - val_accuracy: 0.6531 -
val_auc: 0.7230
Epoch 108/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9579 -
accuracy: 0.6817 - auc: 0.7478 - val_loss: 0.9859 - val_accuracy: 0.6506 -
val_auc: 0.7159
```

```
Epoch 109/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9583 -
accuracy: 0.6740 - auc: 0.7439 - val_loss: 0.9825 - val_accuracy: 0.6494 -
val_auc: 0.7175
Epoch 110/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9546 -
accuracy: 0.6819 - auc: 0.7481 - val_loss: 0.9773 - val_accuracy: 0.6556 -
val_auc: 0.7244
Epoch 111/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9527 -
accuracy: 0.6727 - auc: 0.7467 - val_loss: 0.9791 - val_accuracy: 0.6556 -
val_auc: 0.7196
Epoch 112/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9501 -
accuracy: 0.6825 - auc: 0.7495 - val_loss: 0.9774 - val_accuracy: 0.6568 -
val_auc: 0.7190
Epoch 113/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9497 -
accuracy: 0.6798 - auc: 0.7473 - val_loss: 0.9782 - val_accuracy: 0.6481 -
val_auc: 0.7158
Epoch 114/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9463 -
accuracy: 0.6795 - auc: 0.7484 - val_loss: 0.9747 - val_accuracy: 0.6531 -
val_auc: 0.7176
Epoch 115/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9454 -
accuracy: 0.6835 - auc: 0.7488 - val_loss: 0.9715 - val_accuracy: 0.6568 -
val_auc: 0.7207
Epoch 116/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9451 -
accuracy: 0.6766 - auc: 0.7472 - val_loss: 0.9692 - val_accuracy: 0.6593 -
val_auc: 0.7219
Epoch 117/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9425 -
accuracy: 0.6806 - auc: 0.7489 - val_loss: 0.9671 - val_accuracy: 0.6580 -
val_auc: 0.7228
Epoch 118/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9401 -
accuracy: 0.6753 - auc: 0.7479 - val_loss: 0.9664 - val_accuracy: 0.6506 -
val_auc: 0.7230
Epoch 119/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9372 -
accuracy: 0.6875 - auc: 0.7512 - val_loss: 0.9670 - val_accuracy: 0.6543 -
val_auc: 0.7204
Epoch 120/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9360 -
accuracy: 0.6832 - auc: 0.7499 - val_loss: 0.9662 - val_accuracy: 0.6506 -
val_auc: 0.7181
```

```
Epoch 121/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9355 -
accuracy: 0.6825 - auc: 0.7501 - val_loss: 0.9645 - val_accuracy: 0.6494 -
val_auc: 0.7173
Epoch 122/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9313 -
accuracy: 0.6838 - auc: 0.7535 - val_loss: 0.9598 - val_accuracy: 0.6519 -
val_auc: 0.7213
Epoch 123/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9334 -
accuracy: 0.6809 - auc: 0.7474 - val_loss: 0.9601 - val_accuracy: 0.6519 -
val_auc: 0.7195
Epoch 124/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9295 -
accuracy: 0.6838 - auc: 0.7512 - val_loss: 0.9584 - val_accuracy: 0.6494 -
val_auc: 0.7196
Epoch 125/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9298 -
accuracy: 0.6801 - auc: 0.7501 - val_loss: 0.9516 - val_accuracy: 0.6568 -
val_auc: 0.7278
Epoch 126/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9292 -
accuracy: 0.6788 - auc: 0.7456 - val_loss: 0.9513 - val_accuracy: 0.6543 -
val_auc: 0.7268
Epoch 127/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9254 -
accuracy: 0.6840 - auc: 0.7507 - val_loss: 0.9507 - val_accuracy: 0.6568 -
val_auc: 0.7266
Epoch 128/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9232 -
accuracy: 0.6867 - auc: 0.7523 - val_loss: 0.9526 - val_accuracy: 0.6543 -
val_auc: 0.7217
Epoch 129/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9217 -
accuracy: 0.6809 - auc: 0.7491 - val_loss: 0.9528 - val_accuracy: 0.6457 -
val_auc: 0.7163
Epoch 130/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9201 -
accuracy: 0.6838 - auc: 0.7514 - val_loss: 0.9456 - val_accuracy: 0.6556 -
val_auc: 0.7258
Epoch 131/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9183 -
accuracy: 0.6811 - auc: 0.7523 - val_loss: 0.9501 - val_accuracy: 0.6457 -
val_auc: 0.7167
Epoch 132/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9179 -
accuracy: 0.6814 - auc: 0.7498 - val_loss: 0.9442 - val_accuracy: 0.6531 -
val_auc: 0.7238
```

```
Epoch 133/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9139 -
accuracy: 0.6817 - auc: 0.7530 - val_loss: 0.9476 - val_accuracy: 0.6444 -
val_auc: 0.7156
Epoch 134/1000
119/119 [=====] - 2s 20ms/step - loss: 0.9136 -
accuracy: 0.6825 - auc: 0.7507 - val_loss: 0.9425 - val_accuracy: 0.6531 -
val_auc: 0.7217
Epoch 135/1000
119/119 [=====] - 3s 22ms/step - loss: 0.9112 -
accuracy: 0.6859 - auc: 0.7542 - val_loss: 0.9401 - val_accuracy: 0.6519 -
val_auc: 0.7223
Epoch 136/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9107 -
accuracy: 0.6790 - auc: 0.7507 - val_loss: 0.9387 - val_accuracy: 0.6556 -
val_auc: 0.7233
Epoch 137/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9102 -
accuracy: 0.6838 - auc: 0.7510 - val_loss: 0.9397 - val_accuracy: 0.6494 -
val_auc: 0.7206
Epoch 138/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9079 -
accuracy: 0.6848 - auc: 0.7514 - val_loss: 0.9313 - val_accuracy: 0.6630 -
val_auc: 0.7317
Epoch 139/1000
119/119 [=====] - 3s 21ms/step - loss: 0.9057 -
accuracy: 0.6830 - auc: 0.7532 - val_loss: 0.9315 - val_accuracy: 0.6543 -
val_auc: 0.7268
Epoch 140/1000
119/119 [=====] - 2s 21ms/step - loss: 0.9026 -
accuracy: 0.6835 - auc: 0.7539 - val_loss: 0.9316 - val_accuracy: 0.6531 -
val_auc: 0.7233
Epoch 141/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9035 -
accuracy: 0.6830 - auc: 0.7522 - val_loss: 0.9323 - val_accuracy: 0.6531 -
val_auc: 0.7207
Epoch 142/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9006 -
accuracy: 0.6840 - auc: 0.7527 - val_loss: 0.9320 - val_accuracy: 0.6494 -
val_auc: 0.7189
Epoch 143/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9011 -
accuracy: 0.6795 - auc: 0.7511 - val_loss: 0.9366 - val_accuracy: 0.6383 -
val_auc: 0.7094
Epoch 144/1000
119/119 [=====] - 2s 19ms/step - loss: 0.9010 -
accuracy: 0.6795 - auc: 0.7496 - val_loss: 0.9272 - val_accuracy: 0.6469 -
val_auc: 0.7211
```

```
Epoch 145/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8964 -
accuracy: 0.6896 - auc: 0.7544 - val_loss: 0.9223 - val_accuracy: 0.6531 -
val_auc: 0.7284
Epoch 146/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8936 -
accuracy: 0.6875 - auc: 0.7547 - val_loss: 0.9250 - val_accuracy: 0.6481 -
val_auc: 0.7216
Epoch 147/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8925 -
accuracy: 0.6883 - auc: 0.7563 - val_loss: 0.9286 - val_accuracy: 0.6432 -
val_auc: 0.7135
Epoch 148/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8939 -
accuracy: 0.6854 - auc: 0.7513 - val_loss: 0.9201 - val_accuracy: 0.6556 -
val_auc: 0.7266
Epoch 149/1000
119/119 [=====] - 2s 20ms/step - loss: 0.8904 -
accuracy: 0.6870 - auc: 0.7554 - val_loss: 0.9253 - val_accuracy: 0.6444 -
val_auc: 0.7179
Epoch 150/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8894 -
accuracy: 0.6880 - auc: 0.7560 - val_loss: 0.9190 - val_accuracy: 0.6506 -
val_auc: 0.7234
Epoch 151/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8889 -
accuracy: 0.6862 - auc: 0.7522 - val_loss: 0.9217 - val_accuracy: 0.6481 -
val_auc: 0.7173
Epoch 152/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8875 -
accuracy: 0.6830 - auc: 0.7524 - val_loss: 0.9172 - val_accuracy: 0.6519 -
val_auc: 0.7215
Epoch 153/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8857 -
accuracy: 0.6811 - auc: 0.7557 - val_loss: 0.9104 - val_accuracy: 0.6531 -
val_auc: 0.7297
Epoch 154/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8839 -
accuracy: 0.6832 - auc: 0.7542 - val_loss: 0.9153 - val_accuracy: 0.6519 -
val_auc: 0.7208
Epoch 155/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8815 -
accuracy: 0.6848 - auc: 0.7568 - val_loss: 0.9189 - val_accuracy: 0.6407 -
val_auc: 0.7142
Epoch 156/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8809 -
accuracy: 0.6896 - auc: 0.7554 - val_loss: 0.9085 - val_accuracy: 0.6506 -
val_auc: 0.7277
```

```
Epoch 157/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8774 -
accuracy: 0.6920 - auc: 0.7576 - val_loss: 0.9067 - val_accuracy: 0.6506 -
val_auc: 0.7277
Epoch 158/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8786 -
accuracy: 0.6864 - auc: 0.7533 - val_loss: 0.9054 - val_accuracy: 0.6568 -
val_auc: 0.7267
Epoch 159/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8763 -
accuracy: 0.6864 - auc: 0.7557 - val_loss: 0.9116 - val_accuracy: 0.6481 -
val_auc: 0.7169
Epoch 160/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8777 -
accuracy: 0.6846 - auc: 0.7526 - val_loss: 0.9096 - val_accuracy: 0.6432 -
val_auc: 0.7155
Epoch 161/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8760 -
accuracy: 0.6867 - auc: 0.7528 - val_loss: 0.8993 - val_accuracy: 0.6593 -
val_auc: 0.7303
Epoch 162/1000
119/119 [=====] - 2s 20ms/step - loss: 0.8716 -
accuracy: 0.6893 - auc: 0.7561 - val_loss: 0.8998 - val_accuracy: 0.6494 -
val_auc: 0.7283
Epoch 163/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8701 -
accuracy: 0.6883 - auc: 0.7571 - val_loss: 0.9006 - val_accuracy: 0.6494 -
val_auc: 0.7257
Epoch 164/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8686 -
accuracy: 0.6875 - auc: 0.7569 - val_loss: 0.9008 - val_accuracy: 0.6556 -
val_auc: 0.7228
Epoch 165/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8685 -
accuracy: 0.6840 - auc: 0.7554 - val_loss: 0.9000 - val_accuracy: 0.6494 -
val_auc: 0.7198
Epoch 166/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8670 -
accuracy: 0.6875 - auc: 0.7565 - val_loss: 0.8987 - val_accuracy: 0.6494 -
val_auc: 0.7232
Epoch 167/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8662 -
accuracy: 0.6832 - auc: 0.7566 - val_loss: 0.8935 - val_accuracy: 0.6556 -
val_auc: 0.7287
Epoch 168/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8643 -
accuracy: 0.6867 - auc: 0.7559 - val_loss: 0.8934 - val_accuracy: 0.6531 -
val_auc: 0.7272
```

```
Epoch 169/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8633 -
accuracy: 0.6899 - auc: 0.7563 - val_loss: 0.8937 - val_accuracy: 0.6481 -
val_auc: 0.7235
Epoch 170/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8613 -
accuracy: 0.6854 - auc: 0.7586 - val_loss: 0.8909 - val_accuracy: 0.6531 -
val_auc: 0.7277
Epoch 171/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8606 -
accuracy: 0.6856 - auc: 0.7556 - val_loss: 0.8905 - val_accuracy: 0.6568 -
val_auc: 0.7263
Epoch 172/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8578 -
accuracy: 0.6848 - auc: 0.7577 - val_loss: 0.8890 - val_accuracy: 0.6519 -
val_auc: 0.7250
Epoch 173/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8573 -
accuracy: 0.6870 - auc: 0.7579 - val_loss: 0.8838 - val_accuracy: 0.6556 -
val_auc: 0.7309
Epoch 174/1000
119/119 [=====] - 2s 21ms/step - loss: 0.8558 -
accuracy: 0.6875 - auc: 0.7581 - val_loss: 0.8840 - val_accuracy: 0.6531 -
val_auc: 0.7288
Epoch 175/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8575 -
accuracy: 0.6899 - auc: 0.7555 - val_loss: 0.8840 - val_accuracy: 0.6568 -
val_auc: 0.7288
Epoch 176/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8570 -
accuracy: 0.6854 - auc: 0.7545 - val_loss: 0.8878 - val_accuracy: 0.6469 -
val_auc: 0.7204
Epoch 177/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8534 -
accuracy: 0.6830 - auc: 0.7570 - val_loss: 0.8834 - val_accuracy: 0.6531 -
val_auc: 0.7259
Epoch 178/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8521 -
accuracy: 0.6936 - auc: 0.7586 - val_loss: 0.8844 - val_accuracy: 0.6519 -
val_auc: 0.7236
Epoch 179/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8494 -
accuracy: 0.6875 - auc: 0.7598 - val_loss: 0.8794 - val_accuracy: 0.6519 -
val_auc: 0.7265
Epoch 180/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8475 -
accuracy: 0.6864 - auc: 0.7590 - val_loss: 0.8779 - val_accuracy: 0.6556 -
val_auc: 0.7280
```

```
Epoch 181/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8514 -
accuracy: 0.6848 - auc: 0.7539 - val_loss: 0.8777 - val_accuracy: 0.6568 -
val_auc: 0.7258
Epoch 182/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8458 -
accuracy: 0.6915 - auc: 0.7595 - val_loss: 0.8790 - val_accuracy: 0.6420 -
val_auc: 0.7233
Epoch 183/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8459 -
accuracy: 0.6851 - auc: 0.7571 - val_loss: 0.8749 - val_accuracy: 0.6469 -
val_auc: 0.7261
Epoch 184/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8423 -
accuracy: 0.6891 - auc: 0.7608 - val_loss: 0.8735 - val_accuracy: 0.6506 -
val_auc: 0.7271
Epoch 185/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8448 -
accuracy: 0.6856 - auc: 0.7557 - val_loss: 0.8723 - val_accuracy: 0.6506 -
val_auc: 0.7270
Epoch 186/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8396 -
accuracy: 0.6944 - auc: 0.7613 - val_loss: 0.8749 - val_accuracy: 0.6469 -
val_auc: 0.7208
Epoch 187/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8423 -
accuracy: 0.6846 - auc: 0.7565 - val_loss: 0.8750 - val_accuracy: 0.6494 -
val_auc: 0.7188
Epoch 188/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8395 -
accuracy: 0.6875 - auc: 0.7581 - val_loss: 0.8763 - val_accuracy: 0.6420 -
val_auc: 0.7155
Epoch 189/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8382 -
accuracy: 0.6859 - auc: 0.7597 - val_loss: 0.8710 - val_accuracy: 0.6457 -
val_auc: 0.7218
Epoch 190/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8367 -
accuracy: 0.6880 - auc: 0.7607 - val_loss: 0.8685 - val_accuracy: 0.6543 -
val_auc: 0.7243
Epoch 191/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8368 -
accuracy: 0.6867 - auc: 0.7575 - val_loss: 0.8736 - val_accuracy: 0.6395 -
val_auc: 0.7161
Epoch 192/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8326 -
accuracy: 0.6938 - auc: 0.7634 - val_loss: 0.8689 - val_accuracy: 0.6432 -
val_auc: 0.7220
```

```
Epoch 193/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8322 -
accuracy: 0.6899 - auc: 0.7614 - val_loss: 0.8628 - val_accuracy: 0.6494 -
val_auc: 0.7278
Epoch 194/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8310 -
accuracy: 0.6880 - auc: 0.7599 - val_loss: 0.8688 - val_accuracy: 0.6395 -
val_auc: 0.7170
Epoch 195/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8319 -
accuracy: 0.6901 - auc: 0.7594 - val_loss: 0.8595 - val_accuracy: 0.6556 -
val_auc: 0.7306
Epoch 196/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8300 -
accuracy: 0.6941 - auc: 0.7613 - val_loss: 0.8592 - val_accuracy: 0.6481 -
val_auc: 0.7272
Epoch 197/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8297 -
accuracy: 0.6851 - auc: 0.7581 - val_loss: 0.8567 - val_accuracy: 0.6580 -
val_auc: 0.7316
Epoch 198/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8253 -
accuracy: 0.6907 - auc: 0.7634 - val_loss: 0.8567 - val_accuracy: 0.6605 -
val_auc: 0.7298
Epoch 199/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8248 -
accuracy: 0.6896 - auc: 0.7616 - val_loss: 0.8559 - val_accuracy: 0.6568 -
val_auc: 0.7310
Epoch 200/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8244 -
accuracy: 0.6904 - auc: 0.7596 - val_loss: 0.8593 - val_accuracy: 0.6494 -
val_auc: 0.7258
Epoch 201/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8252 -
accuracy: 0.6856 - auc: 0.7595 - val_loss: 0.8546 - val_accuracy: 0.6494 -
val_auc: 0.7274
Epoch 202/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8231 -
accuracy: 0.6885 - auc: 0.7622 - val_loss: 0.8506 - val_accuracy: 0.6593 -
val_auc: 0.7319
Epoch 203/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8231 -
accuracy: 0.6870 - auc: 0.7585 - val_loss: 0.8550 - val_accuracy: 0.6531 -
val_auc: 0.7258
Epoch 204/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8203 -
accuracy: 0.6888 - auc: 0.7620 - val_loss: 0.8555 - val_accuracy: 0.6481 -
val_auc: 0.7231
```

```
Epoch 205/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8199 -
accuracy: 0.6907 - auc: 0.7604 - val_loss: 0.8505 - val_accuracy: 0.6519 -
val_auc: 0.7283
Epoch 206/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8158 -
accuracy: 0.6928 - auc: 0.7639 - val_loss: 0.8507 - val_accuracy: 0.6519 -
val_auc: 0.7265
Epoch 207/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8154 -
accuracy: 0.6941 - auc: 0.7638 - val_loss: 0.8532 - val_accuracy: 0.6395 -
val_auc: 0.7204
Epoch 208/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8154 -
accuracy: 0.6920 - auc: 0.7621 - val_loss: 0.8553 - val_accuracy: 0.6395 -
val_auc: 0.7173
Epoch 209/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8135 -
accuracy: 0.6912 - auc: 0.7641 - val_loss: 0.8497 - val_accuracy: 0.6432 -
val_auc: 0.7233
Epoch 210/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8130 -
accuracy: 0.6907 - auc: 0.7623 - val_loss: 0.8452 - val_accuracy: 0.6568 -
val_auc: 0.7278
Epoch 211/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8131 -
accuracy: 0.6880 - auc: 0.7608 - val_loss: 0.8539 - val_accuracy: 0.6358 -
val_auc: 0.7129
Epoch 212/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8116 -
accuracy: 0.6877 - auc: 0.7615 - val_loss: 0.8474 - val_accuracy: 0.6481 -
val_auc: 0.7233
Epoch 213/1000
119/119 [=====] - 2s 20ms/step - loss: 0.8100 -
accuracy: 0.6872 - auc: 0.7615 - val_loss: 0.8419 - val_accuracy: 0.6519 -
val_auc: 0.7292
Epoch 214/1000
119/119 [=====] - 2s 20ms/step - loss: 0.8120 -
accuracy: 0.6909 - auc: 0.7581 - val_loss: 0.8417 - val_accuracy: 0.6531 -
val_auc: 0.7284
Epoch 215/1000
119/119 [=====] - 2s 20ms/step - loss: 0.8066 -
accuracy: 0.6917 - auc: 0.7645 - val_loss: 0.8450 - val_accuracy: 0.6469 -
val_auc: 0.7226
Epoch 216/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8048 -
accuracy: 0.6941 - auc: 0.7651 - val_loss: 0.8360 - val_accuracy: 0.6543 -
val_auc: 0.7328
```

```
Epoch 217/1000
119/119 [=====] - 2s 20ms/step - loss: 0.8067 -
accuracy: 0.6946 - auc: 0.7621 - val_loss: 0.8387 - val_accuracy: 0.6543 -
val_auc: 0.7284
Epoch 218/1000
119/119 [=====] - 2s 20ms/step - loss: 0.8057 -
accuracy: 0.6904 - auc: 0.7611 - val_loss: 0.8372 - val_accuracy: 0.6519 -
val_auc: 0.7285
Epoch 219/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8022 -
accuracy: 0.6944 - auc: 0.7655 - val_loss: 0.8371 - val_accuracy: 0.6494 -
val_auc: 0.7269
Epoch 220/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8024 -
accuracy: 0.6907 - auc: 0.7635 - val_loss: 0.8379 - val_accuracy: 0.6556 -
val_auc: 0.7267
Epoch 221/1000
119/119 [=====] - 2s 19ms/step - loss: 0.8021 -
accuracy: 0.6933 - auc: 0.7648 - val_loss: 0.8362 - val_accuracy: 0.6469 -
val_auc: 0.7247
Epoch 222/1000
119/119 [=====] - 2s 20ms/step - loss: 0.8026 -
accuracy: 0.6915 - auc: 0.7621 - val_loss: 0.8328 - val_accuracy: 0.6506 -
val_auc: 0.7280
Epoch 223/1000
119/119 [=====] - 2s 20ms/step - loss: 0.8003 -
accuracy: 0.6960 - auc: 0.7644 - val_loss: 0.8310 - val_accuracy: 0.6556 -
val_auc: 0.7307
Epoch 224/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7988 -
accuracy: 0.6930 - auc: 0.7640 - val_loss: 0.8296 - val_accuracy: 0.6568 -
val_auc: 0.7317
Epoch 225/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7985 -
accuracy: 0.6954 - auc: 0.7628 - val_loss: 0.8292 - val_accuracy: 0.6556 -
val_auc: 0.7308
Epoch 226/1000
119/119 [=====] - 3s 22ms/step - loss: 0.7982 -
accuracy: 0.6907 - auc: 0.7627 - val_loss: 0.8315 - val_accuracy: 0.6444 -
val_auc: 0.7253
Epoch 227/1000
119/119 [=====] - 2s 21ms/step - loss: 0.7961 -
accuracy: 0.6891 - auc: 0.7625 - val_loss: 0.8358 - val_accuracy: 0.6420 -
val_auc: 0.7183
Epoch 228/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7942 -
accuracy: 0.6981 - auc: 0.7652 - val_loss: 0.8265 - val_accuracy: 0.6630 -
val_auc: 0.7304
```

```
Epoch 229/1000
119/119 [=====] - 2s 21ms/step - loss: 0.7919 -
accuracy: 0.6941 - auc: 0.7657 - val_loss: 0.8310 - val_accuracy: 0.6531 -
val_auc: 0.7239
Epoch 230/1000
119/119 [=====] - 3s 21ms/step - loss: 0.7933 -
accuracy: 0.6925 - auc: 0.7631 - val_loss: 0.8309 - val_accuracy: 0.6420 -
val_auc: 0.7212
Epoch 231/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7911 -
accuracy: 0.6901 - auc: 0.7639 - val_loss: 0.8234 - val_accuracy: 0.6568 -
val_auc: 0.7309
Epoch 232/1000
119/119 [=====] - 2s 19ms/step - loss: 0.7912 -
accuracy: 0.6917 - auc: 0.7642 - val_loss: 0.8290 - val_accuracy: 0.6531 -
val_auc: 0.7220
Epoch 233/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7888 -
accuracy: 0.6915 - auc: 0.7648 - val_loss: 0.8266 - val_accuracy: 0.6519 -
val_auc: 0.7246
Epoch 234/1000
119/119 [=====] - 2s 19ms/step - loss: 0.7878 -
accuracy: 0.6967 - auc: 0.7666 - val_loss: 0.8209 - val_accuracy: 0.6630 -
val_auc: 0.7315
Epoch 235/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7892 -
accuracy: 0.6967 - auc: 0.7648 - val_loss: 0.8227 - val_accuracy: 0.6494 -
val_auc: 0.7266
Epoch 236/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7864 -
accuracy: 0.6965 - auc: 0.7654 - val_loss: 0.8225 - val_accuracy: 0.6556 -
val_auc: 0.7260
Epoch 237/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7869 -
accuracy: 0.6954 - auc: 0.7642 - val_loss: 0.8209 - val_accuracy: 0.6469 -
val_auc: 0.7251
Epoch 238/1000
119/119 [=====] - 2s 19ms/step - loss: 0.7836 -
accuracy: 0.6944 - auc: 0.7672 - val_loss: 0.8283 - val_accuracy: 0.6395 -
val_auc: 0.7153
Epoch 239/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7855 -
accuracy: 0.6893 - auc: 0.7629 - val_loss: 0.8185 - val_accuracy: 0.6531 -
val_auc: 0.7273
Epoch 240/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7839 -
accuracy: 0.6896 - auc: 0.7634 - val_loss: 0.8199 - val_accuracy: 0.6494 -
val_auc: 0.7241
```

```
Epoch 241/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7814 -
accuracy: 0.6946 - auc: 0.7664 - val_loss: 0.8195 - val_accuracy: 0.6531 -
val_auc: 0.7244
Epoch 242/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7798 -
accuracy: 0.6960 - auc: 0.7674 - val_loss: 0.8164 - val_accuracy: 0.6580 -
val_auc: 0.7280
Epoch 243/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7790 -
accuracy: 0.6978 - auc: 0.7672 - val_loss: 0.8137 - val_accuracy: 0.6580 -
val_auc: 0.7309
Epoch 244/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7801 -
accuracy: 0.6907 - auc: 0.7642 - val_loss: 0.8150 - val_accuracy: 0.6580 -
val_auc: 0.7270
Epoch 245/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7801 -
accuracy: 0.6880 - auc: 0.7641 - val_loss: 0.8172 - val_accuracy: 0.6519 -
val_auc: 0.7220
Epoch 246/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7759 -
accuracy: 0.6941 - auc: 0.7675 - val_loss: 0.8121 - val_accuracy: 0.6593 -
val_auc: 0.7278
Epoch 247/1000
119/119 [=====] - 2s 19ms/step - loss: 0.7751 -
accuracy: 0.6949 - auc: 0.7667 - val_loss: 0.8128 - val_accuracy: 0.6494 -
val_auc: 0.7265
Epoch 248/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7767 -
accuracy: 0.6904 - auc: 0.7644 - val_loss: 0.8158 - val_accuracy: 0.6444 -
val_auc: 0.7206
Epoch 249/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7733 -
accuracy: 0.6954 - auc: 0.7677 - val_loss: 0.8159 - val_accuracy: 0.6506 -
val_auc: 0.7206
Epoch 250/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7740 -
accuracy: 0.6944 - auc: 0.7659 - val_loss: 0.8128 - val_accuracy: 0.6506 -
val_auc: 0.7221
Epoch 251/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7729 -
accuracy: 0.6930 - auc: 0.7676 - val_loss: 0.8125 - val_accuracy: 0.6494 -
val_auc: 0.7211
Epoch 252/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7725 -
accuracy: 0.6896 - auc: 0.7647 - val_loss: 0.8122 - val_accuracy: 0.6494 -
val_auc: 0.7232
```

```
Epoch 253/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7711 -
accuracy: 0.6891 - auc: 0.7649 - val_loss: 0.8034 - val_accuracy: 0.6679 -
val_auc: 0.7353
Epoch 254/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7696 -
accuracy: 0.7012 - auc: 0.7688 - val_loss: 0.8056 - val_accuracy: 0.6531 -
val_auc: 0.7283
Epoch 255/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7691 -
accuracy: 0.6946 - auc: 0.7673 - val_loss: 0.8034 - val_accuracy: 0.6617 -
val_auc: 0.7328
Epoch 256/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7696 -
accuracy: 0.6917 - auc: 0.7641 - val_loss: 0.8062 - val_accuracy: 0.6531 -
val_auc: 0.7244
Epoch 257/1000
119/119 [=====] - 2s 19ms/step - loss: 0.7690 -
accuracy: 0.6896 - auc: 0.7642 - val_loss: 0.8083 - val_accuracy: 0.6457 -
val_auc: 0.7214
Epoch 258/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7664 -
accuracy: 0.6978 - auc: 0.7672 - val_loss: 0.8022 - val_accuracy: 0.6556 -
val_auc: 0.7294
Epoch 259/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7640 -
accuracy: 0.6946 - auc: 0.7690 - val_loss: 0.8022 - val_accuracy: 0.6494 -
val_auc: 0.7266
Epoch 260/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7658 -
accuracy: 0.6912 - auc: 0.7642 - val_loss: 0.7984 - val_accuracy: 0.6642 -
val_auc: 0.7321
Epoch 261/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7655 -
accuracy: 0.6899 - auc: 0.7653 - val_loss: 0.8016 - val_accuracy: 0.6519 -
val_auc: 0.7257
Epoch 262/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7612 -
accuracy: 0.6970 - auc: 0.7696 - val_loss: 0.7993 - val_accuracy: 0.6593 -
val_auc: 0.7295
Epoch 263/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7635 -
accuracy: 0.6915 - auc: 0.7651 - val_loss: 0.8061 - val_accuracy: 0.6432 -
val_auc: 0.7161
Epoch 264/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7623 -
accuracy: 0.6909 - auc: 0.7662 - val_loss: 0.7990 - val_accuracy: 0.6543 -
val_auc: 0.7275
```

```
Epoch 265/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7616 -
accuracy: 0.6933 - auc: 0.7657 - val_loss: 0.7931 - val_accuracy: 0.6691 -
val_auc: 0.7348
Epoch 266/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7567 -
accuracy: 0.6928 - auc: 0.7703 - val_loss: 0.7970 - val_accuracy: 0.6543 -
val_auc: 0.7277
Epoch 267/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7601 -
accuracy: 0.6896 - auc: 0.7640 - val_loss: 0.7991 - val_accuracy: 0.6556 -
val_auc: 0.7258
Epoch 268/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7587 -
accuracy: 0.6899 - auc: 0.7663 - val_loss: 0.7961 - val_accuracy: 0.6556 -
val_auc: 0.7281
Epoch 269/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7557 -
accuracy: 0.6946 - auc: 0.7697 - val_loss: 0.7929 - val_accuracy: 0.6556 -
val_auc: 0.7300
Epoch 270/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7557 -
accuracy: 0.6975 - auc: 0.7697 - val_loss: 0.7978 - val_accuracy: 0.6469 -
val_auc: 0.7225
Epoch 271/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7572 -
accuracy: 0.6938 - auc: 0.7671 - val_loss: 0.7934 - val_accuracy: 0.6593 -
val_auc: 0.7277
Epoch 272/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7548 -
accuracy: 0.6965 - auc: 0.7682 - val_loss: 0.7942 - val_accuracy: 0.6494 -
val_auc: 0.7245
Epoch 273/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7531 -
accuracy: 0.6928 - auc: 0.7694 - val_loss: 0.7900 - val_accuracy: 0.6556 -
val_auc: 0.7309
Epoch 274/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7521 -
accuracy: 0.6904 - auc: 0.7693 - val_loss: 0.7990 - val_accuracy: 0.6420 -
val_auc: 0.7161
Epoch 275/1000
119/119 [=====] - 2s 19ms/step - loss: 0.7537 -
accuracy: 0.6915 - auc: 0.7662 - val_loss: 0.7867 - val_accuracy: 0.6605 -
val_auc: 0.7330
Epoch 276/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7527 -
accuracy: 0.6957 - auc: 0.7677 - val_loss: 0.7852 - val_accuracy: 0.6716 -
val_auc: 0.7345
```

```
Epoch 277/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7508 -
accuracy: 0.6944 - auc: 0.7683 - val_loss: 0.7838 - val_accuracy: 0.6716 -
val_auc: 0.7345
Epoch 278/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7491 -
accuracy: 0.7004 - auc: 0.7703 - val_loss: 0.7826 - val_accuracy: 0.6679 -
val_auc: 0.7353
Epoch 279/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7501 -
accuracy: 0.6949 - auc: 0.7679 - val_loss: 0.7825 - val_accuracy: 0.6691 -
val_auc: 0.7353
Epoch 280/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7477 -
accuracy: 0.6930 - auc: 0.7684 - val_loss: 0.7890 - val_accuracy: 0.6543 -
val_auc: 0.7242
Epoch 281/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7473 -
accuracy: 0.6973 - auc: 0.7699 - val_loss: 0.7819 - val_accuracy: 0.6617 -
val_auc: 0.7319
Epoch 282/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7477 -
accuracy: 0.6954 - auc: 0.7673 - val_loss: 0.7905 - val_accuracy: 0.6469 -
val_auc: 0.7209
Epoch 283/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7466 -
accuracy: 0.6962 - auc: 0.7670 - val_loss: 0.7880 - val_accuracy: 0.6481 -
val_auc: 0.7244
Epoch 284/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7454 -
accuracy: 0.6970 - auc: 0.7691 - val_loss: 0.7806 - val_accuracy: 0.6679 -
val_auc: 0.7335
Epoch 285/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7451 -
accuracy: 0.6991 - auc: 0.7682 - val_loss: 0.7789 - val_accuracy: 0.6605 -
val_auc: 0.7338
Epoch 286/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7432 -
accuracy: 0.6957 - auc: 0.7703 - val_loss: 0.7828 - val_accuracy: 0.6494 -
val_auc: 0.7248
Epoch 287/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7454 -
accuracy: 0.6970 - auc: 0.7665 - val_loss: 0.7810 - val_accuracy: 0.6580 -
val_auc: 0.7285
Epoch 288/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7430 -
accuracy: 0.6928 - auc: 0.7692 - val_loss: 0.7796 - val_accuracy: 0.6580 -
val_auc: 0.7297
```

```
Epoch 289/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7425 -
accuracy: 0.6930 - auc: 0.7678 - val_loss: 0.7867 - val_accuracy: 0.6506 -
val_auc: 0.7197
Epoch 290/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7419 -
accuracy: 0.6912 - auc: 0.7664 - val_loss: 0.7795 - val_accuracy: 0.6543 -
val_auc: 0.7276
Epoch 291/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7431 -
accuracy: 0.6970 - auc: 0.7673 - val_loss: 0.7775 - val_accuracy: 0.6617 -
val_auc: 0.7304
Epoch 292/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7386 -
accuracy: 0.6925 - auc: 0.7699 - val_loss: 0.7764 - val_accuracy: 0.6679 -
val_auc: 0.7316
Epoch 293/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7379 -
accuracy: 0.6978 - auc: 0.7715 - val_loss: 0.7776 - val_accuracy: 0.6642 -
val_auc: 0.7289
Epoch 294/1000
119/119 [=====] - 2s 19ms/step - loss: 0.7349 -
accuracy: 0.6962 - auc: 0.7739 - val_loss: 0.7766 - val_accuracy: 0.6605 -
val_auc: 0.7289
Epoch 295/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7360 -
accuracy: 0.6949 - auc: 0.7707 - val_loss: 0.7715 - val_accuracy: 0.6679 -
val_auc: 0.7354
Epoch 296/1000
119/119 [=====] - 2s 19ms/step - loss: 0.7339 -
accuracy: 0.6999 - auc: 0.7723 - val_loss: 0.7770 - val_accuracy: 0.6519 -
val_auc: 0.7252
Epoch 297/1000
119/119 [=====] - 2s 21ms/step - loss: 0.7349 -
accuracy: 0.6991 - auc: 0.7703 - val_loss: 0.7724 - val_accuracy: 0.6667 -
val_auc: 0.7322
Epoch 298/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7349 -
accuracy: 0.6946 - auc: 0.7712 - val_loss: 0.7700 - val_accuracy: 0.6642 -
val_auc: 0.7339
Epoch 299/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7335 -
accuracy: 0.6957 - auc: 0.7708 - val_loss: 0.7722 - val_accuracy: 0.6580 -
val_auc: 0.7302
Epoch 300/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7330 -
accuracy: 0.6986 - auc: 0.7711 - val_loss: 0.7741 - val_accuracy: 0.6593 -
val_auc: 0.7277
```

```
Epoch 301/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7324 -
accuracy: 0.6978 - auc: 0.7705 - val_loss: 0.7747 - val_accuracy: 0.6494 -
val_auc: 0.7236
Epoch 302/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7283 -
accuracy: 0.6981 - auc: 0.7739 - val_loss: 0.7663 - val_accuracy: 0.6642 -
val_auc: 0.7352
Epoch 303/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7314 -
accuracy: 0.6967 - auc: 0.7723 - val_loss: 0.7673 - val_accuracy: 0.6630 -
val_auc: 0.7333
Epoch 304/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7309 -
accuracy: 0.6925 - auc: 0.7711 - val_loss: 0.7635 - val_accuracy: 0.6728 -
val_auc: 0.7370
Epoch 305/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7286 -
accuracy: 0.7012 - auc: 0.7719 - val_loss: 0.7681 - val_accuracy: 0.6667 -
val_auc: 0.7311
Epoch 306/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7270 -
accuracy: 0.6922 - auc: 0.7722 - val_loss: 0.7671 - val_accuracy: 0.6691 -
val_auc: 0.7313
Epoch 307/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7281 -
accuracy: 0.6983 - auc: 0.7709 - val_loss: 0.7658 - val_accuracy: 0.6667 -
val_auc: 0.7325
Epoch 308/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7287 -
accuracy: 0.6978 - auc: 0.7698 - val_loss: 0.7634 - val_accuracy: 0.6654 -
val_auc: 0.7337
Epoch 309/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7266 -
accuracy: 0.6952 - auc: 0.7712 - val_loss: 0.7692 - val_accuracy: 0.6630 -
val_auc: 0.7271
Epoch 310/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7249 -
accuracy: 0.7023 - auc: 0.7723 - val_loss: 0.7661 - val_accuracy: 0.6667 -
val_auc: 0.7310
Epoch 311/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7258 -
accuracy: 0.6986 - auc: 0.7714 - val_loss: 0.7700 - val_accuracy: 0.6556 -
val_auc: 0.7252
Epoch 312/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7234 -
accuracy: 0.7007 - auc: 0.7739 - val_loss: 0.7628 - val_accuracy: 0.6580 -
val_auc: 0.7305
```

```
Epoch 313/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7234 -
accuracy: 0.6975 - auc: 0.7723 - val_loss: 0.7611 - val_accuracy: 0.6654 -
val_auc: 0.7325
Epoch 314/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7233 -
accuracy: 0.6967 - auc: 0.7724 - val_loss: 0.7571 - val_accuracy: 0.6704 -
val_auc: 0.7366
Epoch 315/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7187 -
accuracy: 0.7023 - auc: 0.7771 - val_loss: 0.7638 - val_accuracy: 0.6605 -
val_auc: 0.7286
Epoch 316/1000
119/119 [=====] - 3s 22ms/step - loss: 0.7213 -
accuracy: 0.7015 - auc: 0.7738 - val_loss: 0.7593 - val_accuracy: 0.6593 -
val_auc: 0.7314
Epoch 317/1000
119/119 [=====] - 2s 21ms/step - loss: 0.7217 -
accuracy: 0.7004 - auc: 0.7708 - val_loss: 0.7575 - val_accuracy: 0.6679 -
val_auc: 0.7352
Epoch 318/1000
119/119 [=====] - 2s 21ms/step - loss: 0.7220 -
accuracy: 0.6973 - auc: 0.7704 - val_loss: 0.7600 - val_accuracy: 0.6506 -
val_auc: 0.7285
Epoch 319/1000
119/119 [=====] - 3s 22ms/step - loss: 0.7181 -
accuracy: 0.7052 - auc: 0.7745 - val_loss: 0.7664 - val_accuracy: 0.6531 -
val_auc: 0.7214
Epoch 320/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7174 -
accuracy: 0.6989 - auc: 0.7749 - val_loss: 0.7566 - val_accuracy: 0.6667 -
val_auc: 0.7329
Epoch 321/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7167 -
accuracy: 0.6991 - auc: 0.7743 - val_loss: 0.7615 - val_accuracy: 0.6506 -
val_auc: 0.7266
Epoch 322/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7183 -
accuracy: 0.6981 - auc: 0.7733 - val_loss: 0.7517 - val_accuracy: 0.6704 -
val_auc: 0.7387
Epoch 323/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7155 -
accuracy: 0.7023 - auc: 0.7753 - val_loss: 0.7589 - val_accuracy: 0.6654 -
val_auc: 0.7297
Epoch 324/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7157 -
accuracy: 0.6991 - auc: 0.7744 - val_loss: 0.7555 - val_accuracy: 0.6630 -
val_auc: 0.7315
```

```
Epoch 325/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7160 -
accuracy: 0.7018 - auc: 0.7735 - val_loss: 0.7583 - val_accuracy: 0.6556 -
val_auc: 0.7270
Epoch 326/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7157 -
accuracy: 0.7007 - auc: 0.7720 - val_loss: 0.7516 - val_accuracy: 0.6716 -
val_auc: 0.7350
Epoch 327/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7147 -
accuracy: 0.7063 - auc: 0.7740 - val_loss: 0.7540 - val_accuracy: 0.6654 -
val_auc: 0.7331
Epoch 328/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7148 -
accuracy: 0.7063 - auc: 0.7728 - val_loss: 0.7544 - val_accuracy: 0.6556 -
val_auc: 0.7304
Epoch 329/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7121 -
accuracy: 0.7012 - auc: 0.7765 - val_loss: 0.7533 - val_accuracy: 0.6605 -
val_auc: 0.7312
Epoch 330/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7150 -
accuracy: 0.6973 - auc: 0.7706 - val_loss: 0.7515 - val_accuracy: 0.6642 -
val_auc: 0.7333
Epoch 331/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7117 -
accuracy: 0.7060 - auc: 0.7755 - val_loss: 0.7519 - val_accuracy: 0.6667 -
val_auc: 0.7322
Epoch 332/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7122 -
accuracy: 0.6970 - auc: 0.7729 - val_loss: 0.7462 - val_accuracy: 0.6753 -
val_auc: 0.7396
Epoch 333/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7104 -
accuracy: 0.6999 - auc: 0.7747 - val_loss: 0.7461 - val_accuracy: 0.6778 -
val_auc: 0.7387
Epoch 334/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7105 -
accuracy: 0.6970 - auc: 0.7738 - val_loss: 0.7497 - val_accuracy: 0.6630 -
val_auc: 0.7310
Epoch 335/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7085 -
accuracy: 0.7044 - auc: 0.7756 - val_loss: 0.7510 - val_accuracy: 0.6605 -
val_auc: 0.7299
Epoch 336/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7079 -
accuracy: 0.7023 - auc: 0.7738 - val_loss: 0.7464 - val_accuracy: 0.6654 -
val_auc: 0.7354
```

```
Epoch 337/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7073 -
accuracy: 0.7047 - auc: 0.7759 - val_loss: 0.7432 - val_accuracy: 0.6728 -
val_auc: 0.7393
Epoch 338/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7070 -
accuracy: 0.7018 - auc: 0.7755 - val_loss: 0.7466 - val_accuracy: 0.6617 -
val_auc: 0.7326
Epoch 339/1000
119/119 [=====] - 2s 21ms/step - loss: 0.7060 -
accuracy: 0.7015 - auc: 0.7768 - val_loss: 0.7480 - val_accuracy: 0.6605 -
val_auc: 0.7307
Epoch 340/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7055 -
accuracy: 0.6986 - auc: 0.7757 - val_loss: 0.7409 - val_accuracy: 0.6790 -
val_auc: 0.7395
Epoch 341/1000
119/119 [=====] - 3s 21ms/step - loss: 0.7061 -
accuracy: 0.6997 - auc: 0.7742 - val_loss: 0.7465 - val_accuracy: 0.6642 -
val_auc: 0.7316
Epoch 342/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7053 -
accuracy: 0.6975 - auc: 0.7749 - val_loss: 0.7406 - val_accuracy: 0.6765 -
val_auc: 0.7383
Epoch 343/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7036 -
accuracy: 0.7049 - auc: 0.7777 - val_loss: 0.7415 - val_accuracy: 0.6716 -
val_auc: 0.7373
Epoch 344/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7035 -
accuracy: 0.7047 - auc: 0.7769 - val_loss: 0.7436 - val_accuracy: 0.6691 -
val_auc: 0.7345
Epoch 345/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7027 -
accuracy: 0.7031 - auc: 0.7757 - val_loss: 0.7459 - val_accuracy: 0.6630 -
val_auc: 0.7301
Epoch 346/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7018 -
accuracy: 0.6957 - auc: 0.7755 - val_loss: 0.7404 - val_accuracy: 0.6716 -
val_auc: 0.7361
Epoch 347/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7029 -
accuracy: 0.7002 - auc: 0.7755 - val_loss: 0.7427 - val_accuracy: 0.6741 -
val_auc: 0.7341
Epoch 348/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7007 -
accuracy: 0.7010 - auc: 0.7762 - val_loss: 0.7368 - val_accuracy: 0.6815 -
val_auc: 0.7394
```

```
Epoch 349/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7020 -
accuracy: 0.6989 - auc: 0.7757 - val_loss: 0.7347 - val_accuracy: 0.6840 -
val_auc: 0.7419
Epoch 350/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7021 -
accuracy: 0.6989 - auc: 0.7734 - val_loss: 0.7409 - val_accuracy: 0.6704 -
val_auc: 0.7329
Epoch 351/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6979 -
accuracy: 0.7020 - auc: 0.7777 - val_loss: 0.7451 - val_accuracy: 0.6605 -
val_auc: 0.7275
Epoch 352/1000
119/119 [=====] - 2s 20ms/step - loss: 0.7000 -
accuracy: 0.6991 - auc: 0.7744 - val_loss: 0.7355 - val_accuracy: 0.6753 -
val_auc: 0.7398
Epoch 353/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6991 -
accuracy: 0.7015 - auc: 0.7749 - val_loss: 0.7404 - val_accuracy: 0.6630 -
val_auc: 0.7320
Epoch 354/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6966 -
accuracy: 0.7110 - auc: 0.7777 - val_loss: 0.7390 - val_accuracy: 0.6617 -
val_auc: 0.7319
Epoch 355/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6979 -
accuracy: 0.7068 - auc: 0.7756 - val_loss: 0.7384 - val_accuracy: 0.6679 -
val_auc: 0.7332
Epoch 356/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6961 -
accuracy: 0.7031 - auc: 0.7779 - val_loss: 0.7359 - val_accuracy: 0.6728 -
val_auc: 0.7352
Epoch 357/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6973 -
accuracy: 0.6978 - auc: 0.7739 - val_loss: 0.7372 - val_accuracy: 0.6691 -
val_auc: 0.7347
Epoch 358/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6973 -
accuracy: 0.7076 - auc: 0.7744 - val_loss: 0.7460 - val_accuracy: 0.6593 -
val_auc: 0.7239
Epoch 359/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6963 -
accuracy: 0.7036 - auc: 0.7757 - val_loss: 0.7346 - val_accuracy: 0.6691 -
val_auc: 0.7346
Epoch 360/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6967 -
accuracy: 0.6989 - auc: 0.7747 - val_loss: 0.7369 - val_accuracy: 0.6667 -
val_auc: 0.7315
```

```
Epoch 361/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6907 -
accuracy: 0.7028 - auc: 0.7800 - val_loss: 0.7324 - val_accuracy: 0.6704 -
val_auc: 0.7363
Epoch 362/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6937 -
accuracy: 0.7055 - auc: 0.7777 - val_loss: 0.7300 - val_accuracy: 0.6790 -
val_auc: 0.7398
Epoch 363/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6914 -
accuracy: 0.7052 - auc: 0.7785 - val_loss: 0.7365 - val_accuracy: 0.6679 -
val_auc: 0.7319
Epoch 364/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6921 -
accuracy: 0.7023 - auc: 0.7783 - val_loss: 0.7368 - val_accuracy: 0.6617 -
val_auc: 0.7293
Epoch 365/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6929 -
accuracy: 0.7087 - auc: 0.7763 - val_loss: 0.7358 - val_accuracy: 0.6630 -
val_auc: 0.7317
Epoch 366/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6920 -
accuracy: 0.6983 - auc: 0.7766 - val_loss: 0.7333 - val_accuracy: 0.6716 -
val_auc: 0.7348
Epoch 367/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6914 -
accuracy: 0.7047 - auc: 0.7769 - val_loss: 0.7346 - val_accuracy: 0.6630 -
val_auc: 0.7298
Epoch 368/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6880 -
accuracy: 0.7052 - auc: 0.7799 - val_loss: 0.7286 - val_accuracy: 0.6716 -
val_auc: 0.7400
Epoch 369/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6887 -
accuracy: 0.7010 - auc: 0.7789 - val_loss: 0.7394 - val_accuracy: 0.6531 -
val_auc: 0.7237
Epoch 370/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6894 -
accuracy: 0.7047 - auc: 0.7771 - val_loss: 0.7347 - val_accuracy: 0.6519 -
val_auc: 0.7278
Epoch 371/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6906 -
accuracy: 0.7039 - auc: 0.7753 - val_loss: 0.7350 - val_accuracy: 0.6593 -
val_auc: 0.7281
Epoch 372/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6911 -
accuracy: 0.7039 - auc: 0.7745 - val_loss: 0.7277 - val_accuracy: 0.6716 -
val_auc: 0.7370
```

```
Epoch 373/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6895 -
accuracy: 0.7020 - auc: 0.7755 - val_loss: 0.7336 - val_accuracy: 0.6556 -
val_auc: 0.7274
Epoch 374/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6871 -
accuracy: 0.7079 - auc: 0.7791 - val_loss: 0.7257 - val_accuracy: 0.6753 -
val_auc: 0.7386
Epoch 375/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6872 -
accuracy: 0.7015 - auc: 0.7776 - val_loss: 0.7243 - val_accuracy: 0.6815 -
val_auc: 0.7395
Epoch 376/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6862 -
accuracy: 0.7097 - auc: 0.7786 - val_loss: 0.7299 - val_accuracy: 0.6593 -
val_auc: 0.7310
Epoch 377/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6852 -
accuracy: 0.7047 - auc: 0.7789 - val_loss: 0.7302 - val_accuracy: 0.6654 -
val_auc: 0.7305
Epoch 378/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6881 -
accuracy: 0.7034 - auc: 0.7754 - val_loss: 0.7269 - val_accuracy: 0.6642 -
val_auc: 0.7332
Epoch 379/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6851 -
accuracy: 0.7031 - auc: 0.7771 - val_loss: 0.7246 - val_accuracy: 0.6741 -
val_auc: 0.7374
Epoch 380/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6849 -
accuracy: 0.7010 - auc: 0.7776 - val_loss: 0.7270 - val_accuracy: 0.6679 -
val_auc: 0.7326
Epoch 381/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6843 -
accuracy: 0.7007 - auc: 0.7766 - val_loss: 0.7272 - val_accuracy: 0.6617 -
val_auc: 0.7308
Epoch 382/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6825 -
accuracy: 0.7007 - auc: 0.7797 - val_loss: 0.7239 - val_accuracy: 0.6741 -
val_auc: 0.7372
Epoch 383/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6863 -
accuracy: 0.7015 - auc: 0.7753 - val_loss: 0.7224 - val_accuracy: 0.6667 -
val_auc: 0.7374
Epoch 384/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6837 -
accuracy: 0.7020 - auc: 0.7770 - val_loss: 0.7222 - val_accuracy: 0.6691 -
val_auc: 0.7361
```

```
Epoch 385/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6814 -
accuracy: 0.7028 - auc: 0.7793 - val_loss: 0.7266 - val_accuracy: 0.6605 -
val_auc: 0.7289
Epoch 386/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6801 -
accuracy: 0.7028 - auc: 0.7792 - val_loss: 0.7216 - val_accuracy: 0.6753 -
val_auc: 0.7360
Epoch 387/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6811 -
accuracy: 0.7087 - auc: 0.7792 - val_loss: 0.7268 - val_accuracy: 0.6593 -
val_auc: 0.7287
Epoch 388/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6826 -
accuracy: 0.7020 - auc: 0.7759 - val_loss: 0.7164 - val_accuracy: 0.6802 -
val_auc: 0.7411
Epoch 389/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6798 -
accuracy: 0.7002 - auc: 0.7778 - val_loss: 0.7208 - val_accuracy: 0.6617 -
val_auc: 0.7341
Epoch 390/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6801 -
accuracy: 0.6983 - auc: 0.7769 - val_loss: 0.7224 - val_accuracy: 0.6654 -
val_auc: 0.7333
Epoch 391/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6803 -
accuracy: 0.7031 - auc: 0.7781 - val_loss: 0.7218 - val_accuracy: 0.6691 -
val_auc: 0.7340
Epoch 392/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6778 -
accuracy: 0.7023 - auc: 0.7790 - val_loss: 0.7188 - val_accuracy: 0.6679 -
val_auc: 0.7360
Epoch 393/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6780 -
accuracy: 0.7081 - auc: 0.7793 - val_loss: 0.7236 - val_accuracy: 0.6593 -
val_auc: 0.7294
Epoch 394/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6784 -
accuracy: 0.7028 - auc: 0.7794 - val_loss: 0.7175 - val_accuracy: 0.6691 -
val_auc: 0.7362
Epoch 395/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6777 -
accuracy: 0.7031 - auc: 0.7786 - val_loss: 0.7237 - val_accuracy: 0.6556 -
val_auc: 0.7283
Epoch 396/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6771 -
accuracy: 0.7068 - auc: 0.7802 - val_loss: 0.7158 - val_accuracy: 0.6753 -
val_auc: 0.7399
```

```
Epoch 397/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6755 -
accuracy: 0.7073 - auc: 0.7802 - val_loss: 0.7220 - val_accuracy: 0.6741 -
val_auc: 0.7309
Epoch 398/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6783 -
accuracy: 0.7057 - auc: 0.7762 - val_loss: 0.7170 - val_accuracy: 0.6728 -
val_auc: 0.7364
Epoch 399/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6728 -
accuracy: 0.7047 - auc: 0.7823 - val_loss: 0.7147 - val_accuracy: 0.6753 -
val_auc: 0.7388
Epoch 400/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6745 -
accuracy: 0.7076 - auc: 0.7803 - val_loss: 0.7157 - val_accuracy: 0.6741 -
val_auc: 0.7366
Epoch 401/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6741 -
accuracy: 0.7034 - auc: 0.7797 - val_loss: 0.7291 - val_accuracy: 0.6568 -
val_auc: 0.7205
Epoch 402/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6715 -
accuracy: 0.7031 - auc: 0.7816 - val_loss: 0.7170 - val_accuracy: 0.6704 -
val_auc: 0.7351
Epoch 403/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6710 -
accuracy: 0.7065 - auc: 0.7826 - val_loss: 0.7145 - val_accuracy: 0.6679 -
val_auc: 0.7353
Epoch 404/1000
119/119 [=====] - 3s 22ms/step - loss: 0.6711 -
accuracy: 0.7089 - auc: 0.7811 - val_loss: 0.7169 - val_accuracy: 0.6654 -
val_auc: 0.7341
Epoch 405/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6704 -
accuracy: 0.7034 - auc: 0.7819 - val_loss: 0.7182 - val_accuracy: 0.6753 -
val_auc: 0.7338
Epoch 406/1000
119/119 [=====] - 3s 22ms/step - loss: 0.6693 -
accuracy: 0.7076 - auc: 0.7815 - val_loss: 0.7193 - val_accuracy: 0.6593 -
val_auc: 0.7290
Epoch 407/1000
119/119 [=====] - 3s 21ms/step - loss: 0.6722 -
accuracy: 0.7044 - auc: 0.7779 - val_loss: 0.7192 - val_accuracy: 0.6543 -
val_auc: 0.7275
Epoch 408/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6703 -
accuracy: 0.7023 - auc: 0.7793 - val_loss: 0.7097 - val_accuracy: 0.6765 -
val_auc: 0.7403
```

```
Epoch 409/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6683 -
accuracy: 0.7087 - auc: 0.7824 - val_loss: 0.7152 - val_accuracy: 0.6630 -
val_auc: 0.7333
Epoch 410/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6700 -
accuracy: 0.7076 - auc: 0.7821 - val_loss: 0.7102 - val_accuracy: 0.6691 -
val_auc: 0.7381
Epoch 411/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6682 -
accuracy: 0.7060 - auc: 0.7814 - val_loss: 0.7132 - val_accuracy: 0.6654 -
val_auc: 0.7343
Epoch 412/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6707 -
accuracy: 0.7100 - auc: 0.7802 - val_loss: 0.7108 - val_accuracy: 0.6667 -
val_auc: 0.7366
Epoch 413/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6682 -
accuracy: 0.7065 - auc: 0.7813 - val_loss: 0.7104 - val_accuracy: 0.6741 -
val_auc: 0.7378
Epoch 414/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6683 -
accuracy: 0.7063 - auc: 0.7801 - val_loss: 0.7073 - val_accuracy: 0.6753 -
val_auc: 0.7414
Epoch 415/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6675 -
accuracy: 0.7118 - auc: 0.7826 - val_loss: 0.7056 - val_accuracy: 0.6778 -
val_auc: 0.7423
Epoch 416/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6665 -
accuracy: 0.7052 - auc: 0.7824 - val_loss: 0.7122 - val_accuracy: 0.6704 -
val_auc: 0.7365
Epoch 417/1000
119/119 [=====] - 3s 22ms/step - loss: 0.6702 -
accuracy: 0.7026 - auc: 0.7770 - val_loss: 0.7099 - val_accuracy: 0.6691 -
val_auc: 0.7364
Epoch 418/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6644 -
accuracy: 0.7092 - auc: 0.7840 - val_loss: 0.7071 - val_accuracy: 0.6753 -
val_auc: 0.7381
Epoch 419/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6658 -
accuracy: 0.7039 - auc: 0.7803 - val_loss: 0.7070 - val_accuracy: 0.6716 -
val_auc: 0.7388
Epoch 420/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6645 -
accuracy: 0.7068 - auc: 0.7823 - val_loss: 0.7080 - val_accuracy: 0.6753 -
val_auc: 0.7381
```

```
Epoch 421/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6658 -
accuracy: 0.7071 - auc: 0.7801 - val_loss: 0.7059 - val_accuracy: 0.6753 -
val_auc: 0.7396
Epoch 422/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6657 -
accuracy: 0.7071 - auc: 0.7804 - val_loss: 0.7077 - val_accuracy: 0.6667 -
val_auc: 0.7364
Epoch 423/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6656 -
accuracy: 0.7002 - auc: 0.7800 - val_loss: 0.7112 - val_accuracy: 0.6617 -
val_auc: 0.7305
Epoch 424/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6639 -
accuracy: 0.7034 - auc: 0.7809 - val_loss: 0.7047 - val_accuracy: 0.6716 -
val_auc: 0.7392
Epoch 425/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6621 -
accuracy: 0.7094 - auc: 0.7828 - val_loss: 0.7041 - val_accuracy: 0.6753 -
val_auc: 0.7385
Epoch 426/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6646 -
accuracy: 0.7036 - auc: 0.7799 - val_loss: 0.7132 - val_accuracy: 0.6605 -
val_auc: 0.7274
Epoch 427/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6625 -
accuracy: 0.7094 - auc: 0.7821 - val_loss: 0.7050 - val_accuracy: 0.6741 -
val_auc: 0.7373
Epoch 428/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6615 -
accuracy: 0.7087 - auc: 0.7832 - val_loss: 0.7080 - val_accuracy: 0.6580 -
val_auc: 0.7324
Epoch 429/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6590 -
accuracy: 0.7124 - auc: 0.7847 - val_loss: 0.7002 - val_accuracy: 0.6778 -
val_auc: 0.7435
Epoch 430/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6606 -
accuracy: 0.7057 - auc: 0.7825 - val_loss: 0.7042 - val_accuracy: 0.6765 -
val_auc: 0.7387
Epoch 431/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6591 -
accuracy: 0.7076 - auc: 0.7837 - val_loss: 0.7056 - val_accuracy: 0.6617 -
val_auc: 0.7365
Epoch 432/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6630 -
accuracy: 0.7055 - auc: 0.7787 - val_loss: 0.7097 - val_accuracy: 0.6642 -
val_auc: 0.7298
```

```
Epoch 433/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6588 -
accuracy: 0.7071 - auc: 0.7827 - val_loss: 0.7069 - val_accuracy: 0.6716 -
val_auc: 0.7328
Epoch 434/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6589 -
accuracy: 0.7073 - auc: 0.7838 - val_loss: 0.6999 - val_accuracy: 0.6790 -
val_auc: 0.7420
Epoch 435/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6592 -
accuracy: 0.7089 - auc: 0.7820 - val_loss: 0.6988 - val_accuracy: 0.6778 -
val_auc: 0.7420
Epoch 436/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6600 -
accuracy: 0.7102 - auc: 0.7805 - val_loss: 0.7062 - val_accuracy: 0.6667 -
val_auc: 0.7326
Epoch 437/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6595 -
accuracy: 0.7068 - auc: 0.7808 - val_loss: 0.6983 - val_accuracy: 0.6741 -
val_auc: 0.7438
Epoch 438/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6571 -
accuracy: 0.7126 - auc: 0.7830 - val_loss: 0.7016 - val_accuracy: 0.6679 -
val_auc: 0.7374
Epoch 439/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6568 -
accuracy: 0.7052 - auc: 0.7825 - val_loss: 0.6996 - val_accuracy: 0.6778 -
val_auc: 0.7409
Epoch 440/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6572 -
accuracy: 0.7076 - auc: 0.7825 - val_loss: 0.7043 - val_accuracy: 0.6679 -
val_auc: 0.7330
Epoch 441/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6552 -
accuracy: 0.7116 - auc: 0.7834 - val_loss: 0.7033 - val_accuracy: 0.6630 -
val_auc: 0.7332
Epoch 442/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6557 -
accuracy: 0.7044 - auc: 0.7832 - val_loss: 0.7036 - val_accuracy: 0.6580 -
val_auc: 0.7330
Epoch 443/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6533 -
accuracy: 0.7102 - auc: 0.7851 - val_loss: 0.7051 - val_accuracy: 0.6728 -
val_auc: 0.7331
Epoch 444/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6541 -
accuracy: 0.7052 - auc: 0.7845 - val_loss: 0.6968 - val_accuracy: 0.6753 -
val_auc: 0.7424
```

```
Epoch 445/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6561 -
accuracy: 0.7102 - auc: 0.7820 - val_loss: 0.7042 - val_accuracy: 0.6630 -
val_auc: 0.7298
Epoch 446/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6526 -
accuracy: 0.7137 - auc: 0.7856 - val_loss: 0.6963 - val_accuracy: 0.6765 -
val_auc: 0.7419
Epoch 447/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6552 -
accuracy: 0.7094 - auc: 0.7832 - val_loss: 0.7000 - val_accuracy: 0.6691 -
val_auc: 0.7368
Epoch 448/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6533 -
accuracy: 0.7094 - auc: 0.7837 - val_loss: 0.7016 - val_accuracy: 0.6728 -
val_auc: 0.7349
Epoch 449/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6514 -
accuracy: 0.7108 - auc: 0.7855 - val_loss: 0.6997 - val_accuracy: 0.6667 -
val_auc: 0.7346
Epoch 450/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6532 -
accuracy: 0.7087 - auc: 0.7829 - val_loss: 0.6959 - val_accuracy: 0.6753 -
val_auc: 0.7406
Epoch 451/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6519 -
accuracy: 0.7065 - auc: 0.7840 - val_loss: 0.6955 - val_accuracy: 0.6765 -
val_auc: 0.7413
Epoch 452/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6529 -
accuracy: 0.7100 - auc: 0.7838 - val_loss: 0.7009 - val_accuracy: 0.6691 -
val_auc: 0.7345
Epoch 453/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6496 -
accuracy: 0.7079 - auc: 0.7863 - val_loss: 0.7042 - val_accuracy: 0.6580 -
val_auc: 0.7279
Epoch 454/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6519 -
accuracy: 0.7102 - auc: 0.7828 - val_loss: 0.6916 - val_accuracy: 0.6765 -
val_auc: 0.7427
Epoch 455/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6498 -
accuracy: 0.7097 - auc: 0.7854 - val_loss: 0.6961 - val_accuracy: 0.6691 -
val_auc: 0.7383
Epoch 456/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6507 -
accuracy: 0.7150 - auc: 0.7852 - val_loss: 0.6953 - val_accuracy: 0.6753 -
val_auc: 0.7408
```

```
Epoch 457/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6513 -
accuracy: 0.7087 - auc: 0.7832 - val_loss: 0.6969 - val_accuracy: 0.6753 -
val_auc: 0.7380
Epoch 458/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6524 -
accuracy: 0.7092 - auc: 0.7815 - val_loss: 0.6958 - val_accuracy: 0.6704 -
val_auc: 0.7358
Epoch 459/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6464 -
accuracy: 0.7089 - auc: 0.7889 - val_loss: 0.6907 - val_accuracy: 0.6790 -
val_auc: 0.7428
Epoch 460/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6479 -
accuracy: 0.7105 - auc: 0.7862 - val_loss: 0.6915 - val_accuracy: 0.6802 -
val_auc: 0.7413
Epoch 461/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6483 -
accuracy: 0.7071 - auc: 0.7842 - val_loss: 0.6967 - val_accuracy: 0.6728 -
val_auc: 0.7362
Epoch 462/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6484 -
accuracy: 0.7137 - auc: 0.7857 - val_loss: 0.6927 - val_accuracy: 0.6765 -
val_auc: 0.7405
Epoch 463/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6493 -
accuracy: 0.7092 - auc: 0.7830 - val_loss: 0.6897 - val_accuracy: 0.6827 -
val_auc: 0.7434
Epoch 464/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6475 -
accuracy: 0.7063 - auc: 0.7856 - val_loss: 0.6949 - val_accuracy: 0.6691 -
val_auc: 0.7354
Epoch 465/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6452 -
accuracy: 0.7129 - auc: 0.7868 - val_loss: 0.6968 - val_accuracy: 0.6642 -
val_auc: 0.7319
Epoch 466/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6479 -
accuracy: 0.7113 - auc: 0.7838 - val_loss: 0.6918 - val_accuracy: 0.6753 -
val_auc: 0.7405
Epoch 467/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6465 -
accuracy: 0.7110 - auc: 0.7854 - val_loss: 0.6942 - val_accuracy: 0.6667 -
val_auc: 0.7355
Epoch 468/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6463 -
accuracy: 0.7142 - auc: 0.7865 - val_loss: 0.6932 - val_accuracy: 0.6753 -
val_auc: 0.7371
```

```
Epoch 469/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6440 -
accuracy: 0.7102 - auc: 0.7881 - val_loss: 0.6901 - val_accuracy: 0.6704 -
val_auc: 0.7387
Epoch 470/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6447 -
accuracy: 0.7092 - auc: 0.7868 - val_loss: 0.6937 - val_accuracy: 0.6765 -
val_auc: 0.7370
Epoch 471/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6455 -
accuracy: 0.7155 - auc: 0.7864 - val_loss: 0.6884 - val_accuracy: 0.6704 -
val_auc: 0.7414
Epoch 472/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6442 -
accuracy: 0.7124 - auc: 0.7860 - val_loss: 0.6900 - val_accuracy: 0.6741 -
val_auc: 0.7415
Epoch 473/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6439 -
accuracy: 0.7100 - auc: 0.7850 - val_loss: 0.6896 - val_accuracy: 0.6728 -
val_auc: 0.7401
Epoch 474/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6435 -
accuracy: 0.7094 - auc: 0.7855 - val_loss: 0.7057 - val_accuracy: 0.6580 -
val_auc: 0.7192
Epoch 475/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6431 -
accuracy: 0.7108 - auc: 0.7862 - val_loss: 0.6921 - val_accuracy: 0.6753 -
val_auc: 0.7381
Epoch 476/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6427 -
accuracy: 0.7044 - auc: 0.7864 - val_loss: 0.6911 - val_accuracy: 0.6691 -
val_auc: 0.7379
Epoch 477/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6453 -
accuracy: 0.7071 - auc: 0.7833 - val_loss: 0.6892 - val_accuracy: 0.6728 -
val_auc: 0.7395
Epoch 478/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6427 -
accuracy: 0.7089 - auc: 0.7860 - val_loss: 0.6899 - val_accuracy: 0.6704 -
val_auc: 0.7378
Epoch 479/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6410 -
accuracy: 0.7145 - auc: 0.7880 - val_loss: 0.6880 - val_accuracy: 0.6704 -
val_auc: 0.7409
Epoch 480/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6431 -
accuracy: 0.7102 - auc: 0.7858 - val_loss: 0.6830 - val_accuracy: 0.6815 -
val_auc: 0.7469
```

```
Epoch 481/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6422 -
accuracy: 0.7065 - auc: 0.7859 - val_loss: 0.6929 - val_accuracy: 0.6753 -
val_auc: 0.7342
Epoch 482/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6424 -
accuracy: 0.7134 - auc: 0.7856 - val_loss: 0.6863 - val_accuracy: 0.6704 -
val_auc: 0.7412
Epoch 483/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6405 -
accuracy: 0.7132 - auc: 0.7873 - val_loss: 0.6816 - val_accuracy: 0.6852 -
val_auc: 0.7467
Epoch 484/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6409 -
accuracy: 0.7134 - auc: 0.7867 - val_loss: 0.6864 - val_accuracy: 0.6716 -
val_auc: 0.7387
Epoch 485/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6392 -
accuracy: 0.7132 - auc: 0.7879 - val_loss: 0.6838 - val_accuracy: 0.6765 -
val_auc: 0.7424
Epoch 486/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6382 -
accuracy: 0.7087 - auc: 0.7880 - val_loss: 0.6824 - val_accuracy: 0.6790 -
val_auc: 0.7431
Epoch 487/1000
119/119 [=====] - 3s 21ms/step - loss: 0.6393 -
accuracy: 0.7100 - auc: 0.7869 - val_loss: 0.6821 - val_accuracy: 0.6802 -
val_auc: 0.7436
Epoch 488/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6396 -
accuracy: 0.7105 - auc: 0.7864 - val_loss: 0.6878 - val_accuracy: 0.6704 -
val_auc: 0.7373
Epoch 489/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6392 -
accuracy: 0.7132 - auc: 0.7860 - val_loss: 0.6856 - val_accuracy: 0.6728 -
val_auc: 0.7398
Epoch 490/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6376 -
accuracy: 0.7132 - auc: 0.7884 - val_loss: 0.6840 - val_accuracy: 0.6741 -
val_auc: 0.7408
Epoch 491/1000
119/119 [=====] - 3s 21ms/step - loss: 0.6398 -
accuracy: 0.7139 - auc: 0.7852 - val_loss: 0.6903 - val_accuracy: 0.6728 -
val_auc: 0.7345
Epoch 492/1000
119/119 [=====] - 3s 22ms/step - loss: 0.6372 -
accuracy: 0.7110 - auc: 0.7889 - val_loss: 0.6858 - val_accuracy: 0.6704 -
val_auc: 0.7389
```

```
Epoch 493/1000
119/119 [=====] - 3s 22ms/step - loss: 0.6362 -
accuracy: 0.7087 - auc: 0.7888 - val_loss: 0.6880 - val_accuracy: 0.6728 -
val_auc: 0.7366
Epoch 494/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6371 -
accuracy: 0.7163 - auc: 0.7873 - val_loss: 0.6805 - val_accuracy: 0.6778 -
val_auc: 0.7447
Epoch 495/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6371 -
accuracy: 0.7092 - auc: 0.7863 - val_loss: 0.6821 - val_accuracy: 0.6765 -
val_auc: 0.7424
Epoch 496/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6350 -
accuracy: 0.7163 - auc: 0.7897 - val_loss: 0.6802 - val_accuracy: 0.6790 -
val_auc: 0.7431
Epoch 497/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6387 -
accuracy: 0.7087 - auc: 0.7837 - val_loss: 0.6815 - val_accuracy: 0.6790 -
val_auc: 0.7419
Epoch 498/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6378 -
accuracy: 0.7126 - auc: 0.7853 - val_loss: 0.6868 - val_accuracy: 0.6716 -
val_auc: 0.7351
Epoch 499/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6350 -
accuracy: 0.7142 - auc: 0.7892 - val_loss: 0.6818 - val_accuracy: 0.6704 -
val_auc: 0.7399
Epoch 500/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6372 -
accuracy: 0.7150 - auc: 0.7860 - val_loss: 0.6787 - val_accuracy: 0.6765 -
val_auc: 0.7440
Epoch 501/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6364 -
accuracy: 0.7105 - auc: 0.7856 - val_loss: 0.6854 - val_accuracy: 0.6741 -
val_auc: 0.7366
Epoch 502/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6350 -
accuracy: 0.7153 - auc: 0.7884 - val_loss: 0.6781 - val_accuracy: 0.6802 -
val_auc: 0.7434
Epoch 503/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6346 -
accuracy: 0.7124 - auc: 0.7892 - val_loss: 0.6802 - val_accuracy: 0.6765 -
val_auc: 0.7435
Epoch 504/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6319 -
accuracy: 0.7192 - auc: 0.7912 - val_loss: 0.6796 - val_accuracy: 0.6790 -
val_auc: 0.7442
```

```
Epoch 505/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6330 -
accuracy: 0.7216 - auc: 0.7897 - val_loss: 0.6780 - val_accuracy: 0.6815 -
val_auc: 0.7449
Epoch 506/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6323 -
accuracy: 0.7177 - auc: 0.7902 - val_loss: 0.6813 - val_accuracy: 0.6691 -
val_auc: 0.7371
Epoch 507/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6334 -
accuracy: 0.7145 - auc: 0.7878 - val_loss: 0.6768 - val_accuracy: 0.6802 -
val_auc: 0.7447
Epoch 508/1000
119/119 [=====] - 3s 21ms/step - loss: 0.6325 -
accuracy: 0.7177 - auc: 0.7893 - val_loss: 0.6779 - val_accuracy: 0.6753 -
val_auc: 0.7443
Epoch 509/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6340 -
accuracy: 0.7087 - auc: 0.7862 - val_loss: 0.6815 - val_accuracy: 0.6728 -
val_auc: 0.7383
Epoch 510/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6330 -
accuracy: 0.7171 - auc: 0.7890 - val_loss: 0.6778 - val_accuracy: 0.6802 -
val_auc: 0.7431
Epoch 511/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6294 -
accuracy: 0.7166 - auc: 0.7929 - val_loss: 0.6772 - val_accuracy: 0.6815 -
val_auc: 0.7425
Epoch 512/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6308 -
accuracy: 0.7105 - auc: 0.7892 - val_loss: 0.6796 - val_accuracy: 0.6778 -
val_auc: 0.7415
Epoch 513/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6326 -
accuracy: 0.7108 - auc: 0.7870 - val_loss: 0.6777 - val_accuracy: 0.6802 -
val_auc: 0.7441
Epoch 514/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6301 -
accuracy: 0.7161 - auc: 0.7895 - val_loss: 0.6808 - val_accuracy: 0.6716 -
val_auc: 0.7385
Epoch 515/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6289 -
accuracy: 0.7163 - auc: 0.7915 - val_loss: 0.6752 - val_accuracy: 0.6765 -
val_auc: 0.7451
Epoch 516/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6301 -
accuracy: 0.7129 - auc: 0.7895 - val_loss: 0.6716 - val_accuracy: 0.6790 -
val_auc: 0.7485
```

```
Epoch 517/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6290 -
accuracy: 0.7192 - auc: 0.7909 - val_loss: 0.6726 - val_accuracy: 0.6877 -
val_auc: 0.7490
Epoch 518/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6279 -
accuracy: 0.7166 - auc: 0.7909 - val_loss: 0.6767 - val_accuracy: 0.6840 -
val_auc: 0.7433
Epoch 519/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6299 -
accuracy: 0.7134 - auc: 0.7884 - val_loss: 0.6732 - val_accuracy: 0.6815 -
val_auc: 0.7464
Epoch 520/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6302 -
accuracy: 0.7129 - auc: 0.7872 - val_loss: 0.6822 - val_accuracy: 0.6716 -
val_auc: 0.7353
Epoch 521/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6290 -
accuracy: 0.7129 - auc: 0.7906 - val_loss: 0.6817 - val_accuracy: 0.6716 -
val_auc: 0.7337
Epoch 522/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6283 -
accuracy: 0.7147 - auc: 0.7898 - val_loss: 0.6746 - val_accuracy: 0.6827 -
val_auc: 0.7440
Epoch 523/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6273 -
accuracy: 0.7153 - auc: 0.7906 - val_loss: 0.6777 - val_accuracy: 0.6716 -
val_auc: 0.7414
Epoch 524/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6272 -
accuracy: 0.7166 - auc: 0.7910 - val_loss: 0.6762 - val_accuracy: 0.6790 -
val_auc: 0.7426
Epoch 525/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6267 -
accuracy: 0.7163 - auc: 0.7911 - val_loss: 0.6753 - val_accuracy: 0.6790 -
val_auc: 0.7419
Epoch 526/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6276 -
accuracy: 0.7192 - auc: 0.7889 - val_loss: 0.6774 - val_accuracy: 0.6778 -
val_auc: 0.7409
Epoch 527/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6315 -
accuracy: 0.7150 - auc: 0.7859 - val_loss: 0.6732 - val_accuracy: 0.6840 -
val_auc: 0.7437
Epoch 528/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6246 -
accuracy: 0.7097 - auc: 0.7912 - val_loss: 0.6818 - val_accuracy: 0.6679 -
val_auc: 0.7354
```

```
Epoch 529/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6248 -
accuracy: 0.7211 - auc: 0.7928 - val_loss: 0.6729 - val_accuracy: 0.6753 -
val_auc: 0.7453
Epoch 530/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6261 -
accuracy: 0.7158 - auc: 0.7906 - val_loss: 0.6741 - val_accuracy: 0.6753 -
val_auc: 0.7426
Epoch 531/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6261 -
accuracy: 0.7174 - auc: 0.7908 - val_loss: 0.6709 - val_accuracy: 0.6815 -
val_auc: 0.7456
Epoch 532/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6262 -
accuracy: 0.7137 - auc: 0.7899 - val_loss: 0.6790 - val_accuracy: 0.6741 -
val_auc: 0.7353
Epoch 533/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6258 -
accuracy: 0.7192 - auc: 0.7897 - val_loss: 0.6727 - val_accuracy: 0.6790 -
val_auc: 0.7435
Epoch 534/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6240 -
accuracy: 0.7158 - auc: 0.7911 - val_loss: 0.6737 - val_accuracy: 0.6716 -
val_auc: 0.7427
Epoch 535/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6262 -
accuracy: 0.7113 - auc: 0.7884 - val_loss: 0.6759 - val_accuracy: 0.6667 -
val_auc: 0.7389
Epoch 536/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6259 -
accuracy: 0.7137 - auc: 0.7899 - val_loss: 0.6760 - val_accuracy: 0.6704 -
val_auc: 0.7396
Epoch 537/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6240 -
accuracy: 0.7158 - auc: 0.7919 - val_loss: 0.6734 - val_accuracy: 0.6716 -
val_auc: 0.7428
Epoch 538/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6256 -
accuracy: 0.7121 - auc: 0.7890 - val_loss: 0.6711 - val_accuracy: 0.6815 -
val_auc: 0.7429
Epoch 539/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6239 -
accuracy: 0.7182 - auc: 0.7906 - val_loss: 0.6707 - val_accuracy: 0.6815 -
val_auc: 0.7448
Epoch 540/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6230 -
accuracy: 0.7147 - auc: 0.7908 - val_loss: 0.6733 - val_accuracy: 0.6716 -
val_auc: 0.7397
```

```
Epoch 541/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6245 -
accuracy: 0.7139 - auc: 0.7889 - val_loss: 0.6708 - val_accuracy: 0.6840 -
val_auc: 0.7457
Epoch 542/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6245 -
accuracy: 0.7158 - auc: 0.7899 - val_loss: 0.6721 - val_accuracy: 0.6741 -
val_auc: 0.7419
Epoch 543/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6212 -
accuracy: 0.7219 - auc: 0.7940 - val_loss: 0.6713 - val_accuracy: 0.6741 -
val_auc: 0.7432
Epoch 544/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6221 -
accuracy: 0.7155 - auc: 0.7921 - val_loss: 0.6679 - val_accuracy: 0.6802 -
val_auc: 0.7471
Epoch 545/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6227 -
accuracy: 0.7179 - auc: 0.7905 - val_loss: 0.6766 - val_accuracy: 0.6691 -
val_auc: 0.7353
Epoch 546/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6228 -
accuracy: 0.7116 - auc: 0.7898 - val_loss: 0.6763 - val_accuracy: 0.6704 -
val_auc: 0.7342
Epoch 547/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6189 -
accuracy: 0.7192 - auc: 0.7947 - val_loss: 0.6711 - val_accuracy: 0.6728 -
val_auc: 0.7430
Epoch 548/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6221 -
accuracy: 0.7169 - auc: 0.7913 - val_loss: 0.6770 - val_accuracy: 0.6753 -
val_auc: 0.7347
Epoch 549/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6236 -
accuracy: 0.7182 - auc: 0.7885 - val_loss: 0.6704 - val_accuracy: 0.6741 -
val_auc: 0.7427
Epoch 550/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6189 -
accuracy: 0.7184 - auc: 0.7939 - val_loss: 0.6676 - val_accuracy: 0.6802 -
val_auc: 0.7455
Epoch 551/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6186 -
accuracy: 0.7179 - auc: 0.7944 - val_loss: 0.6660 - val_accuracy: 0.6864 -
val_auc: 0.7476
Epoch 552/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6187 -
accuracy: 0.7158 - auc: 0.7933 - val_loss: 0.6675 - val_accuracy: 0.6802 -
val_auc: 0.7456
```

```
Epoch 553/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6213 -
accuracy: 0.7179 - auc: 0.7909 - val_loss: 0.6711 - val_accuracy: 0.6741 -
val_auc: 0.7418
Epoch 554/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6199 -
accuracy: 0.7150 - auc: 0.7911 - val_loss: 0.6650 - val_accuracy: 0.6815 -
val_auc: 0.7494
Epoch 555/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6211 -
accuracy: 0.7177 - auc: 0.7906 - val_loss: 0.6721 - val_accuracy: 0.6716 -
val_auc: 0.7396
Epoch 556/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6204 -
accuracy: 0.7129 - auc: 0.7903 - val_loss: 0.6743 - val_accuracy: 0.6667 -
val_auc: 0.7370
Epoch 557/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6195 -
accuracy: 0.7132 - auc: 0.7908 - val_loss: 0.6653 - val_accuracy: 0.6852 -
val_auc: 0.7474
Epoch 558/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6181 -
accuracy: 0.7182 - auc: 0.7924 - val_loss: 0.6697 - val_accuracy: 0.6765 -
val_auc: 0.7418
Epoch 559/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6206 -
accuracy: 0.7166 - auc: 0.7901 - val_loss: 0.6707 - val_accuracy: 0.6704 -
val_auc: 0.7401
Epoch 560/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6182 -
accuracy: 0.7145 - auc: 0.7919 - val_loss: 0.6664 - val_accuracy: 0.6728 -
val_auc: 0.7445
Epoch 561/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6170 -
accuracy: 0.7216 - auc: 0.7940 - val_loss: 0.6639 - val_accuracy: 0.6864 -
val_auc: 0.7481
Epoch 562/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6176 -
accuracy: 0.7161 - auc: 0.7925 - val_loss: 0.6705 - val_accuracy: 0.6679 -
val_auc: 0.7409
Epoch 563/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6158 -
accuracy: 0.7145 - auc: 0.7938 - val_loss: 0.6671 - val_accuracy: 0.6753 -
val_auc: 0.7446
Epoch 564/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6181 -
accuracy: 0.7142 - auc: 0.7911 - val_loss: 0.6666 - val_accuracy: 0.6815 -
val_auc: 0.7442
```

```
Epoch 565/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6201 -
accuracy: 0.7132 - auc: 0.7888 - val_loss: 0.6654 - val_accuracy: 0.6815 -
val_auc: 0.7449
Epoch 566/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6140 -
accuracy: 0.7211 - auc: 0.7961 - val_loss: 0.6627 - val_accuracy: 0.6802 -
val_auc: 0.7478
Epoch 567/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6177 -
accuracy: 0.7158 - auc: 0.7910 - val_loss: 0.6616 - val_accuracy: 0.6840 -
val_auc: 0.7494
Epoch 568/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6211 -
accuracy: 0.7211 - auc: 0.7887 - val_loss: 0.6644 - val_accuracy: 0.6753 -
val_auc: 0.7446
Epoch 569/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6172 -
accuracy: 0.7179 - auc: 0.7923 - val_loss: 0.6637 - val_accuracy: 0.6790 -
val_auc: 0.7460
Epoch 570/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6158 -
accuracy: 0.7129 - auc: 0.7922 - val_loss: 0.6602 - val_accuracy: 0.6815 -
val_auc: 0.7504
Epoch 571/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6158 -
accuracy: 0.7116 - auc: 0.7919 - val_loss: 0.6640 - val_accuracy: 0.6840 -
val_auc: 0.7460
Epoch 572/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6132 -
accuracy: 0.7169 - auc: 0.7954 - val_loss: 0.6656 - val_accuracy: 0.6753 -
val_auc: 0.7429
Epoch 573/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6169 -
accuracy: 0.7153 - auc: 0.7914 - val_loss: 0.6607 - val_accuracy: 0.6852 -
val_auc: 0.7478
Epoch 574/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6175 -
accuracy: 0.7163 - auc: 0.7885 - val_loss: 0.6695 - val_accuracy: 0.6691 -
val_auc: 0.7408
Epoch 575/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6167 -
accuracy: 0.7126 - auc: 0.7906 - val_loss: 0.6657 - val_accuracy: 0.6728 -
val_auc: 0.7420
Epoch 576/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6162 -
accuracy: 0.7158 - auc: 0.7909 - val_loss: 0.6612 - val_accuracy: 0.6852 -
val_auc: 0.7481
```

```
Epoch 577/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6161 -
accuracy: 0.7118 - auc: 0.7913 - val_loss: 0.6696 - val_accuracy: 0.6728 -
val_auc: 0.7377
Epoch 578/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6144 -
accuracy: 0.7161 - auc: 0.7943 - val_loss: 0.6671 - val_accuracy: 0.6728 -
val_auc: 0.7392
Epoch 579/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6175 -
accuracy: 0.7166 - auc: 0.7910 - val_loss: 0.6659 - val_accuracy: 0.6716 -
val_auc: 0.7423
Epoch 580/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6123 -
accuracy: 0.7195 - auc: 0.7957 - val_loss: 0.6656 - val_accuracy: 0.6753 -
val_auc: 0.7432
Epoch 581/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6137 -
accuracy: 0.7208 - auc: 0.7932 - val_loss: 0.6633 - val_accuracy: 0.6778 -
val_auc: 0.7459
Epoch 582/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6144 -
accuracy: 0.7216 - auc: 0.7931 - val_loss: 0.6700 - val_accuracy: 0.6716 -
val_auc: 0.7351
Epoch 583/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6130 -
accuracy: 0.7163 - auc: 0.7937 - val_loss: 0.6624 - val_accuracy: 0.6790 -
val_auc: 0.7447
Epoch 584/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6138 -
accuracy: 0.7184 - auc: 0.7930 - val_loss: 0.6600 - val_accuracy: 0.6815 -
val_auc: 0.7475
Epoch 585/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6120 -
accuracy: 0.7192 - auc: 0.7947 - val_loss: 0.6633 - val_accuracy: 0.6679 -
val_auc: 0.7418
Epoch 586/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6133 -
accuracy: 0.7163 - auc: 0.7928 - val_loss: 0.6658 - val_accuracy: 0.6716 -
val_auc: 0.7421
Epoch 587/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6148 -
accuracy: 0.7192 - auc: 0.7913 - val_loss: 0.6635 - val_accuracy: 0.6728 -
val_auc: 0.7405
Epoch 588/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6122 -
accuracy: 0.7129 - auc: 0.7933 - val_loss: 0.6613 - val_accuracy: 0.6802 -
val_auc: 0.7451
```

```
Epoch 589/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6130 -
accuracy: 0.7216 - auc: 0.7925 - val_loss: 0.6624 - val_accuracy: 0.6741 -
val_auc: 0.7442
Epoch 590/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6126 -
accuracy: 0.7153 - auc: 0.7930 - val_loss: 0.6593 - val_accuracy: 0.6852 -
val_auc: 0.7475
Epoch 591/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6106 -
accuracy: 0.7214 - auc: 0.7959 - val_loss: 0.6594 - val_accuracy: 0.6815 -
val_auc: 0.7482
Epoch 592/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6144 -
accuracy: 0.7147 - auc: 0.7902 - val_loss: 0.6574 - val_accuracy: 0.6877 -
val_auc: 0.7495
Epoch 593/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6125 -
accuracy: 0.7142 - auc: 0.7918 - val_loss: 0.6572 - val_accuracy: 0.6790 -
val_auc: 0.7482
Epoch 594/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6104 -
accuracy: 0.7169 - auc: 0.7942 - val_loss: 0.6599 - val_accuracy: 0.6827 -
val_auc: 0.7462
Epoch 595/1000
119/119 [=====] - 3s 22ms/step - loss: 0.6113 -
accuracy: 0.7227 - auc: 0.7937 - val_loss: 0.6645 - val_accuracy: 0.6716 -
val_auc: 0.7393
Epoch 596/1000
119/119 [=====] - 3s 22ms/step - loss: 0.6098 -
accuracy: 0.7179 - auc: 0.7941 - val_loss: 0.6693 - val_accuracy: 0.6765 -
val_auc: 0.7330
Epoch 597/1000
119/119 [=====] - 3s 21ms/step - loss: 0.6090 -
accuracy: 0.7184 - auc: 0.7947 - val_loss: 0.6603 - val_accuracy: 0.6753 -
val_auc: 0.7444
Epoch 598/1000
119/119 [=====] - 3s 22ms/step - loss: 0.6100 -
accuracy: 0.7195 - auc: 0.7945 - val_loss: 0.6604 - val_accuracy: 0.6790 -
val_auc: 0.7449
Epoch 599/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6122 -
accuracy: 0.7118 - auc: 0.7917 - val_loss: 0.6606 - val_accuracy: 0.6802 -
val_auc: 0.7434
Epoch 600/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6100 -
accuracy: 0.7211 - auc: 0.7957 - val_loss: 0.6581 - val_accuracy: 0.6864 -
val_auc: 0.7482
```

```
Epoch 601/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6102 -
accuracy: 0.7177 - auc: 0.7931 - val_loss: 0.6628 - val_accuracy: 0.6790 -
val_auc: 0.7452
Epoch 602/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6084 -
accuracy: 0.7203 - auc: 0.7955 - val_loss: 0.6624 - val_accuracy: 0.6765 -
val_auc: 0.7435
Epoch 603/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6115 -
accuracy: 0.7137 - auc: 0.7920 - val_loss: 0.6580 - val_accuracy: 0.6802 -
val_auc: 0.7466
Epoch 604/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6098 -
accuracy: 0.7198 - auc: 0.7934 - val_loss: 0.6639 - val_accuracy: 0.6778 -
val_auc: 0.7414
Epoch 605/1000
119/119 [=====] - 3s 21ms/step - loss: 0.6073 -
accuracy: 0.7150 - auc: 0.7948 - val_loss: 0.6592 - val_accuracy: 0.6802 -
val_auc: 0.7452
Epoch 606/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6114 -
accuracy: 0.7179 - auc: 0.7909 - val_loss: 0.6591 - val_accuracy: 0.6790 -
val_auc: 0.7447
Epoch 607/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6076 -
accuracy: 0.7174 - auc: 0.7952 - val_loss: 0.6586 - val_accuracy: 0.6815 -
val_auc: 0.7462
Epoch 608/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6093 -
accuracy: 0.7163 - auc: 0.7937 - val_loss: 0.6587 - val_accuracy: 0.6802 -
val_auc: 0.7451
Epoch 609/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6118 -
accuracy: 0.7108 - auc: 0.7906 - val_loss: 0.6563 - val_accuracy: 0.6741 -
val_auc: 0.7469
Epoch 610/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6115 -
accuracy: 0.7179 - auc: 0.7909 - val_loss: 0.6582 - val_accuracy: 0.6765 -
val_auc: 0.7445
Epoch 611/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6107 -
accuracy: 0.7187 - auc: 0.7912 - val_loss: 0.6635 - val_accuracy: 0.6716 -
val_auc: 0.7390
Epoch 612/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6080 -
accuracy: 0.7224 - auc: 0.7946 - val_loss: 0.6572 - val_accuracy: 0.6790 -
val_auc: 0.7459
```

```
Epoch 613/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6071 -
accuracy: 0.7182 - auc: 0.7956 - val_loss: 0.6617 - val_accuracy: 0.6741 -
val_auc: 0.7436
Epoch 614/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6074 -
accuracy: 0.7198 - auc: 0.7957 - val_loss: 0.6543 - val_accuracy: 0.6827 -
val_auc: 0.7501
Epoch 615/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6064 -
accuracy: 0.7206 - auc: 0.7957 - val_loss: 0.6657 - val_accuracy: 0.6716 -
val_auc: 0.7372
Epoch 616/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6070 -
accuracy: 0.7200 - auc: 0.7945 - val_loss: 0.6579 - val_accuracy: 0.6728 -
val_auc: 0.7443
Epoch 617/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6077 -
accuracy: 0.7224 - auc: 0.7942 - val_loss: 0.6536 - val_accuracy: 0.6864 -
val_auc: 0.7510
Epoch 618/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6069 -
accuracy: 0.7163 - auc: 0.7951 - val_loss: 0.6611 - val_accuracy: 0.6691 -
val_auc: 0.7415
Epoch 619/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6037 -
accuracy: 0.7232 - auc: 0.7963 - val_loss: 0.6614 - val_accuracy: 0.6728 -
val_auc: 0.7404
Epoch 620/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6053 -
accuracy: 0.7219 - auc: 0.7949 - val_loss: 0.6593 - val_accuracy: 0.6741 -
val_auc: 0.7452
Epoch 621/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6062 -
accuracy: 0.7208 - auc: 0.7950 - val_loss: 0.6574 - val_accuracy: 0.6802 -
val_auc: 0.7453
Epoch 622/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6068 -
accuracy: 0.7200 - auc: 0.7938 - val_loss: 0.6576 - val_accuracy: 0.6728 -
val_auc: 0.7456
Epoch 623/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6075 -
accuracy: 0.7161 - auc: 0.7925 - val_loss: 0.6547 - val_accuracy: 0.6840 -
val_auc: 0.7478
Epoch 624/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6056 -
accuracy: 0.7227 - auc: 0.7958 - val_loss: 0.6545 - val_accuracy: 0.6852 -
val_auc: 0.7492
```

```
Epoch 625/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6087 -
accuracy: 0.7195 - auc: 0.7910 - val_loss: 0.6564 - val_accuracy: 0.6741 -
val_auc: 0.7448
Epoch 626/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6065 -
accuracy: 0.7211 - auc: 0.7939 - val_loss: 0.6553 - val_accuracy: 0.6852 -
val_auc: 0.7475
Epoch 627/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6035 -
accuracy: 0.7229 - auc: 0.7973 - val_loss: 0.6545 - val_accuracy: 0.6827 -
val_auc: 0.7466
Epoch 628/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6067 -
accuracy: 0.7177 - auc: 0.7935 - val_loss: 0.6557 - val_accuracy: 0.6741 -
val_auc: 0.7455
Epoch 629/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6065 -
accuracy: 0.7214 - auc: 0.7940 - val_loss: 0.6541 - val_accuracy: 0.6827 -
val_auc: 0.7482
Epoch 630/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6049 -
accuracy: 0.7166 - auc: 0.7951 - val_loss: 0.6536 - val_accuracy: 0.6802 -
val_auc: 0.7475
Epoch 631/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6033 -
accuracy: 0.7219 - auc: 0.7973 - val_loss: 0.6518 - val_accuracy: 0.6815 -
val_auc: 0.7496
Epoch 632/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6032 -
accuracy: 0.7221 - auc: 0.7967 - val_loss: 0.6596 - val_accuracy: 0.6716 -
val_auc: 0.7416
Epoch 633/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6052 -
accuracy: 0.7187 - auc: 0.7934 - val_loss: 0.6576 - val_accuracy: 0.6753 -
val_auc: 0.7447
Epoch 634/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6050 -
accuracy: 0.7235 - auc: 0.7949 - val_loss: 0.6627 - val_accuracy: 0.6716 -
val_auc: 0.7389
Epoch 635/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6027 -
accuracy: 0.7203 - auc: 0.7965 - val_loss: 0.6568 - val_accuracy: 0.6741 -
val_auc: 0.7427
Epoch 636/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6056 -
accuracy: 0.7232 - auc: 0.7936 - val_loss: 0.6545 - val_accuracy: 0.6778 -
val_auc: 0.7458
```

```
Epoch 637/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6049 -
accuracy: 0.7203 - auc: 0.7945 - val_loss: 0.6546 - val_accuracy: 0.6778 -
val_auc: 0.7430
Epoch 638/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6022 -
accuracy: 0.7190 - auc: 0.7969 - val_loss: 0.6532 - val_accuracy: 0.6802 -
val_auc: 0.7460
Epoch 639/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6022 -
accuracy: 0.7227 - auc: 0.7960 - val_loss: 0.6528 - val_accuracy: 0.6753 -
val_auc: 0.7466
Epoch 640/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6017 -
accuracy: 0.7200 - auc: 0.7962 - val_loss: 0.6515 - val_accuracy: 0.6852 -
val_auc: 0.7492
Epoch 641/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6030 -
accuracy: 0.7208 - auc: 0.7953 - val_loss: 0.6507 - val_accuracy: 0.6790 -
val_auc: 0.7482
Epoch 642/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6020 -
accuracy: 0.7208 - auc: 0.7956 - val_loss: 0.6533 - val_accuracy: 0.6802 -
val_auc: 0.7475
Epoch 643/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6044 -
accuracy: 0.7192 - auc: 0.7933 - val_loss: 0.6510 - val_accuracy: 0.6840 -
val_auc: 0.7497
Epoch 644/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6051 -
accuracy: 0.7208 - auc: 0.7919 - val_loss: 0.6498 - val_accuracy: 0.6901 -
val_auc: 0.7501
Epoch 645/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6022 -
accuracy: 0.7198 - auc: 0.7950 - val_loss: 0.6506 - val_accuracy: 0.6889 -
val_auc: 0.7480
Epoch 646/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6028 -
accuracy: 0.7171 - auc: 0.7949 - val_loss: 0.6528 - val_accuracy: 0.6753 -
val_auc: 0.7449
Epoch 647/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6027 -
accuracy: 0.7206 - auc: 0.7949 - val_loss: 0.6513 - val_accuracy: 0.6840 -
val_auc: 0.7471
Epoch 648/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6036 -
accuracy: 0.7161 - auc: 0.7942 - val_loss: 0.6554 - val_accuracy: 0.6741 -
val_auc: 0.7419
```

```
Epoch 649/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6023 -
accuracy: 0.7211 - auc: 0.7963 - val_loss: 0.6517 - val_accuracy: 0.6802 -
val_auc: 0.7471
Epoch 650/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6025 -
accuracy: 0.7253 - auc: 0.7960 - val_loss: 0.6479 - val_accuracy: 0.6852 -
val_auc: 0.7509
Epoch 651/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6034 -
accuracy: 0.7166 - auc: 0.7933 - val_loss: 0.6533 - val_accuracy: 0.6765 -
val_auc: 0.7453
Epoch 652/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6001 -
accuracy: 0.7245 - auc: 0.7981 - val_loss: 0.6512 - val_accuracy: 0.6815 -
val_auc: 0.7490
Epoch 653/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6005 -
accuracy: 0.7195 - auc: 0.7969 - val_loss: 0.6493 - val_accuracy: 0.6827 -
val_auc: 0.7493
Epoch 654/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6005 -
accuracy: 0.7277 - auc: 0.7967 - val_loss: 0.6513 - val_accuracy: 0.6802 -
val_auc: 0.7476
Epoch 655/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6022 -
accuracy: 0.7211 - auc: 0.7949 - val_loss: 0.6519 - val_accuracy: 0.6815 -
val_auc: 0.7472
Epoch 656/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5999 -
accuracy: 0.7187 - auc: 0.7969 - val_loss: 0.6532 - val_accuracy: 0.6753 -
val_auc: 0.7441
Epoch 657/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6013 -
accuracy: 0.7211 - auc: 0.7949 - val_loss: 0.6536 - val_accuracy: 0.6728 -
val_auc: 0.7430
Epoch 658/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5998 -
accuracy: 0.7208 - auc: 0.7969 - val_loss: 0.6504 - val_accuracy: 0.6827 -
val_auc: 0.7476
Epoch 659/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5997 -
accuracy: 0.7145 - auc: 0.7956 - val_loss: 0.6495 - val_accuracy: 0.6802 -
val_auc: 0.7483
Epoch 660/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5989 -
accuracy: 0.7219 - auc: 0.7974 - val_loss: 0.6511 - val_accuracy: 0.6802 -
val_auc: 0.7456
```

```
Epoch 661/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6000 -
accuracy: 0.7224 - auc: 0.7969 - val_loss: 0.6524 - val_accuracy: 0.6802 -
val_auc: 0.7484
Epoch 662/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5956 -
accuracy: 0.7216 - auc: 0.8009 - val_loss: 0.6562 - val_accuracy: 0.6728 -
val_auc: 0.7415
Epoch 663/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6040 -
accuracy: 0.7200 - auc: 0.7923 - val_loss: 0.6540 - val_accuracy: 0.6840 -
val_auc: 0.7427
Epoch 664/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6021 -
accuracy: 0.7171 - auc: 0.7928 - val_loss: 0.6513 - val_accuracy: 0.6815 -
val_auc: 0.7449
Epoch 665/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5992 -
accuracy: 0.7227 - auc: 0.7967 - val_loss: 0.6507 - val_accuracy: 0.6704 -
val_auc: 0.7449
Epoch 666/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6000 -
accuracy: 0.7208 - auc: 0.7946 - val_loss: 0.6617 - val_accuracy: 0.6741 -
val_auc: 0.7336
Epoch 667/1000
119/119 [=====] - 2s 20ms/step - loss: 0.6000 -
accuracy: 0.7192 - auc: 0.7968 - val_loss: 0.6527 - val_accuracy: 0.6815 -
val_auc: 0.7467
Epoch 668/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5974 -
accuracy: 0.7240 - auc: 0.7985 - val_loss: 0.6512 - val_accuracy: 0.6778 -
val_auc: 0.7457
Epoch 669/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5987 -
accuracy: 0.7219 - auc: 0.7957 - val_loss: 0.6540 - val_accuracy: 0.6753 -
val_auc: 0.7420
Epoch 670/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5959 -
accuracy: 0.7237 - auc: 0.7987 - val_loss: 0.6491 - val_accuracy: 0.6802 -
val_auc: 0.7485
Epoch 671/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5991 -
accuracy: 0.7203 - auc: 0.7956 - val_loss: 0.6494 - val_accuracy: 0.6815 -
val_auc: 0.7461
Epoch 672/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5976 -
accuracy: 0.7251 - auc: 0.7969 - val_loss: 0.6473 - val_accuracy: 0.6864 -
val_auc: 0.7479
```

```
Epoch 673/1000
119/119 [=====] - 2s 21ms/step - loss: 0.6006 -
accuracy: 0.7208 - auc: 0.7936 - val_loss: 0.6522 - val_accuracy: 0.6802 -
val_auc: 0.7446
Epoch 674/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5979 -
accuracy: 0.7203 - auc: 0.7962 - val_loss: 0.6491 - val_accuracy: 0.6765 -
val_auc: 0.7463
Epoch 675/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5964 -
accuracy: 0.7253 - auc: 0.7987 - val_loss: 0.6521 - val_accuracy: 0.6778 -
val_auc: 0.7432
Epoch 676/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5972 -
accuracy: 0.7208 - auc: 0.7958 - val_loss: 0.6481 - val_accuracy: 0.6815 -
val_auc: 0.7473
Epoch 677/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5960 -
accuracy: 0.7240 - auc: 0.7975 - val_loss: 0.6471 - val_accuracy: 0.6827 -
val_auc: 0.7484
Epoch 678/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5974 -
accuracy: 0.7187 - auc: 0.7961 - val_loss: 0.6564 - val_accuracy: 0.6765 -
val_auc: 0.7391
Epoch 679/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5965 -
accuracy: 0.7227 - auc: 0.7971 - val_loss: 0.6525 - val_accuracy: 0.6728 -
val_auc: 0.7424
Epoch 680/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5991 -
accuracy: 0.7214 - auc: 0.7937 - val_loss: 0.6500 - val_accuracy: 0.6778 -
val_auc: 0.7470
Epoch 681/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5948 -
accuracy: 0.7259 - auc: 0.7997 - val_loss: 0.6525 - val_accuracy: 0.6778 -
val_auc: 0.7446
Epoch 682/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5964 -
accuracy: 0.7227 - auc: 0.7976 - val_loss: 0.6463 - val_accuracy: 0.6852 -
val_auc: 0.7494
Epoch 683/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5959 -
accuracy: 0.7206 - auc: 0.7977 - val_loss: 0.6526 - val_accuracy: 0.6765 -
val_auc: 0.7415
Epoch 684/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5986 -
accuracy: 0.7195 - auc: 0.7951 - val_loss: 0.6503 - val_accuracy: 0.6778 -
val_auc: 0.7435
```

```
Epoch 685/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5960 -
accuracy: 0.7206 - auc: 0.7976 - val_loss: 0.6483 - val_accuracy: 0.6765 -
val_auc: 0.7452
Epoch 686/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5957 -
accuracy: 0.7221 - auc: 0.7976 - val_loss: 0.6499 - val_accuracy: 0.6778 -
val_auc: 0.7475
Epoch 687/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5946 -
accuracy: 0.7216 - auc: 0.7982 - val_loss: 0.6505 - val_accuracy: 0.6753 -
val_auc: 0.7451
Epoch 688/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5963 -
accuracy: 0.7211 - auc: 0.7968 - val_loss: 0.6485 - val_accuracy: 0.6765 -
val_auc: 0.7460
Epoch 689/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5953 -
accuracy: 0.7235 - auc: 0.7971 - val_loss: 0.6484 - val_accuracy: 0.6778 -
val_auc: 0.7467
Epoch 690/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5980 -
accuracy: 0.7221 - auc: 0.7955 - val_loss: 0.6476 - val_accuracy: 0.6802 -
val_auc: 0.7451
Epoch 691/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5972 -
accuracy: 0.7192 - auc: 0.7954 - val_loss: 0.6452 - val_accuracy: 0.6815 -
val_auc: 0.7478
Epoch 692/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5970 -
accuracy: 0.7237 - auc: 0.7952 - val_loss: 0.6483 - val_accuracy: 0.6765 -
val_auc: 0.7454
Epoch 693/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5961 -
accuracy: 0.7166 - auc: 0.7967 - val_loss: 0.6489 - val_accuracy: 0.6728 -
val_auc: 0.7460
Epoch 694/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5948 -
accuracy: 0.7227 - auc: 0.7980 - val_loss: 0.6473 - val_accuracy: 0.6790 -
val_auc: 0.7468
Epoch 695/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5953 -
accuracy: 0.7245 - auc: 0.7981 - val_loss: 0.6475 - val_accuracy: 0.6815 -
val_auc: 0.7470
Epoch 696/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5924 -
accuracy: 0.7274 - auc: 0.7993 - val_loss: 0.6427 - val_accuracy: 0.6901 -
val_auc: 0.7516
```

```
Epoch 697/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5981 -
accuracy: 0.7190 - auc: 0.7937 - val_loss: 0.6467 - val_accuracy: 0.6778 -
val_auc: 0.7475
Epoch 698/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5976 -
accuracy: 0.7229 - auc: 0.7938 - val_loss: 0.6442 - val_accuracy: 0.6889 -
val_auc: 0.7492
Epoch 699/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5943 -
accuracy: 0.7208 - auc: 0.7987 - val_loss: 0.6465 - val_accuracy: 0.6728 -
val_auc: 0.7478
Epoch 700/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5951 -
accuracy: 0.7259 - auc: 0.7979 - val_loss: 0.6442 - val_accuracy: 0.6852 -
val_auc: 0.7502
Epoch 701/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5955 -
accuracy: 0.7190 - auc: 0.7977 - val_loss: 0.6462 - val_accuracy: 0.6815 -
val_auc: 0.7463
Epoch 702/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5953 -
accuracy: 0.7248 - auc: 0.7969 - val_loss: 0.6478 - val_accuracy: 0.6802 -
val_auc: 0.7464
Epoch 703/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5950 -
accuracy: 0.7224 - auc: 0.7980 - val_loss: 0.6417 - val_accuracy: 0.6802 -
val_auc: 0.7515
Epoch 704/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5951 -
accuracy: 0.7155 - auc: 0.7955 - val_loss: 0.6497 - val_accuracy: 0.6753 -
val_auc: 0.7442
Epoch 705/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5942 -
accuracy: 0.7232 - auc: 0.7978 - val_loss: 0.6414 - val_accuracy: 0.6852 -
val_auc: 0.7512
Epoch 706/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5973 -
accuracy: 0.7171 - auc: 0.7941 - val_loss: 0.6501 - val_accuracy: 0.6765 -
val_auc: 0.7431
Epoch 707/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5912 -
accuracy: 0.7237 - auc: 0.8002 - val_loss: 0.6582 - val_accuracy: 0.6753 -
val_auc: 0.7341
Epoch 708/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5951 -
accuracy: 0.7211 - auc: 0.7955 - val_loss: 0.6479 - val_accuracy: 0.6753 -
val_auc: 0.7453
```

```
Epoch 709/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5904 -
accuracy: 0.7269 - auc: 0.8015 - val_loss: 0.6533 - val_accuracy: 0.6765 -
val_auc: 0.7380
Epoch 710/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5947 -
accuracy: 0.7179 - auc: 0.7956 - val_loss: 0.6467 - val_accuracy: 0.6778 -
val_auc: 0.7471
Epoch 711/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5939 -
accuracy: 0.7224 - auc: 0.7962 - val_loss: 0.6552 - val_accuracy: 0.6753 -
val_auc: 0.7362
Epoch 712/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5926 -
accuracy: 0.7211 - auc: 0.7982 - val_loss: 0.6477 - val_accuracy: 0.6765 -
val_auc: 0.7450
Epoch 713/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5912 -
accuracy: 0.7248 - auc: 0.8004 - val_loss: 0.6483 - val_accuracy: 0.6802 -
val_auc: 0.7469
Epoch 714/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5919 -
accuracy: 0.7224 - auc: 0.7987 - val_loss: 0.6453 - val_accuracy: 0.6864 -
val_auc: 0.7494
Epoch 715/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5916 -
accuracy: 0.7229 - auc: 0.7988 - val_loss: 0.6510 - val_accuracy: 0.6802 -
val_auc: 0.7434
Epoch 716/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5939 -
accuracy: 0.7216 - auc: 0.7966 - val_loss: 0.6455 - val_accuracy: 0.6889 -
val_auc: 0.7489
Epoch 717/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5919 -
accuracy: 0.7272 - auc: 0.7982 - val_loss: 0.6445 - val_accuracy: 0.6864 -
val_auc: 0.7492
Epoch 718/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5941 -
accuracy: 0.7277 - auc: 0.7963 - val_loss: 0.6463 - val_accuracy: 0.6778 -
val_auc: 0.7460
Epoch 719/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5925 -
accuracy: 0.7187 - auc: 0.7980 - val_loss: 0.6442 - val_accuracy: 0.6840 -
val_auc: 0.7494
Epoch 720/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5938 -
accuracy: 0.7235 - auc: 0.7966 - val_loss: 0.6445 - val_accuracy: 0.6840 -
val_auc: 0.7492
```

```
Epoch 721/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5956 -
accuracy: 0.7206 - auc: 0.7944 - val_loss: 0.6431 - val_accuracy: 0.6889 -
val_auc: 0.7497
Epoch 722/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5940 -
accuracy: 0.7182 - auc: 0.7959 - val_loss: 0.6422 - val_accuracy: 0.6889 -
val_auc: 0.7512
Epoch 723/1000
119/119 [=====] - 3s 22ms/step - loss: 0.5923 -
accuracy: 0.7269 - auc: 0.7974 - val_loss: 0.6397 - val_accuracy: 0.6877 -
val_auc: 0.7523
Epoch 724/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5903 -
accuracy: 0.7214 - auc: 0.7997 - val_loss: 0.6423 - val_accuracy: 0.6840 -
val_auc: 0.7492
Epoch 725/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5891 -
accuracy: 0.7288 - auc: 0.8006 - val_loss: 0.6454 - val_accuracy: 0.6840 -
val_auc: 0.7475
Epoch 726/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5917 -
accuracy: 0.7277 - auc: 0.7989 - val_loss: 0.6418 - val_accuracy: 0.6852 -
val_auc: 0.7490
Epoch 727/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5941 -
accuracy: 0.7184 - auc: 0.7941 - val_loss: 0.6435 - val_accuracy: 0.6827 -
val_auc: 0.7488
Epoch 728/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5901 -
accuracy: 0.7192 - auc: 0.7993 - val_loss: 0.6449 - val_accuracy: 0.6852 -
val_auc: 0.7478
Epoch 729/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5919 -
accuracy: 0.7211 - auc: 0.7959 - val_loss: 0.6452 - val_accuracy: 0.6790 -
val_auc: 0.7460
Epoch 730/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5902 -
accuracy: 0.7227 - auc: 0.7988 - val_loss: 0.6464 - val_accuracy: 0.6753 -
val_auc: 0.7447
Epoch 731/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5917 -
accuracy: 0.7227 - auc: 0.7971 - val_loss: 0.6400 - val_accuracy: 0.6852 -
val_auc: 0.7518
Epoch 732/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5896 -
accuracy: 0.7235 - auc: 0.7984 - val_loss: 0.6433 - val_accuracy: 0.6802 -
val_auc: 0.7471
```

```
Epoch 733/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5942 -
accuracy: 0.7237 - auc: 0.7935 - val_loss: 0.6500 - val_accuracy: 0.6778 -
val_auc: 0.7393
Epoch 734/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5901 -
accuracy: 0.7187 - auc: 0.7983 - val_loss: 0.6434 - val_accuracy: 0.6827 -
val_auc: 0.7485
Epoch 735/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5886 -
accuracy: 0.7203 - auc: 0.8002 - val_loss: 0.6456 - val_accuracy: 0.6790 -
val_auc: 0.7457
Epoch 736/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5917 -
accuracy: 0.7232 - auc: 0.7982 - val_loss: 0.6418 - val_accuracy: 0.6864 -
val_auc: 0.7491
Epoch 737/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5892 -
accuracy: 0.7259 - auc: 0.7992 - val_loss: 0.6426 - val_accuracy: 0.6790 -
val_auc: 0.7482
Epoch 738/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5889 -
accuracy: 0.7235 - auc: 0.8002 - val_loss: 0.6433 - val_accuracy: 0.6765 -
val_auc: 0.7472
Epoch 739/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5881 -
accuracy: 0.7290 - auc: 0.7998 - val_loss: 0.6579 - val_accuracy: 0.6716 -
val_auc: 0.7327
Epoch 740/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5895 -
accuracy: 0.7227 - auc: 0.7978 - val_loss: 0.6445 - val_accuracy: 0.6765 -
val_auc: 0.7464
Epoch 741/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5877 -
accuracy: 0.7280 - auc: 0.8006 - val_loss: 0.6376 - val_accuracy: 0.6926 -
val_auc: 0.7543
Epoch 742/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5895 -
accuracy: 0.7227 - auc: 0.7970 - val_loss: 0.6426 - val_accuracy: 0.6827 -
val_auc: 0.7473
Epoch 743/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5890 -
accuracy: 0.7227 - auc: 0.7992 - val_loss: 0.6403 - val_accuracy: 0.6864 -
val_auc: 0.7491
Epoch 744/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5854 -
accuracy: 0.7259 - auc: 0.8006 - val_loss: 0.6479 - val_accuracy: 0.6741 -
val_auc: 0.7439
```

```
Epoch 745/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5915 -
accuracy: 0.7195 - auc: 0.7962 - val_loss: 0.6400 - val_accuracy: 0.6864 -
val_auc: 0.7504
Epoch 746/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5885 -
accuracy: 0.7243 - auc: 0.7998 - val_loss: 0.6419 - val_accuracy: 0.6827 -
val_auc: 0.7487
Epoch 747/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5871 -
accuracy: 0.7282 - auc: 0.8006 - val_loss: 0.6425 - val_accuracy: 0.6815 -
val_auc: 0.7469
Epoch 748/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5869 -
accuracy: 0.7282 - auc: 0.8012 - val_loss: 0.6393 - val_accuracy: 0.6852 -
val_auc: 0.7510
Epoch 749/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5907 -
accuracy: 0.7261 - auc: 0.7974 - val_loss: 0.6412 - val_accuracy: 0.6864 -
val_auc: 0.7499
Epoch 750/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5915 -
accuracy: 0.7203 - auc: 0.7957 - val_loss: 0.6409 - val_accuracy: 0.6852 -
val_auc: 0.7492
Epoch 751/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5911 -
accuracy: 0.7206 - auc: 0.7959 - val_loss: 0.6373 - val_accuracy: 0.6901 -
val_auc: 0.7536
Epoch 752/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5890 -
accuracy: 0.7274 - auc: 0.7980 - val_loss: 0.6432 - val_accuracy: 0.6864 -
val_auc: 0.7478
Epoch 753/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5920 -
accuracy: 0.7200 - auc: 0.7968 - val_loss: 0.6414 - val_accuracy: 0.6852 -
val_auc: 0.7492
Epoch 754/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5862 -
accuracy: 0.7237 - auc: 0.8006 - val_loss: 0.6406 - val_accuracy: 0.6901 -
val_auc: 0.7494
Epoch 755/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5866 -
accuracy: 0.7227 - auc: 0.8006 - val_loss: 0.6364 - val_accuracy: 0.6901 -
val_auc: 0.7543
Epoch 756/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5892 -
accuracy: 0.7251 - auc: 0.7976 - val_loss: 0.6419 - val_accuracy: 0.6778 -
val_auc: 0.7459
```

```
Epoch 757/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5911 -
accuracy: 0.7227 - auc: 0.7971 - val_loss: 0.6403 - val_accuracy: 0.6877 -
val_auc: 0.7495
Epoch 758/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5896 -
accuracy: 0.7261 - auc: 0.7988 - val_loss: 0.6401 - val_accuracy: 0.6852 -
val_auc: 0.7504
Epoch 759/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5886 -
accuracy: 0.7206 - auc: 0.7989 - val_loss: 0.6415 - val_accuracy: 0.6840 -
val_auc: 0.7504
Epoch 760/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5888 -
accuracy: 0.7224 - auc: 0.7981 - val_loss: 0.6408 - val_accuracy: 0.6864 -
val_auc: 0.7490
Epoch 761/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5842 -
accuracy: 0.7317 - auc: 0.8036 - val_loss: 0.6394 - val_accuracy: 0.6864 -
val_auc: 0.7516
Epoch 762/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5893 -
accuracy: 0.7285 - auc: 0.7982 - val_loss: 0.6399 - val_accuracy: 0.6901 -
val_auc: 0.7507
Epoch 763/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5896 -
accuracy: 0.7216 - auc: 0.7971 - val_loss: 0.6415 - val_accuracy: 0.6827 -
val_auc: 0.7462
Epoch 764/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5868 -
accuracy: 0.7266 - auc: 0.7995 - val_loss: 0.6415 - val_accuracy: 0.6815 -
val_auc: 0.7468
Epoch 765/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5878 -
accuracy: 0.7240 - auc: 0.7990 - val_loss: 0.6425 - val_accuracy: 0.6790 -
val_auc: 0.7459
Epoch 766/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5845 -
accuracy: 0.7251 - auc: 0.8020 - val_loss: 0.6386 - val_accuracy: 0.6901 -
val_auc: 0.7501
Epoch 767/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5867 -
accuracy: 0.7269 - auc: 0.7997 - val_loss: 0.6370 - val_accuracy: 0.6889 -
val_auc: 0.7529
Epoch 768/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5871 -
accuracy: 0.7214 - auc: 0.7989 - val_loss: 0.6410 - val_accuracy: 0.6840 -
val_auc: 0.7481
```

```
Epoch 769/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5851 -
accuracy: 0.7253 - auc: 0.8011 - val_loss: 0.6413 - val_accuracy: 0.6790 -
val_auc: 0.7461
Epoch 770/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5883 -
accuracy: 0.7256 - auc: 0.7984 - val_loss: 0.6450 - val_accuracy: 0.6765 -
val_auc: 0.7449
Epoch 771/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5863 -
accuracy: 0.7253 - auc: 0.8004 - val_loss: 0.6384 - val_accuracy: 0.6802 -
val_auc: 0.7495
Epoch 772/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5879 -
accuracy: 0.7221 - auc: 0.7977 - val_loss: 0.6468 - val_accuracy: 0.6790 -
val_auc: 0.7403
Epoch 773/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5855 -
accuracy: 0.7256 - auc: 0.8004 - val_loss: 0.6375 - val_accuracy: 0.6926 -
val_auc: 0.7504
Epoch 774/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5887 -
accuracy: 0.7182 - auc: 0.7969 - val_loss: 0.6381 - val_accuracy: 0.6815 -
val_auc: 0.7513
Epoch 775/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5874 -
accuracy: 0.7266 - auc: 0.7990 - val_loss: 0.6383 - val_accuracy: 0.6827 -
val_auc: 0.7504
Epoch 776/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5856 -
accuracy: 0.7221 - auc: 0.7990 - val_loss: 0.6510 - val_accuracy: 0.6790 -
val_auc: 0.7359
Epoch 777/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5859 -
accuracy: 0.7256 - auc: 0.8020 - val_loss: 0.6399 - val_accuracy: 0.6827 -
val_auc: 0.7477
Epoch 778/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5876 -
accuracy: 0.7224 - auc: 0.7990 - val_loss: 0.6414 - val_accuracy: 0.6815 -
val_auc: 0.7458
Epoch 779/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5844 -
accuracy: 0.7269 - auc: 0.8005 - val_loss: 0.6381 - val_accuracy: 0.6864 -
val_auc: 0.7496
Epoch 780/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5852 -
accuracy: 0.7259 - auc: 0.8007 - val_loss: 0.6419 - val_accuracy: 0.6815 -
val_auc: 0.7450
```

```
Epoch 781/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5840 -
accuracy: 0.7304 - auc: 0.8020 - val_loss: 0.6344 - val_accuracy: 0.6901 -
val_auc: 0.7530
Epoch 782/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5880 -
accuracy: 0.7184 - auc: 0.7962 - val_loss: 0.6424 - val_accuracy: 0.6790 -
val_auc: 0.7462
Epoch 783/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5867 -
accuracy: 0.7216 - auc: 0.7979 - val_loss: 0.6367 - val_accuracy: 0.6864 -
val_auc: 0.7505
Epoch 784/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5856 -
accuracy: 0.7306 - auc: 0.8008 - val_loss: 0.6429 - val_accuracy: 0.6753 -
val_auc: 0.7443
Epoch 785/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5860 -
accuracy: 0.7274 - auc: 0.8007 - val_loss: 0.6368 - val_accuracy: 0.6852 -
val_auc: 0.7502
Epoch 786/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5878 -
accuracy: 0.7245 - auc: 0.7984 - val_loss: 0.6347 - val_accuracy: 0.6877 -
val_auc: 0.7520
Epoch 787/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5881 -
accuracy: 0.7237 - auc: 0.7972 - val_loss: 0.6414 - val_accuracy: 0.6815 -
val_auc: 0.7449
Epoch 788/1000
119/119 [=====] - 3s 22ms/step - loss: 0.5848 -
accuracy: 0.7264 - auc: 0.8007 - val_loss: 0.6364 - val_accuracy: 0.6877 -
val_auc: 0.7499
Epoch 789/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5835 -
accuracy: 0.7325 - auc: 0.8034 - val_loss: 0.6363 - val_accuracy: 0.6840 -
val_auc: 0.7494
Epoch 790/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5893 -
accuracy: 0.7214 - auc: 0.7945 - val_loss: 0.6426 - val_accuracy: 0.6802 -
val_auc: 0.7437
Epoch 791/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5838 -
accuracy: 0.7261 - auc: 0.8008 - val_loss: 0.6360 - val_accuracy: 0.6889 -
val_auc: 0.7522
Epoch 792/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5856 -
accuracy: 0.7272 - auc: 0.7995 - val_loss: 0.6396 - val_accuracy: 0.6864 -
val_auc: 0.7478
```

```
Epoch 793/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5845 -
accuracy: 0.7280 - auc: 0.8003 - val_loss: 0.6336 - val_accuracy: 0.6901 -
val_auc: 0.7545
Epoch 794/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5865 -
accuracy: 0.7290 - auc: 0.7981 - val_loss: 0.6374 - val_accuracy: 0.6901 -
val_auc: 0.7509
Epoch 795/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5836 -
accuracy: 0.7235 - auc: 0.8006 - val_loss: 0.6450 - val_accuracy: 0.6815 -
val_auc: 0.7406
Epoch 796/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5839 -
accuracy: 0.7235 - auc: 0.8006 - val_loss: 0.6380 - val_accuracy: 0.6852 -
val_auc: 0.7482
Epoch 797/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5834 -
accuracy: 0.7261 - auc: 0.8018 - val_loss: 0.6349 - val_accuracy: 0.6827 -
val_auc: 0.7520
Epoch 798/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5862 -
accuracy: 0.7229 - auc: 0.7981 - val_loss: 0.6331 - val_accuracy: 0.6889 -
val_auc: 0.7537
Epoch 799/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5814 -
accuracy: 0.7282 - auc: 0.8029 - val_loss: 0.6414 - val_accuracy: 0.6765 -
val_auc: 0.7465
Epoch 800/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5845 -
accuracy: 0.7264 - auc: 0.7987 - val_loss: 0.6358 - val_accuracy: 0.6889 -
val_auc: 0.7524
Epoch 801/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5804 -
accuracy: 0.7261 - auc: 0.8036 - val_loss: 0.6365 - val_accuracy: 0.6889 -
val_auc: 0.7517
Epoch 802/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5850 -
accuracy: 0.7243 - auc: 0.7979 - val_loss: 0.6317 - val_accuracy: 0.6889 -
val_auc: 0.7544
Epoch 803/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5846 -
accuracy: 0.7221 - auc: 0.7987 - val_loss: 0.6381 - val_accuracy: 0.6852 -
val_auc: 0.7475
Epoch 804/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5821 -
accuracy: 0.7304 - auc: 0.8016 - val_loss: 0.6359 - val_accuracy: 0.6901 -
val_auc: 0.7503
```

```
Epoch 805/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5826 -
accuracy: 0.7251 - auc: 0.8010 - val_loss: 0.6406 - val_accuracy: 0.6753 -
val_auc: 0.7445
Epoch 806/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5850 -
accuracy: 0.7214 - auc: 0.7976 - val_loss: 0.6411 - val_accuracy: 0.6802 -
val_auc: 0.7464
Epoch 807/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5820 -
accuracy: 0.7248 - auc: 0.8024 - val_loss: 0.6336 - val_accuracy: 0.6889 -
val_auc: 0.7521
Epoch 808/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5829 -
accuracy: 0.7261 - auc: 0.8004 - val_loss: 0.6347 - val_accuracy: 0.6852 -
val_auc: 0.7523
Epoch 809/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5849 -
accuracy: 0.7272 - auc: 0.7979 - val_loss: 0.6423 - val_accuracy: 0.6765 -
val_auc: 0.7462
Epoch 810/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5847 -
accuracy: 0.7288 - auc: 0.7988 - val_loss: 0.6354 - val_accuracy: 0.6889 -
val_auc: 0.7500
Epoch 811/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5818 -
accuracy: 0.7306 - auc: 0.8021 - val_loss: 0.6397 - val_accuracy: 0.6778 -
val_auc: 0.7457
Epoch 812/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5829 -
accuracy: 0.7288 - auc: 0.8020 - val_loss: 0.6347 - val_accuracy: 0.6889 -
val_auc: 0.7524
Epoch 813/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5836 -
accuracy: 0.7282 - auc: 0.8003 - val_loss: 0.6325 - val_accuracy: 0.6889 -
val_auc: 0.7538
Epoch 814/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5837 -
accuracy: 0.7272 - auc: 0.7993 - val_loss: 0.6383 - val_accuracy: 0.6840 -
val_auc: 0.7481
Epoch 815/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5849 -
accuracy: 0.7243 - auc: 0.7981 - val_loss: 0.6475 - val_accuracy: 0.6765 -
val_auc: 0.7404
Epoch 816/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5855 -
accuracy: 0.7256 - auc: 0.7981 - val_loss: 0.6371 - val_accuracy: 0.6889 -
val_auc: 0.7499
```

```
Epoch 817/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5833 -
accuracy: 0.7240 - auc: 0.8011 - val_loss: 0.6340 - val_accuracy: 0.6901 -
val_auc: 0.7514
Epoch 818/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5839 -
accuracy: 0.7266 - auc: 0.8004 - val_loss: 0.6352 - val_accuracy: 0.6914 -
val_auc: 0.7501
Epoch 819/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5790 -
accuracy: 0.7325 - auc: 0.8053 - val_loss: 0.6361 - val_accuracy: 0.6877 -
val_auc: 0.7496
Epoch 820/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5833 -
accuracy: 0.7235 - auc: 0.7989 - val_loss: 0.6373 - val_accuracy: 0.6864 -
val_auc: 0.7483
Epoch 821/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5850 -
accuracy: 0.7264 - auc: 0.7976 - val_loss: 0.6391 - val_accuracy: 0.6778 -
val_auc: 0.7478
Epoch 822/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5828 -
accuracy: 0.7301 - auc: 0.7998 - val_loss: 0.6338 - val_accuracy: 0.6877 -
val_auc: 0.7510
Epoch 823/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5834 -
accuracy: 0.7224 - auc: 0.7990 - val_loss: 0.6362 - val_accuracy: 0.6901 -
val_auc: 0.7481
Epoch 824/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5837 -
accuracy: 0.7171 - auc: 0.7979 - val_loss: 0.6414 - val_accuracy: 0.6778 -
val_auc: 0.7442
Epoch 825/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5798 -
accuracy: 0.7274 - auc: 0.8039 - val_loss: 0.6398 - val_accuracy: 0.6827 -
val_auc: 0.7471
Epoch 826/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5829 -
accuracy: 0.7253 - auc: 0.7993 - val_loss: 0.6336 - val_accuracy: 0.6901 -
val_auc: 0.7502
Epoch 827/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5782 -
accuracy: 0.7272 - auc: 0.8050 - val_loss: 0.6343 - val_accuracy: 0.6901 -
val_auc: 0.7517
Epoch 828/1000
119/119 [=====] - 3s 22ms/step - loss: 0.5808 -
accuracy: 0.7206 - auc: 0.8015 - val_loss: 0.6381 - val_accuracy: 0.6877 -
val_auc: 0.7481
```

```
Epoch 829/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5836 -
accuracy: 0.7272 - auc: 0.7992 - val_loss: 0.6371 - val_accuracy: 0.6852 -
val_auc: 0.7482
Epoch 830/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5805 -
accuracy: 0.7264 - auc: 0.8026 - val_loss: 0.6330 - val_accuracy: 0.6889 -
val_auc: 0.7514
Epoch 831/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5835 -
accuracy: 0.7306 - auc: 0.7999 - val_loss: 0.6363 - val_accuracy: 0.6852 -
val_auc: 0.7478
Epoch 832/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5833 -
accuracy: 0.7253 - auc: 0.8002 - val_loss: 0.6416 - val_accuracy: 0.6864 -
val_auc: 0.7449
Epoch 833/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5805 -
accuracy: 0.7251 - auc: 0.8008 - val_loss: 0.6383 - val_accuracy: 0.6815 -
val_auc: 0.7466
Epoch 834/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5795 -
accuracy: 0.7229 - auc: 0.8036 - val_loss: 0.6314 - val_accuracy: 0.6901 -
val_auc: 0.7539
Epoch 835/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5846 -
accuracy: 0.7245 - auc: 0.7991 - val_loss: 0.6339 - val_accuracy: 0.6864 -
val_auc: 0.7512
Epoch 836/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5823 -
accuracy: 0.7269 - auc: 0.8005 - val_loss: 0.6381 - val_accuracy: 0.6815 -
val_auc: 0.7459
Epoch 837/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5792 -
accuracy: 0.7296 - auc: 0.8033 - val_loss: 0.6334 - val_accuracy: 0.6889 -
val_auc: 0.7520
Epoch 838/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5813 -
accuracy: 0.7251 - auc: 0.7998 - val_loss: 0.6338 - val_accuracy: 0.6889 -
val_auc: 0.7495
Epoch 839/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5817 -
accuracy: 0.7282 - auc: 0.7998 - val_loss: 0.6336 - val_accuracy: 0.6840 -
val_auc: 0.7504
Epoch 840/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5807 -
accuracy: 0.7266 - auc: 0.8017 - val_loss: 0.6333 - val_accuracy: 0.6877 -
val_auc: 0.7518
```

```
Epoch 841/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5783 -
accuracy: 0.7248 - auc: 0.8014 - val_loss: 0.6350 - val_accuracy: 0.6889 -
val_auc: 0.7518
Epoch 842/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5817 -
accuracy: 0.7203 - auc: 0.7995 - val_loss: 0.6367 - val_accuracy: 0.6852 -
val_auc: 0.7479
Epoch 843/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5810 -
accuracy: 0.7266 - auc: 0.8022 - val_loss: 0.6309 - val_accuracy: 0.6852 -
val_auc: 0.7540
Epoch 844/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5813 -
accuracy: 0.7282 - auc: 0.8004 - val_loss: 0.6363 - val_accuracy: 0.6926 -
val_auc: 0.7500
Epoch 845/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5802 -
accuracy: 0.7243 - auc: 0.8028 - val_loss: 0.6308 - val_accuracy: 0.6877 -
val_auc: 0.7543
Epoch 846/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5814 -
accuracy: 0.7240 - auc: 0.8007 - val_loss: 0.6313 - val_accuracy: 0.6901 -
val_auc: 0.7533
Epoch 847/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5816 -
accuracy: 0.7282 - auc: 0.8007 - val_loss: 0.6333 - val_accuracy: 0.6852 -
val_auc: 0.7514
Epoch 848/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5807 -
accuracy: 0.7264 - auc: 0.8014 - val_loss: 0.6325 - val_accuracy: 0.6926 -
val_auc: 0.7521
Epoch 849/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5805 -
accuracy: 0.7203 - auc: 0.8022 - val_loss: 0.6387 - val_accuracy: 0.6765 -
val_auc: 0.7455
Epoch 850/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5792 -
accuracy: 0.7229 - auc: 0.8024 - val_loss: 0.6352 - val_accuracy: 0.6901 -
val_auc: 0.7502
Epoch 851/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5803 -
accuracy: 0.7256 - auc: 0.8014 - val_loss: 0.6304 - val_accuracy: 0.6914 -
val_auc: 0.7539
Epoch 852/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5814 -
accuracy: 0.7248 - auc: 0.8006 - val_loss: 0.6330 - val_accuracy: 0.6889 -
val_auc: 0.7511
```

```
Epoch 853/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5824 -
accuracy: 0.7266 - auc: 0.7997 - val_loss: 0.6337 - val_accuracy: 0.6914 -
val_auc: 0.7516
Epoch 854/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5800 -
accuracy: 0.7309 - auc: 0.8019 - val_loss: 0.6379 - val_accuracy: 0.6852 -
val_auc: 0.7451
Epoch 855/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5792 -
accuracy: 0.7264 - auc: 0.8017 - val_loss: 0.6396 - val_accuracy: 0.6815 -
val_auc: 0.7452
Epoch 856/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5781 -
accuracy: 0.7293 - auc: 0.8028 - val_loss: 0.6348 - val_accuracy: 0.6864 -
val_auc: 0.7490
Epoch 857/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5774 -
accuracy: 0.7304 - auc: 0.8048 - val_loss: 0.6356 - val_accuracy: 0.6864 -
val_auc: 0.7502
Epoch 858/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5774 -
accuracy: 0.7285 - auc: 0.8035 - val_loss: 0.6322 - val_accuracy: 0.6938 -
val_auc: 0.7533
Epoch 859/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5786 -
accuracy: 0.7266 - auc: 0.8021 - val_loss: 0.6304 - val_accuracy: 0.6864 -
val_auc: 0.7556
Epoch 860/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5801 -
accuracy: 0.7259 - auc: 0.8000 - val_loss: 0.6365 - val_accuracy: 0.6926 -
val_auc: 0.7484
Epoch 861/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5793 -
accuracy: 0.7301 - auc: 0.8011 - val_loss: 0.6361 - val_accuracy: 0.6852 -
val_auc: 0.7483
Epoch 862/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5807 -
accuracy: 0.7216 - auc: 0.7993 - val_loss: 0.6341 - val_accuracy: 0.6914 -
val_auc: 0.7511
Epoch 863/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5787 -
accuracy: 0.7272 - auc: 0.8015 - val_loss: 0.6352 - val_accuracy: 0.6901 -
val_auc: 0.7492
Epoch 864/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5800 -
accuracy: 0.7309 - auc: 0.8011 - val_loss: 0.6384 - val_accuracy: 0.6827 -
val_auc: 0.7457
```

```
Epoch 865/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5802 -
accuracy: 0.7232 - auc: 0.8016 - val_loss: 0.6333 - val_accuracy: 0.6963 -
val_auc: 0.7516
Epoch 866/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5768 -
accuracy: 0.7274 - auc: 0.8050 - val_loss: 0.6350 - val_accuracy: 0.6889 -
val_auc: 0.7492
Epoch 867/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5776 -
accuracy: 0.7277 - auc: 0.8029 - val_loss: 0.6355 - val_accuracy: 0.6840 -
val_auc: 0.7483
Epoch 868/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5818 -
accuracy: 0.7245 - auc: 0.7985 - val_loss: 0.6344 - val_accuracy: 0.6938 -
val_auc: 0.7508
Epoch 869/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5801 -
accuracy: 0.7274 - auc: 0.8020 - val_loss: 0.6349 - val_accuracy: 0.6914 -
val_auc: 0.7517
Epoch 870/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5782 -
accuracy: 0.7282 - auc: 0.8022 - val_loss: 0.6342 - val_accuracy: 0.6938 -
val_auc: 0.7516
Epoch 871/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5766 -
accuracy: 0.7274 - auc: 0.8038 - val_loss: 0.6360 - val_accuracy: 0.6852 -
val_auc: 0.7471
Epoch 872/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5786 -
accuracy: 0.7304 - auc: 0.8027 - val_loss: 0.6391 - val_accuracy: 0.6778 -
val_auc: 0.7460
Epoch 873/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5781 -
accuracy: 0.7211 - auc: 0.8015 - val_loss: 0.6326 - val_accuracy: 0.6938 -
val_auc: 0.7515
Epoch 874/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5754 -
accuracy: 0.7274 - auc: 0.8045 - val_loss: 0.6311 - val_accuracy: 0.6951 -
val_auc: 0.7517
Epoch 875/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5748 -
accuracy: 0.7317 - auc: 0.8062 - val_loss: 0.6366 - val_accuracy: 0.6864 -
val_auc: 0.7476
Epoch 876/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5792 -
accuracy: 0.7240 - auc: 0.8010 - val_loss: 0.6307 - val_accuracy: 0.6914 -
val_auc: 0.7518
```

```
Epoch 877/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5783 -
accuracy: 0.7256 - auc: 0.8025 - val_loss: 0.6323 - val_accuracy: 0.6901 -
val_auc: 0.7525
Epoch 878/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5763 -
accuracy: 0.7251 - auc: 0.8042 - val_loss: 0.6321 - val_accuracy: 0.6889 -
val_auc: 0.7516
Epoch 879/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5783 -
accuracy: 0.7256 - auc: 0.8013 - val_loss: 0.6305 - val_accuracy: 0.6926 -
val_auc: 0.7524
Epoch 880/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5832 -
accuracy: 0.7190 - auc: 0.7963 - val_loss: 0.6337 - val_accuracy: 0.6901 -
val_auc: 0.7510
Epoch 881/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5796 -
accuracy: 0.7248 - auc: 0.8013 - val_loss: 0.6370 - val_accuracy: 0.6778 -
val_auc: 0.7467
Epoch 882/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5746 -
accuracy: 0.7322 - auc: 0.8053 - val_loss: 0.6347 - val_accuracy: 0.6815 -
val_auc: 0.7503
Epoch 883/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5773 -
accuracy: 0.7327 - auc: 0.8027 - val_loss: 0.6254 - val_accuracy: 0.6938 -
val_auc: 0.7571
Epoch 884/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5771 -
accuracy: 0.7304 - auc: 0.8030 - val_loss: 0.6299 - val_accuracy: 0.6877 -
val_auc: 0.7542
Epoch 885/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5779 -
accuracy: 0.7277 - auc: 0.8015 - val_loss: 0.6361 - val_accuracy: 0.6815 -
val_auc: 0.7473
Epoch 886/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5789 -
accuracy: 0.7253 - auc: 0.8009 - val_loss: 0.6342 - val_accuracy: 0.6901 -
val_auc: 0.7484
Epoch 887/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5768 -
accuracy: 0.7296 - auc: 0.8016 - val_loss: 0.6400 - val_accuracy: 0.6802 -
val_auc: 0.7422
Epoch 888/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5783 -
accuracy: 0.7243 - auc: 0.8013 - val_loss: 0.6300 - val_accuracy: 0.6938 -
val_auc: 0.7533
```

Epoch 889/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5778 -
accuracy: 0.7272 - auc: 0.8019 - val_loss: 0.6365 - val_accuracy: 0.6790 -
val_auc: 0.7449
Epoch 890/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5774 -
accuracy: 0.7314 - auc: 0.8033 - val_loss: 0.6341 - val_accuracy: 0.6877 -
val_auc: 0.7476
Epoch 891/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5772 -
accuracy: 0.7277 - auc: 0.8029 - val_loss: 0.6339 - val_accuracy: 0.6889 -
val_auc: 0.7486
Epoch 892/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5817 -
accuracy: 0.7198 - auc: 0.7971 - val_loss: 0.6361 - val_accuracy: 0.6815 -
val_auc: 0.7477
Epoch 893/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5748 -
accuracy: 0.7362 - auc: 0.8061 - val_loss: 0.6357 - val_accuracy: 0.6877 -
val_auc: 0.7488
Epoch 894/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5770 -
accuracy: 0.7272 - auc: 0.8045 - val_loss: 0.6313 - val_accuracy: 0.6889 -
val_auc: 0.7516
Epoch 895/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5789 -
accuracy: 0.7243 - auc: 0.8018 - val_loss: 0.6293 - val_accuracy: 0.6914 -
val_auc: 0.7531
Epoch 896/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5768 -
accuracy: 0.7282 - auc: 0.8030 - val_loss: 0.6301 - val_accuracy: 0.6901 -
val_auc: 0.7538
Epoch 897/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5757 -
accuracy: 0.7264 - auc: 0.8023 - val_loss: 0.6283 - val_accuracy: 0.6938 -
val_auc: 0.7566
Epoch 898/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5758 -
accuracy: 0.7274 - auc: 0.8036 - val_loss: 0.6349 - val_accuracy: 0.6877 -
val_auc: 0.7491
Epoch 899/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5777 -
accuracy: 0.7256 - auc: 0.8021 - val_loss: 0.6307 - val_accuracy: 0.6914 -
val_auc: 0.7522
Epoch 900/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5767 -
accuracy: 0.7282 - auc: 0.8028 - val_loss: 0.6334 - val_accuracy: 0.6864 -
val_auc: 0.7512

```
Epoch 901/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5761 -
accuracy: 0.7309 - auc: 0.8022 - val_loss: 0.6298 - val_accuracy: 0.6889 -
val_auc: 0.7529
Epoch 902/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5763 -
accuracy: 0.7277 - auc: 0.8028 - val_loss: 0.6309 - val_accuracy: 0.6914 -
val_auc: 0.7529
Epoch 903/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5721 -
accuracy: 0.7301 - auc: 0.8059 - val_loss: 0.6292 - val_accuracy: 0.6988 -
val_auc: 0.7532
Epoch 904/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5776 -
accuracy: 0.7266 - auc: 0.8008 - val_loss: 0.6297 - val_accuracy: 0.6938 -
val_auc: 0.7525
Epoch 905/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5779 -
accuracy: 0.7224 - auc: 0.8004 - val_loss: 0.6359 - val_accuracy: 0.6864 -
val_auc: 0.7465
Epoch 906/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5749 -
accuracy: 0.7280 - auc: 0.8039 - val_loss: 0.6319 - val_accuracy: 0.6877 -
val_auc: 0.7520
Epoch 907/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5755 -
accuracy: 0.7256 - auc: 0.8040 - val_loss: 0.6308 - val_accuracy: 0.6889 -
val_auc: 0.7515
Epoch 908/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5794 -
accuracy: 0.7232 - auc: 0.7998 - val_loss: 0.6346 - val_accuracy: 0.6926 -
val_auc: 0.7482
Epoch 909/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5740 -
accuracy: 0.7296 - auc: 0.8052 - val_loss: 0.6304 - val_accuracy: 0.6938 -
val_auc: 0.7542
Epoch 910/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5785 -
accuracy: 0.7245 - auc: 0.8003 - val_loss: 0.6297 - val_accuracy: 0.6926 -
val_auc: 0.7526
Epoch 911/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5777 -
accuracy: 0.7330 - auc: 0.8023 - val_loss: 0.6273 - val_accuracy: 0.6975 -
val_auc: 0.7549
Epoch 912/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5751 -
accuracy: 0.7290 - auc: 0.8030 - val_loss: 0.6321 - val_accuracy: 0.6901 -
val_auc: 0.7523
```

```
Epoch 913/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5736 -
accuracy: 0.7306 - auc: 0.8059 - val_loss: 0.6304 - val_accuracy: 0.6877 -
val_auc: 0.7527
Epoch 914/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5730 -
accuracy: 0.7243 - auc: 0.8063 - val_loss: 0.6266 - val_accuracy: 0.6975 -
val_auc: 0.7560
Epoch 915/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5773 -
accuracy: 0.7309 - auc: 0.8024 - val_loss: 0.6349 - val_accuracy: 0.6852 -
val_auc: 0.7487
Epoch 916/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5794 -
accuracy: 0.7277 - auc: 0.7992 - val_loss: 0.6433 - val_accuracy: 0.6753 -
val_auc: 0.7364
Epoch 917/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5780 -
accuracy: 0.7211 - auc: 0.7997 - val_loss: 0.6285 - val_accuracy: 0.6938 -
val_auc: 0.7528
Epoch 918/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5740 -
accuracy: 0.7253 - auc: 0.8049 - val_loss: 0.6277 - val_accuracy: 0.6926 -
val_auc: 0.7552
Epoch 919/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5756 -
accuracy: 0.7314 - auc: 0.8031 - val_loss: 0.6404 - val_accuracy: 0.6852 -
val_auc: 0.7427
Epoch 920/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5744 -
accuracy: 0.7280 - auc: 0.8038 - val_loss: 0.6345 - val_accuracy: 0.6889 -
val_auc: 0.7482
Epoch 921/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5755 -
accuracy: 0.7256 - auc: 0.8032 - val_loss: 0.6280 - val_accuracy: 0.6889 -
val_auc: 0.7563
Epoch 922/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5744 -
accuracy: 0.7317 - auc: 0.8032 - val_loss: 0.6383 - val_accuracy: 0.6815 -
val_auc: 0.7445
Epoch 923/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5758 -
accuracy: 0.7235 - auc: 0.8025 - val_loss: 0.6375 - val_accuracy: 0.6802 -
val_auc: 0.7440
Epoch 924/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5752 -
accuracy: 0.7277 - auc: 0.8034 - val_loss: 0.6298 - val_accuracy: 0.6889 -
val_auc: 0.7518
```

```
Epoch 925/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5750 -
accuracy: 0.7280 - auc: 0.8047 - val_loss: 0.6330 - val_accuracy: 0.6914 -
val_auc: 0.7496
Epoch 926/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5752 -
accuracy: 0.7293 - auc: 0.8046 - val_loss: 0.6296 - val_accuracy: 0.6914 -
val_auc: 0.7520
Epoch 927/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5735 -
accuracy: 0.7282 - auc: 0.8054 - val_loss: 0.6331 - val_accuracy: 0.6889 -
val_auc: 0.7480
Epoch 928/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5744 -
accuracy: 0.7325 - auc: 0.8039 - val_loss: 0.6296 - val_accuracy: 0.6951 -
val_auc: 0.7535
Epoch 929/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5746 -
accuracy: 0.7306 - auc: 0.8036 - val_loss: 0.6295 - val_accuracy: 0.6926 -
val_auc: 0.7533
Epoch 930/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5739 -
accuracy: 0.7288 - auc: 0.8047 - val_loss: 0.6274 - val_accuracy: 0.6926 -
val_auc: 0.7564
Epoch 931/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5762 -
accuracy: 0.7277 - auc: 0.8015 - val_loss: 0.6301 - val_accuracy: 0.6889 -
val_auc: 0.7530
Epoch 932/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5751 -
accuracy: 0.7298 - auc: 0.8036 - val_loss: 0.6313 - val_accuracy: 0.6877 -
val_auc: 0.7518
Epoch 933/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5742 -
accuracy: 0.7293 - auc: 0.8048 - val_loss: 0.6358 - val_accuracy: 0.6889 -
val_auc: 0.7465
Epoch 934/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5745 -
accuracy: 0.7364 - auc: 0.8045 - val_loss: 0.6322 - val_accuracy: 0.6864 -
val_auc: 0.7509
Epoch 935/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5749 -
accuracy: 0.7245 - auc: 0.8033 - val_loss: 0.6280 - val_accuracy: 0.6901 -
val_auc: 0.7527
Epoch 936/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5759 -
accuracy: 0.7378 - auc: 0.8029 - val_loss: 0.6290 - val_accuracy: 0.6914 -
val_auc: 0.7533
```

```
Epoch 937/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5757 -
accuracy: 0.7219 - auc: 0.8027 - val_loss: 0.6313 - val_accuracy: 0.6901 -
val_auc: 0.7523
Epoch 938/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5747 -
accuracy: 0.7290 - auc: 0.8042 - val_loss: 0.6322 - val_accuracy: 0.6926 -
val_auc: 0.7517
Epoch 939/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5750 -
accuracy: 0.7327 - auc: 0.8030 - val_loss: 0.6282 - val_accuracy: 0.6926 -
val_auc: 0.7542
Epoch 940/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5752 -
accuracy: 0.7256 - auc: 0.8034 - val_loss: 0.6327 - val_accuracy: 0.6852 -
val_auc: 0.7496
Epoch 941/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5714 -
accuracy: 0.7322 - auc: 0.8065 - val_loss: 0.6331 - val_accuracy: 0.6864 -
val_auc: 0.7506
Epoch 942/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5717 -
accuracy: 0.7309 - auc: 0.8063 - val_loss: 0.6338 - val_accuracy: 0.6802 -
val_auc: 0.7484
Epoch 943/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5755 -
accuracy: 0.7219 - auc: 0.8019 - val_loss: 0.6275 - val_accuracy: 0.6938 -
val_auc: 0.7539
Epoch 944/1000
119/119 [=====] - 3s 22ms/step - loss: 0.5784 -
accuracy: 0.7245 - auc: 0.8000 - val_loss: 0.6292 - val_accuracy: 0.6914 -
val_auc: 0.7540
Epoch 945/1000
119/119 [=====] - 3s 22ms/step - loss: 0.5747 -
accuracy: 0.7333 - auc: 0.8033 - val_loss: 0.6392 - val_accuracy: 0.6815 -
val_auc: 0.7422
Epoch 946/1000
119/119 [=====] - 3s 22ms/step - loss: 0.5733 -
accuracy: 0.7298 - auc: 0.8050 - val_loss: 0.6271 - val_accuracy: 0.6938 -
val_auc: 0.7546
Epoch 947/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5742 -
accuracy: 0.7364 - auc: 0.8041 - val_loss: 0.6273 - val_accuracy: 0.6889 -
val_auc: 0.7567
Epoch 948/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5734 -
accuracy: 0.7285 - auc: 0.8033 - val_loss: 0.6395 - val_accuracy: 0.6852 -
val_auc: 0.7425
```

Epoch 949/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5757 -
accuracy: 0.7274 - auc: 0.8020 - val_loss: 0.6289 - val_accuracy: 0.6914 -
val_auc: 0.7524
Epoch 950/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5789 -
accuracy: 0.7248 - auc: 0.7981 - val_loss: 0.6321 - val_accuracy: 0.6938 -
val_auc: 0.7512
Epoch 951/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5746 -
accuracy: 0.7335 - auc: 0.8042 - val_loss: 0.6327 - val_accuracy: 0.6840 -
val_auc: 0.7483
Epoch 952/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5746 -
accuracy: 0.7288 - auc: 0.8032 - val_loss: 0.6295 - val_accuracy: 0.6889 -
val_auc: 0.7543
Epoch 953/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5719 -
accuracy: 0.7325 - auc: 0.8059 - val_loss: 0.6338 - val_accuracy: 0.6840 -
val_auc: 0.7490
Epoch 954/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5746 -
accuracy: 0.7304 - auc: 0.8037 - val_loss: 0.6383 - val_accuracy: 0.6840 -
val_auc: 0.7422
Epoch 955/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5728 -
accuracy: 0.7327 - auc: 0.8052 - val_loss: 0.6304 - val_accuracy: 0.6914 -
val_auc: 0.7520
Epoch 956/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5781 -
accuracy: 0.7293 - auc: 0.7997 - val_loss: 0.6246 - val_accuracy: 0.6975 -
val_auc: 0.7561
Epoch 957/1000
119/119 [=====] - 3s 22ms/step - loss: 0.5761 -
accuracy: 0.7277 - auc: 0.8021 - val_loss: 0.6306 - val_accuracy: 0.6889 -
val_auc: 0.7488
Epoch 958/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5749 -
accuracy: 0.7304 - auc: 0.8039 - val_loss: 0.6321 - val_accuracy: 0.6926 -
val_auc: 0.7503
Epoch 959/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5712 -
accuracy: 0.7319 - auc: 0.8076 - val_loss: 0.6350 - val_accuracy: 0.6877 -
val_auc: 0.7453
Epoch 960/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5779 -
accuracy: 0.7211 - auc: 0.8001 - val_loss: 0.6253 - val_accuracy: 0.6901 -
val_auc: 0.7554

```
Epoch 961/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5737 -
accuracy: 0.7351 - auc: 0.8044 - val_loss: 0.6302 - val_accuracy: 0.6877 -
val_auc: 0.7517
Epoch 962/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5731 -
accuracy: 0.7311 - auc: 0.8038 - val_loss: 0.6294 - val_accuracy: 0.6901 -
val_auc: 0.7519
Epoch 963/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5744 -
accuracy: 0.7311 - auc: 0.8035 - val_loss: 0.6337 - val_accuracy: 0.6901 -
val_auc: 0.7475
Epoch 964/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5749 -
accuracy: 0.7290 - auc: 0.8030 - val_loss: 0.6277 - val_accuracy: 0.6901 -
val_auc: 0.7546
Epoch 965/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5726 -
accuracy: 0.7290 - auc: 0.8049 - val_loss: 0.6289 - val_accuracy: 0.6864 -
val_auc: 0.7528
Epoch 966/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5752 -
accuracy: 0.7280 - auc: 0.8019 - val_loss: 0.6340 - val_accuracy: 0.6914 -
val_auc: 0.7487
Epoch 967/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5733 -
accuracy: 0.7277 - auc: 0.8034 - val_loss: 0.6295 - val_accuracy: 0.6877 -
val_auc: 0.7519
Epoch 968/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5704 -
accuracy: 0.7356 - auc: 0.8072 - val_loss: 0.6328 - val_accuracy: 0.6864 -
val_auc: 0.7481
Epoch 969/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5731 -
accuracy: 0.7282 - auc: 0.8038 - val_loss: 0.6265 - val_accuracy: 0.6926 -
val_auc: 0.7535
Epoch 970/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5725 -
accuracy: 0.7298 - auc: 0.8056 - val_loss: 0.6319 - val_accuracy: 0.6938 -
val_auc: 0.7495
Epoch 971/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5760 -
accuracy: 0.7243 - auc: 0.8007 - val_loss: 0.6244 - val_accuracy: 0.6938 -
val_auc: 0.7556
Epoch 972/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5741 -
accuracy: 0.7232 - auc: 0.8027 - val_loss: 0.6327 - val_accuracy: 0.6938 -
val_auc: 0.7489
```

```
Epoch 973/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5726 -
accuracy: 0.7341 - auc: 0.8038 - val_loss: 0.6280 - val_accuracy: 0.6864 -
val_auc: 0.7524
Epoch 974/1000
119/119 [=====] - 3s 22ms/step - loss: 0.5718 -
accuracy: 0.7290 - auc: 0.8044 - val_loss: 0.6295 - val_accuracy: 0.6914 -
val_auc: 0.7528
Epoch 975/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5748 -
accuracy: 0.7272 - auc: 0.8030 - val_loss: 0.6270 - val_accuracy: 0.6926 -
val_auc: 0.7535
Epoch 976/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5714 -
accuracy: 0.7256 - auc: 0.8054 - val_loss: 0.6271 - val_accuracy: 0.6938 -
val_auc: 0.7533
Epoch 977/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5744 -
accuracy: 0.7282 - auc: 0.8025 - val_loss: 0.6272 - val_accuracy: 0.6951 -
val_auc: 0.7536
Epoch 978/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5757 -
accuracy: 0.7301 - auc: 0.8017 - val_loss: 0.6331 - val_accuracy: 0.6877 -
val_auc: 0.7474
Epoch 979/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5713 -
accuracy: 0.7296 - auc: 0.8061 - val_loss: 0.6316 - val_accuracy: 0.6877 -
val_auc: 0.7506
Epoch 980/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5706 -
accuracy: 0.7359 - auc: 0.8063 - val_loss: 0.6302 - val_accuracy: 0.6901 -
val_auc: 0.7512
Epoch 981/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5730 -
accuracy: 0.7301 - auc: 0.8035 - val_loss: 0.6332 - val_accuracy: 0.6827 -
val_auc: 0.7464
Epoch 982/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5715 -
accuracy: 0.7290 - auc: 0.8048 - val_loss: 0.6265 - val_accuracy: 0.6889 -
val_auc: 0.7553
Epoch 983/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5745 -
accuracy: 0.7293 - auc: 0.8021 - val_loss: 0.6302 - val_accuracy: 0.6914 -
val_auc: 0.7518
Epoch 984/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5725 -
accuracy: 0.7333 - auc: 0.8048 - val_loss: 0.6268 - val_accuracy: 0.6914 -
val_auc: 0.7543
```

```
Epoch 985/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5737 -
accuracy: 0.7335 - auc: 0.8039 - val_loss: 0.6255 - val_accuracy: 0.6914 -
val_auc: 0.7545
Epoch 986/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5722 -
accuracy: 0.7280 - auc: 0.8025 - val_loss: 0.6281 - val_accuracy: 0.6975 -
val_auc: 0.7533
Epoch 987/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5726 -
accuracy: 0.7285 - auc: 0.8043 - val_loss: 0.6300 - val_accuracy: 0.6938 -
val_auc: 0.7528
Epoch 988/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5710 -
accuracy: 0.7322 - auc: 0.8048 - val_loss: 0.6276 - val_accuracy: 0.6963 -
val_auc: 0.7528
Epoch 989/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5739 -
accuracy: 0.7325 - auc: 0.8034 - val_loss: 0.6253 - val_accuracy: 0.6951 -
val_auc: 0.7547
Epoch 990/1000
119/119 [=====] - 3s 21ms/step - loss: 0.5682 -
accuracy: 0.7290 - auc: 0.8093 - val_loss: 0.6270 - val_accuracy: 0.6963 -
val_auc: 0.7537
Epoch 991/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5734 -
accuracy: 0.7259 - auc: 0.8040 - val_loss: 0.6295 - val_accuracy: 0.6840 -
val_auc: 0.7515
Epoch 992/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5720 -
accuracy: 0.7243 - auc: 0.8043 - val_loss: 0.6256 - val_accuracy: 0.6926 -
val_auc: 0.7559
Epoch 993/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5729 -
accuracy: 0.7309 - auc: 0.8033 - val_loss: 0.6302 - val_accuracy: 0.6914 -
val_auc: 0.7499
Epoch 994/1000
119/119 [=====] - 2s 20ms/step - loss: 0.5726 -
accuracy: 0.7269 - auc: 0.8036 - val_loss: 0.6353 - val_accuracy: 0.6765 -
val_auc: 0.7461
Epoch 995/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5706 -
accuracy: 0.7327 - auc: 0.8055 - val_loss: 0.6262 - val_accuracy: 0.6914 -
val_auc: 0.7540
Epoch 996/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5719 -
accuracy: 0.7288 - auc: 0.8042 - val_loss: 0.6311 - val_accuracy: 0.6926 -
val_auc: 0.7512
```

```

Epoch 997/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5697 -
accuracy: 0.7325 - auc: 0.8075 - val_loss: 0.6339 - val_accuracy: 0.6790 -
val_auc: 0.7472
Epoch 998/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5716 -
accuracy: 0.7306 - auc: 0.8047 - val_loss: 0.6361 - val_accuracy: 0.6852 -
val_auc: 0.7443
Epoch 999/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5702 -
accuracy: 0.7293 - auc: 0.8064 - val_loss: 0.6316 - val_accuracy: 0.6840 -
val_auc: 0.7496
Epoch 1000/1000
119/119 [=====] - 2s 21ms/step - loss: 0.5693 -
accuracy: 0.7351 - auc: 0.8081 - val_loss: 0.6283 - val_accuracy: 0.6938 -
val_auc: 0.7526
training done

```

We will evaluate with same batch size of 64

```

[ ]: folder = "/content/drive/MyDrive/sCNNEPOCHS1000Bengali(Class_weight)/"
[ ]: import pickle
[ ]: with open(folder + 'trainHistoryDict', 'wb') as file_pi:
      pickle.dump(history, file_pi)

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
3 of 3). These functions will not be directly callable after loading.

[ ]: model.save( folder + 'model.h5')
[ ]: history = pickle.load(open(folder + 'trainHistoryDict', "rb"))
model = tf.keras.models.load_model(folder + 'model.h5')

[ ]: if same:
    evaluation = model.evaluate(X_test, Y_test, batch_size=32)
    print("loss =", evaluation[0])
    print("binary accuracy = {}".format(evaluation[1] * 100))
    print("# iterations =", len(history.history['loss']))

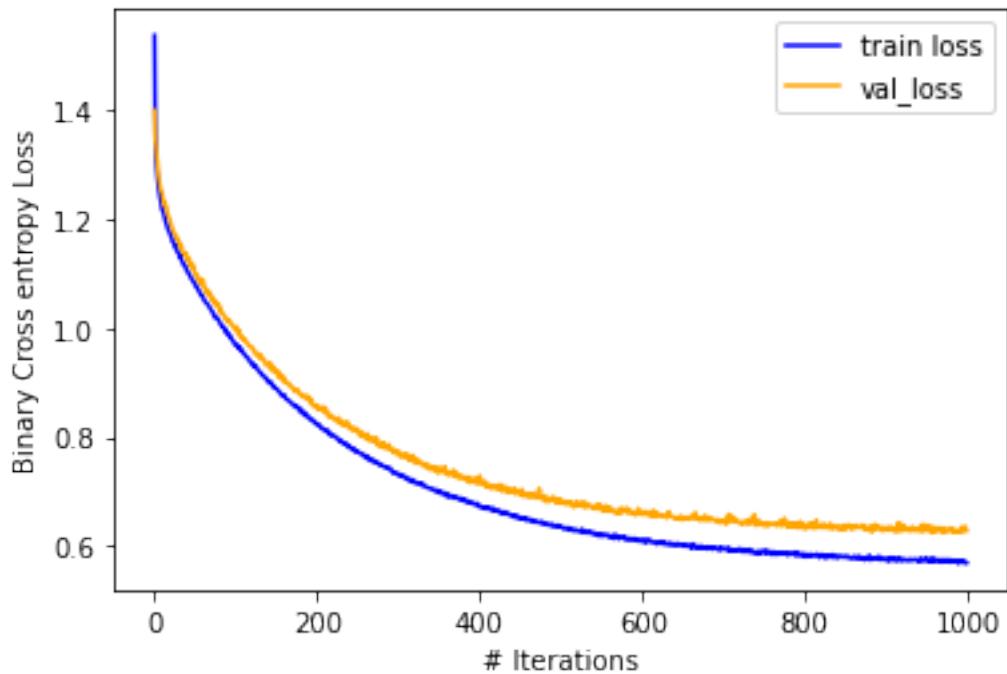
26/26 [=====] - 0s 16ms/step - loss: 0.6120 - accuracy:
0.7090 - auc: 0.7750
loss = 0.6119549870491028
binary accuracy = 70.90012431144714%
# iterations = 1000

[ ]: evaluation

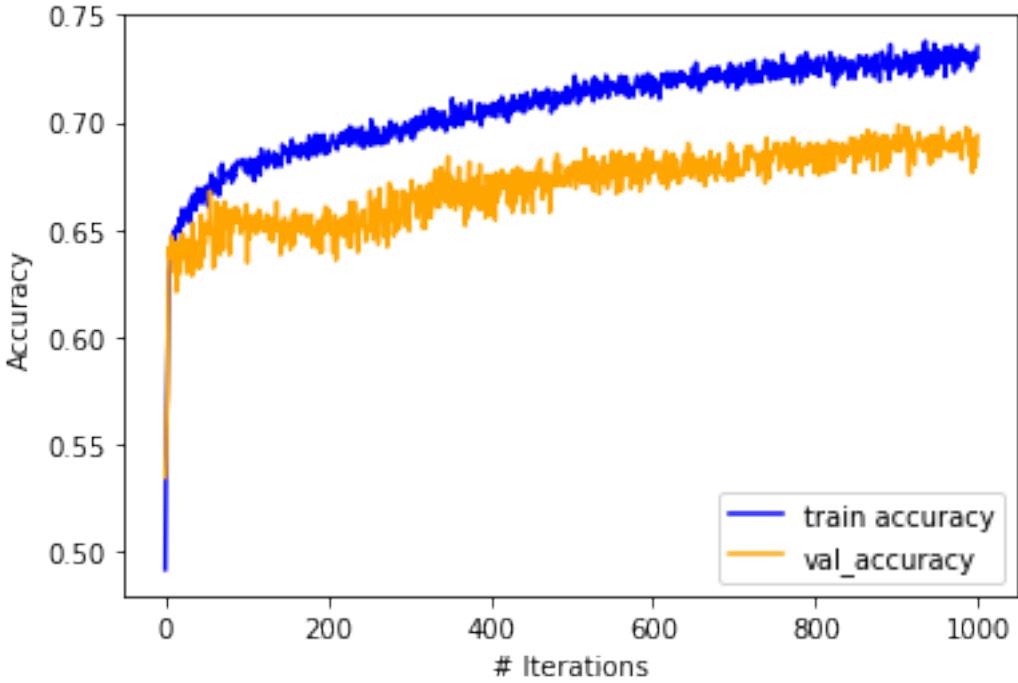
```

```
[ ]: [0.6119549870491028, 0.7090012431144714, 0.7749683856964111]
```

```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Binary Cross entropy Loss')
plt.plot(history.history['loss'], color='blue',label = "train loss")
plt.plot(history.history['val_loss'], color='orange',label = "val_loss")
plt.savefig(folder + "loss.jpeg")
plt.legend()
# plt.grid()
plt.show()
```



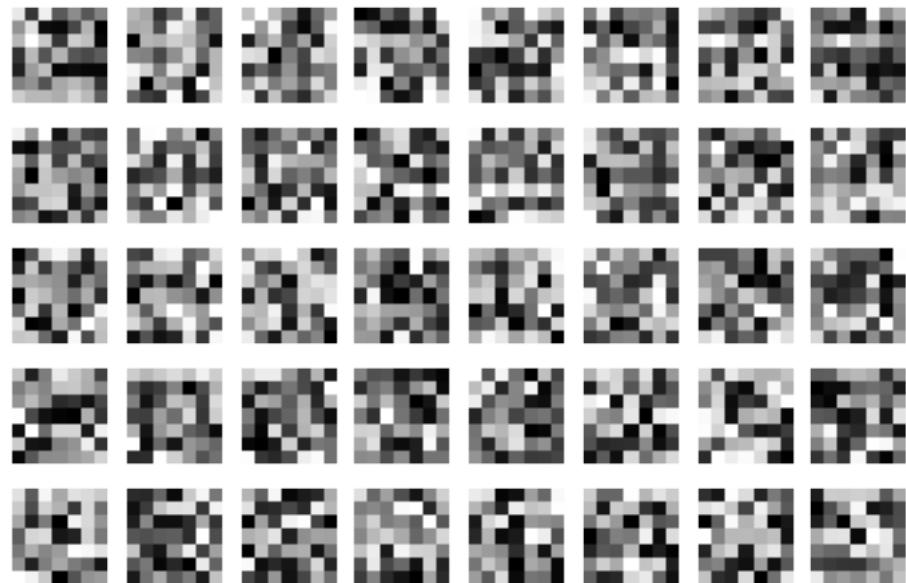
```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Accuracy')
plt.plot(history.history['accuracy'], color='blue',label = "train accuracy")
plt.plot(history.history['val_accuracy'], color='orange',label = "val_accuracy")
# plt.grid()
plt.legend()
plt.savefig(folder + "accuracy.jpeg")
plt.show()
```



```
[ ]: fig, ax = plt.subplots(5,8)

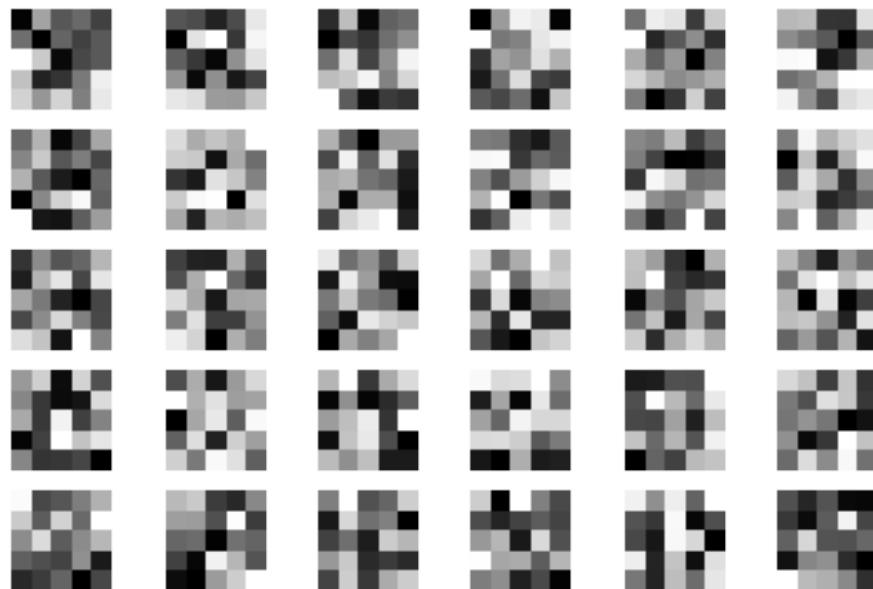
for i in range(40):
    ax[i%5, i//5].imshow(model.layers[0].weights[0][:,:,0,i].numpy(),cmap='gray')
    ax[i%5, i//5].set_axis_off()
fig.suptitle('Layer 1')
plt.show()
```

Layer 1



```
[ ]: fig, ax = plt.subplots(5,6, squeeze=False)
for i in range(30):
    ax[i%5, i//5].imshow(model.layers[3].weights[0][:,:,0,i].numpy(), cmap = 'gray')
    ax[i%5, i//5].set_axis_off()
fig.suptitle('Layer 2')
plt.show()
```

Layer 2



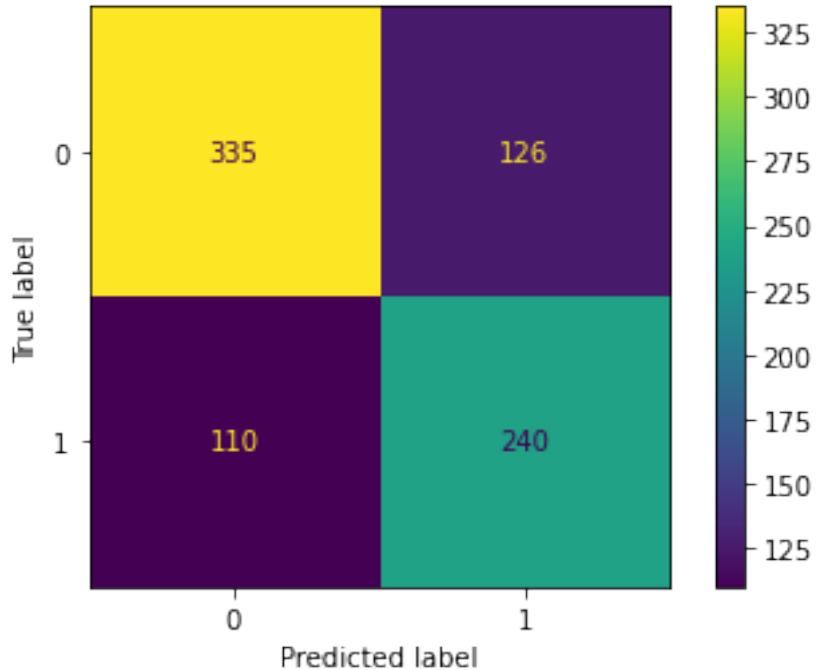
```
[ ]: fig, ax = plt.subplots(5, 4)
for i in range(20):
    ax[i%5, i//5].imshow(model.layers[6].weights[0] [:,:,0,i].numpy(), 'gray')
    ax[i%5, i//5].set_axis_off()
fig.suptitle('Layer 3')
plt.show()
```

Layer 3



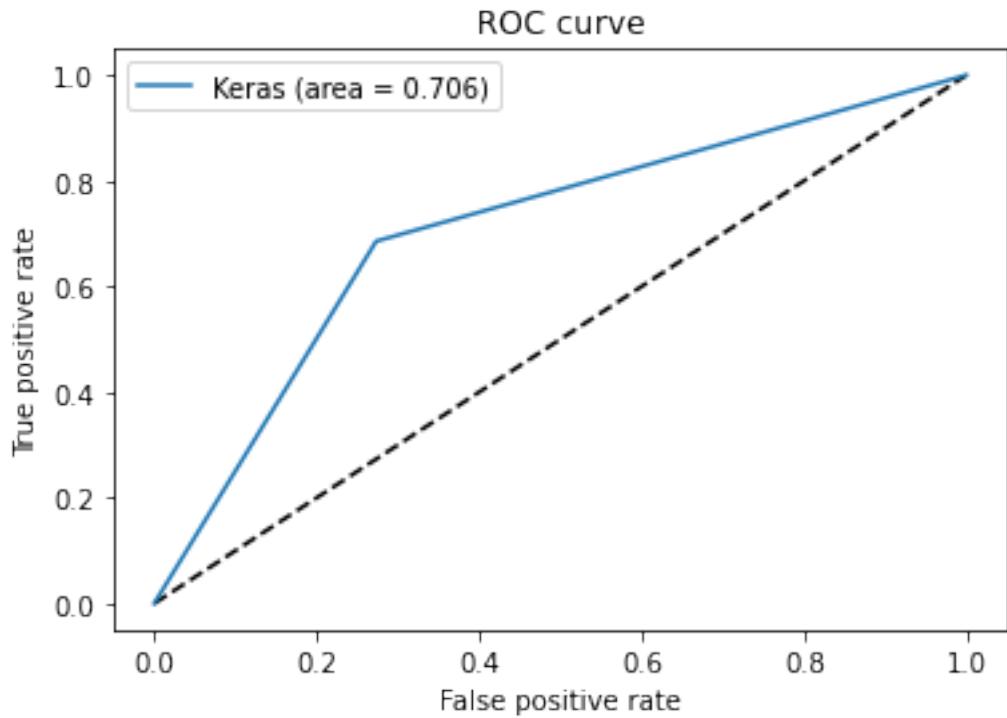
```
[ ]: ConfusionMatrixDisplay(confusion_matrix(tf.argmax(Y_test, axis = 1), tf.  
    ↪argmax(model.predict(X_test), axis = 1))).plot()  
plt.savefig( folder + "confusionmatrix.jpeg")  
plt.show()
```

26/26 [=====] - 0s 7ms/step



```
[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = tf.argmax(model.predict(X_test),axis = 1)
fpr_keras, tpr_keras, thresholds_keras = roc_curve(tf.argmax(Y_test,axis = 1),y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.jpeg")
plt.show()
```

26/26 [=====] - 0s 7ms/step



[]:

Model1_SCNN_BHSig260HindiTraining

November 14, 2024

1 Shallow Convolution Model

This model has only 3 convolution layers with filters of size 7, 5, 3 and 3 max pooling layers.

```
[ ]: from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import LeakyReLU, Softmax
from keras.layers import Conv2D, Activation, Input, Dropout, Lambda, Flatten, Dense
from keras.layers import Dense, Flatten, Reshape, Activation, MaxPool2D
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model

[ ]: import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
[ ]: from google.colab import drive  
drive.mount('/content/drive')
```

Mounted at /content/drive

There are 3 databases available with me. * BhSig100Bengali * BhSig160Hindi * Cedar * Sig-comp2011

We can select any of the three available.

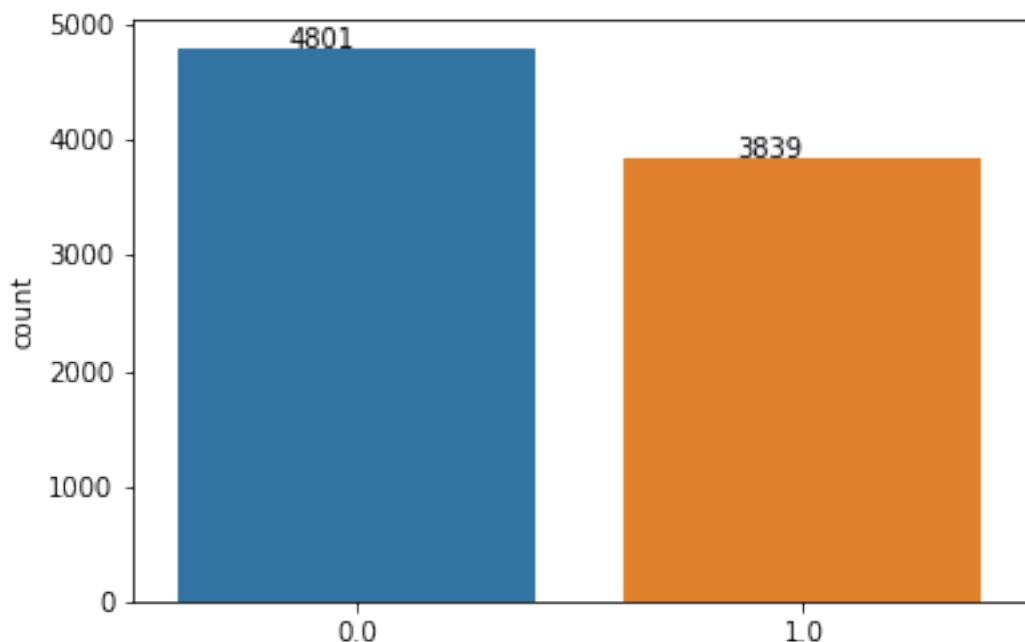
```
[ ]: # database = input('Database Name: ')  
# Models\model1_scnn\database\bhsig100bengali_128x64.h5  
# database = database.lower() + '_128x64.h5'  
# BASE_DIR = os.path.join(os.pardir, os.pardir)  
metadata = {}  
file = "/content/drive/MyDrive/bhsig260hindi_128x643.h5"  
print(file)  
try:  
    with File(file, 'r') as hdf:  
        X = np.array(hdf.get('X'))  
        Y = np.array(hdf.get('Y'))  
        S = np.array(hdf.get('S'))  
except Exception as ex:  
    template = "An exception of type {0} occurred. Arguments:\n{1!r}"  
    message = template.format(type(ex).__name__, ex.args)  
    print(message)  
X = X / 255.0  
Y = Y * 1.0  
# Y = one_hot(Y * 1.0, depth=2)  
# Y = reshape(Y, (-1, 2))  
print("Feature shape =", X.shape)  
print("Label shape =", Y.shape)
```

/content/drive/MyDrive/bhsig260hindi_128x643.h5

Feature shape = (8640, 64, 128, 3)

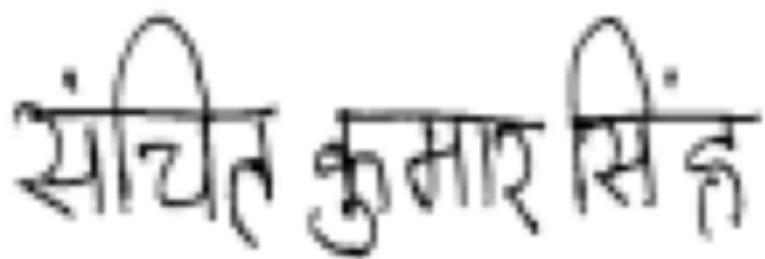
Label shape = (8640, 1)

```
[ ]: ax = sns.countplot(x = Y.reshape(Y.shape[0]))  
for p in ax.patches:  
    ax.annotate('{:}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.  
01))
```



```
[ ]: x = np.random.randint(0,8640)
img = X[x]
print(x)
print(img.shape)
plt.imshow((img * 255).astype(np.uint8))
plt.axis("off")
plt.show()
```

3553
(64, 128, 3)



```
[ ]: # load the image
# img = load_img('bird.jpg')
# convert to numpy array
data = img_to_array(img *255)
# expand dimension to one sample
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(rotation_range=20)
# prepare iterator
it = datagen.flow(samples, batch_size=1)
# generate samples and plot
for i in range(9):
    # define subplot
    plt.subplot(330 + 1 + i)
    # generate batch of images
    batch = it.next()
    # convert to unsigned integers for viewing
    image = batch[0].astype('uint8')
    # plot raw pixel data
    plt.imshow(image)
# pyplot.axis("off")
# show the figure
plt.show()
```



```
[ ]: import sklearn
class_weights = sklearn.utils.class_weight.compute_class_weight(class_weight = "balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict
```

```
[ ]: {0: 0.8998125390543636, 1: 1.1252930450638188}
```

```
[ ]: seed=randint(10)
metadata['split_seed']=seed
print('seed =', seed)
indices = permutation(X.shape[0])
# print(X)
same = True
if same:
    m = int(0.70 * X.shape[0])
    n = int(0.15 * X.shape[0])
    training_id, validation_id, test_id = indices[:m], indices[m: m + n], indices[m+n:]
    X_train, X_test, X_validate = X[training_id], X[test_id], X[validation_id]
    Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]
else:
    m = int(0.70 * X.shape[0])
    training_id, validation_id = indices[:m], indices[m:]
    X_train, X_validate = X[training_id], X[validation_id]
    Y_train, Y_validate = Y[training_id], Y[validation_id]
print("Shape of Features in training set =", X_train.shape)
print("Shape of Labels in training set =", Y_train.shape)

del Y,X
```

```
seed = 1
Shape of Features in training set = (6048, 64, 128, 3)
Shape of Labels in training set = (6048, 1)
```

```
[ ]: #One hot Encoding
Y_train = one_hot(Y_train, depth=2)
Y_train = reshape(Y_train, (-1, 2))

Y_test = one_hot(Y_test, depth=2)
Y_test = reshape(Y_test, (-1, 2))

Y_validate = one_hot(Y_validate, depth=2)
Y_validate = reshape(Y_validate, (-1, 2))
```

```
[ ]: print("Shape of Labels in training set =", Y_train.shape)
```

```
Shape of Labels in training set = (6048, 2)
```

```
[ ]: # X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size = 0.7,
   ↪random_state = 42)

[ ]: # X_test, X_validate, Y_test, Y_validate = train_test_split(X_test, Y_test,
   ↪train_size = 0.5, random_state = 42)

[ ]: # del Y,X

[ ]: seed = randint(10)
metadata['init_seed']=seed
print('seed='+str(seed))
weights1 = np.random.rand(7,7,3,40)
bias1 = np.random.rand(40)
weights2 = np.random.rand(5,5,40,30)
bias2 = np.random.rand(30)
weights3 = np.random.rand(3,3,30,20)
bias3 = np.random.rand(20)

model = Sequential()
model.add(Input(shape = (64, 128, 3),name='input'))
conv1 = Conv2D(40, 7, strides = 1,activation='relu',name='conv1',use_bias = ↪
   ↪True)
model.add(conv1)
model.add(BatchNormalization(epsilon = 1e-08, axis = 1,name = 'batchnorm1', ↪
   ↪momentum = 0.9))
model.add(MaxPool2D(strides=2,name = "Pool1"))

conv2 = Conv2D(30, 5,strides = 1, activation='relu', name='conv2',use_bias = ↪
   ↪True)
model.add(conv2)
model.add(BatchNormalization(epsilon = 1e-08, axis = 1,name = 'batchnorm2', ↪
   ↪momentum = 0.9))
model.add(MaxPool2D(strides=3,name = "Pool2"))

conv3 = Conv2D(20, 3, strides = 1,activation='relu',name='conv3',use_bias = ↪
   ↪True)
model.add(conv3)
model.add(BatchNormalization(epsilon = 1e-08, axis = 1,name = 'batchnorm3', ↪
   ↪momentum = 0.9))
model.add(MaxPool2D(strides=3,name = "Pool3"))

model.add(Flatten())
model.add(Dense(32, activation='relu',
   ↪kernel_initializer=glorot_uniform(seed=seed), kernel_regularizer='l2'))
model.add(Dense(2, activation='softmax',
   ↪kernel_initializer=glorot_uniform(seed=seed), kernel_regularizer='l2'))
```

```

seed=2

[ ]: model.layers

[ ]: model.layers[0].set_weights([weights1,bias1])
model.layers[3].set_weights([weights2,bias2])
model.layers[6].set_weights([weights3,bias3])

[ ]: model.layers[6].get_weights()[0].shape

[ ]: (3, 3, 30, 20)

[ ]: # len(conv1.get_weights())
# [len(a) for a in conv1.get_weights()]

```

The summary of the model is given below:

```
[ ]: print(model.summary())

Model: "sequential"
-----
Layer (type)          Output Shape       Param #
=====
conv1 (Conv2D)        (None, 58, 122, 40)    5920
batchnorm1 (BatchNormalizat (None, 58, 122, 40)    232
ion)

Pool1 (MaxPooling2D)   (None, 29, 61, 40)      0
conv2 (Conv2D)         (None, 25, 57, 30)      30030
batchnorm2 (BatchNormalizat (None, 25, 57, 30)    100
ion)

Pool2 (MaxPooling2D)   (None, 8, 19, 30)      0
conv3 (Conv2D)         (None, 6, 17, 20)      5420
batchnorm3 (BatchNormalizat (None, 6, 17, 20)    24
ion)

Pool3 (MaxPooling2D)   (None, 2, 6, 20)      0
flatten (Flatten)     (None, 240)            0
dense (Dense)          (None, 32)             7712
dense_1 (Dense)        (None, 2)              66
```

```
=====
Total params: 49,504
Trainable params: 49,326
Non-trainable params: 178
```

```
-----  
None
```

In comile process * Optimizer is *Adam* * Loss is *Binary Cross Entropy Loss* * Metrics involve *Binary Accuracy*

```
[ ]: with tf.device('/device:GPU:0'):
    SGD = tf.keras.optimizers.SGD(learning_rate = 1e-04,momentum = 0.9,clipvalue=7)
    model.compile(optimizer = SGD, loss='binary_crossentropy',metrics=['accuracy','AUC'])
    #, 'TruePositives', 'TrueNegatives', 'FalsePositives', 'FalseNegatives'])
```

1.1 Batch Size Testing

```
[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
               epochs = 3,
               batch_size=8)
```

```
Epoch 1/3
648/648 [=====] - 6s 7ms/step - loss: 1.2955 -
binary_accuracy: 0.4379 - val_loss: 1.2547 - val_binary_accuracy: 0.4433
Epoch 2/3
648/648 [=====] - 5s 7ms/step - loss: 1.2257 -
binary_accuracy: 0.4379 - val_loss: 1.2153 - val_binary_accuracy: 0.4433
Epoch 3/3
648/648 [=====] - 4s 7ms/step - loss: 1.1953 -
binary_accuracy: 0.4379 - val_loss: 1.1880 - val_binary_accuracy: 0.4433
```

```
[ ]: <keras.callbacks.History at 0x7fe7b203df50>
```

```
[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
               batch_size=16)
```

```
378/378 [=====] - 6s 13ms/step - loss: 1.5875 -
binary_accuracy: 0.4383 - auc: 0.5000 - val_loss: 1.6461 - val_binary_accuracy:
0.4406 - val_auc: 0.5000
```

```
[ ]: <keras.callbacks.History at 0x7f02b93e4c90>
```

```
[ ]: model.fit(X_train, Y_train, epochs = 10, batch_size = 32,
               validation_data=(X_validate, Y_validate),
               shuffle=True,class_weight = class_weight_dict)
```

```
Epoch 1/10
189/189 [=====] - 4s 19ms/step - loss: 1.2786 -
accuracy: 0.5982 - auc: 0.6303 - val_loss: 1.2695 - val_accuracy: 0.6088 -
val_auc: 0.6411
Epoch 2/10
189/189 [=====] - 3s 18ms/step - loss: 1.2592 -
accuracy: 0.6068 - auc: 0.6442 - val_loss: 1.2550 - val_accuracy: 0.6096 -
val_auc: 0.6480
Epoch 3/10
189/189 [=====] - 3s 18ms/step - loss: 1.2427 -
accuracy: 0.6197 - auc: 0.6619 - val_loss: 1.2426 - val_accuracy: 0.6235 -
val_auc: 0.6662
Epoch 4/10
189/189 [=====] - 3s 18ms/step - loss: 1.2303 -
accuracy: 0.6204 - auc: 0.6707 - val_loss: 1.2340 - val_accuracy: 0.6157 -
val_auc: 0.6640
Epoch 5/10
189/189 [=====] - 3s 18ms/step - loss: 1.2223 -
accuracy: 0.6217 - auc: 0.6740 - val_loss: 1.2245 - val_accuracy: 0.6235 -
val_auc: 0.6730
Epoch 6/10
189/189 [=====] - 3s 18ms/step - loss: 1.2153 -
accuracy: 0.6195 - auc: 0.6763 - val_loss: 1.2156 - val_accuracy: 0.6296 -
val_auc: 0.6782
Epoch 7/10
189/189 [=====] - 3s 18ms/step - loss: 1.2071 -
accuracy: 0.6285 - auc: 0.6808 - val_loss: 1.2139 - val_accuracy: 0.6250 -
val_auc: 0.6685
Epoch 8/10
189/189 [=====] - 4s 19ms/step - loss: 1.1999 -
accuracy: 0.6348 - auc: 0.6831 - val_loss: 1.2049 - val_accuracy: 0.6273 -
val_auc: 0.6782
Epoch 9/10
189/189 [=====] - 4s 19ms/step - loss: 1.1924 -
accuracy: 0.6326 - auc: 0.6876 - val_loss: 1.2009 - val_accuracy: 0.6304 -
val_auc: 0.6769
Epoch 10/10
189/189 [=====] - 4s 20ms/step - loss: 1.1878 -
accuracy: 0.6348 - auc: 0.6875 - val_loss: 1.1969 - val_accuracy: 0.6289 -
val_auc: 0.6759
```

```
[ ]: <keras.callbacks.History at 0x7f72905e3390>
```

```
[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
batch_size=64)
```

```
95/95 [=====] - 4s 28ms/step - loss: 9644.4453 -
binary_accuracy: 0.4395 - val_loss: 2332.6816 - val_binary_accuracy: 0.4591
```

```
[ ]: <keras.callbacks.History at 0x7f2f404ee050>
[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
               batch_size=128)
48/48 [=====] - 4s 51ms/step - loss: 10394.5078 -
binary_accuracy: 0.4395 - val_loss: 4305.1074 - val_binary_accuracy: 0.4591
[ ]: <keras.callbacks.History at 0x7f2f403fa810>
```

1.2 In fit process

- epochs = 20
- batch size = 32

```
[ ]: with tf.device('/device:GPU:0'):
    SGD = tf.keras.optimizers.SGD(learning_rate = 1e-04,momentum = 0.9,clipvalue_
    ↴= 7)
    model.compile(optimizer = SGD, loss='binary_crossentropy',_
    ↴metrics=['accuracy','AUC'])
    #, 'TruePositives', 'TrueNegatives', 'FalsePositives', 'FalseNegatives'])
```



```
[ ]: history = model.fit(X_train, Y_train, epochs = 1000, batch_size = 32,_
    ↴validation_data=(X_validate, Y_validate),shuffle=True
                           ,class_weight = class_weight_dict)
print('training done')
```

Epoch 1/1000
189/189 [=====] - 13s 22ms/step - loss: 1.3943 -
accuracy: 0.5403 - auc: 0.5446 - val_loss: 1.3437 - val_accuracy: 0.5586 -
val_auc: 0.5774
Epoch 2/1000
189/189 [=====] - 3s 18ms/step - loss: 1.3096 -
accuracy: 0.5777 - auc: 0.6093 - val_loss: 1.3031 - val_accuracy: 0.5756 -
val_auc: 0.6072
Epoch 3/1000
189/189 [=====] - 3s 18ms/step - loss: 1.2816 -
accuracy: 0.5952 - auc: 0.6337 - val_loss: 1.2812 - val_accuracy: 0.5918 -
val_auc: 0.6218
Epoch 4/1000
189/189 [=====] - 3s 18ms/step - loss: 1.2630 -
accuracy: 0.6083 - auc: 0.6468 - val_loss: 1.2642 - val_accuracy: 0.5918 -
val_auc: 0.6315
Epoch 5/1000
189/189 [=====] - 3s 18ms/step - loss: 1.2481 -
accuracy: 0.6200 - auc: 0.6594 - val_loss: 1.2533 - val_accuracy: 0.5988 -
val_auc: 0.6391
Epoch 6/1000
189/189 [=====] - 4s 19ms/step - loss: 1.2389 -

```
accuracy: 0.6192 - auc: 0.6635 - val_loss: 1.2416 - val_accuracy: 0.6073 -  
val_auc: 0.6477  
Epoch 7/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.2304 -  
accuracy: 0.6224 - auc: 0.6683 - val_loss: 1.2362 - val_accuracy: 0.6042 -  
val_auc: 0.6493  
Epoch 8/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.2206 -  
accuracy: 0.6288 - auc: 0.6757 - val_loss: 1.2285 - val_accuracy: 0.6034 -  
val_auc: 0.6535  
Epoch 9/1000  
189/189 [=====] - 4s 22ms/step - loss: 1.2156 -  
accuracy: 0.6250 - auc: 0.6750 - val_loss: 1.2222 - val_accuracy: 0.6080 -  
val_auc: 0.6552  
Epoch 10/1000  
189/189 [=====] - 4s 23ms/step - loss: 1.2067 -  
accuracy: 0.6326 - auc: 0.6825 - val_loss: 1.2180 - val_accuracy: 0.6119 -  
val_auc: 0.6531  
Epoch 11/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.2026 -  
accuracy: 0.6319 - auc: 0.6802 - val_loss: 1.2109 - val_accuracy: 0.6088 -  
val_auc: 0.6581  
Epoch 12/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1971 -  
accuracy: 0.6290 - auc: 0.6818 - val_loss: 1.2053 - val_accuracy: 0.6103 -  
val_auc: 0.6591  
Epoch 13/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1907 -  
accuracy: 0.6336 - auc: 0.6847 - val_loss: 1.2027 - val_accuracy: 0.6096 -  
val_auc: 0.6561  
Epoch 14/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1843 -  
accuracy: 0.6359 - auc: 0.6874 - val_loss: 1.1978 - val_accuracy: 0.6134 -  
val_auc: 0.6574  
Epoch 15/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1787 -  
accuracy: 0.6402 - auc: 0.6900 - val_loss: 1.1879 - val_accuracy: 0.6265 -  
val_auc: 0.6695  
Epoch 16/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1744 -  
accuracy: 0.6407 - auc: 0.6904 - val_loss: 1.1815 - val_accuracy: 0.6304 -  
val_auc: 0.6722  
Epoch 17/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1713 -  
accuracy: 0.6353 - auc: 0.6871 - val_loss: 1.1821 - val_accuracy: 0.6173 -  
val_auc: 0.6619  
Epoch 18/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1657 -
```

```
accuracy: 0.6384 - auc: 0.6904 - val_loss: 1.1741 - val_accuracy: 0.6250 -  
val_auc: 0.6684  
Epoch 19/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1609 -  
accuracy: 0.6402 - auc: 0.6929 - val_loss: 1.1695 - val_accuracy: 0.6258 -  
val_auc: 0.6694  
Epoch 20/1000  
189/189 [=====] - 3s 19ms/step - loss: 1.1588 -  
accuracy: 0.6417 - auc: 0.6887 - val_loss: 1.1667 - val_accuracy: 0.6196 -  
val_auc: 0.6670  
Epoch 21/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1502 -  
accuracy: 0.6367 - auc: 0.6939 - val_loss: 1.1603 - val_accuracy: 0.6296 -  
val_auc: 0.6710  
Epoch 22/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1471 -  
accuracy: 0.6427 - auc: 0.6939 - val_loss: 1.1563 - val_accuracy: 0.6289 -  
val_auc: 0.6711  
Epoch 23/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1426 -  
accuracy: 0.6419 - auc: 0.6956 - val_loss: 1.1523 - val_accuracy: 0.6273 -  
val_auc: 0.6713  
Epoch 24/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1385 -  
accuracy: 0.6391 - auc: 0.6926 - val_loss: 1.1488 - val_accuracy: 0.6242 -  
val_auc: 0.6712  
Epoch 25/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.1328 -  
accuracy: 0.6452 - auc: 0.6993 - val_loss: 1.1423 - val_accuracy: 0.6312 -  
val_auc: 0.6759  
Epoch 26/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1311 -  
accuracy: 0.6384 - auc: 0.6950 - val_loss: 1.1407 - val_accuracy: 0.6296 -  
val_auc: 0.6714  
Epoch 27/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1254 -  
accuracy: 0.6432 - auc: 0.6973 - val_loss: 1.1384 - val_accuracy: 0.6242 -  
val_auc: 0.6694  
Epoch 28/1000  
189/189 [=====] - 3s 19ms/step - loss: 1.1210 -  
accuracy: 0.6414 - auc: 0.6977 - val_loss: 1.1282 - val_accuracy: 0.6304 -  
val_auc: 0.6799  
Epoch 29/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1180 -  
accuracy: 0.6429 - auc: 0.6978 - val_loss: 1.1292 - val_accuracy: 0.6258 -  
val_auc: 0.6711  
Epoch 30/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.1149 -
```

```
accuracy: 0.6419 - auc: 0.6956 - val_loss: 1.1235 - val_accuracy: 0.6296 -  
val_auc: 0.6745  
Epoch 31/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.1089 -  
accuracy: 0.6472 - auc: 0.6999 - val_loss: 1.1207 - val_accuracy: 0.6242 -  
val_auc: 0.6729  
Epoch 32/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.1043 -  
accuracy: 0.6452 - auc: 0.6994 - val_loss: 1.1164 - val_accuracy: 0.6265 -  
val_auc: 0.6754  
Epoch 33/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.0992 -  
accuracy: 0.6462 - auc: 0.7046 - val_loss: 1.1098 - val_accuracy: 0.6319 -  
val_auc: 0.6797  
Epoch 34/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.0971 -  
accuracy: 0.6468 - auc: 0.7031 - val_loss: 1.1084 - val_accuracy: 0.6281 -  
val_auc: 0.6761  
Epoch 35/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.0932 -  
accuracy: 0.6467 - auc: 0.7018 - val_loss: 1.1040 - val_accuracy: 0.6335 -  
val_auc: 0.6788  
Epoch 36/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0898 -  
accuracy: 0.6508 - auc: 0.7015 - val_loss: 1.1000 - val_accuracy: 0.6281 -  
val_auc: 0.6783  
Epoch 37/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.0872 -  
accuracy: 0.6432 - auc: 0.7022 - val_loss: 1.0961 - val_accuracy: 0.6343 -  
val_auc: 0.6799  
Epoch 38/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0824 -  
accuracy: 0.6472 - auc: 0.7023 - val_loss: 1.0907 - val_accuracy: 0.6358 -  
val_auc: 0.6833  
Epoch 39/1000  
189/189 [=====] - 4s 20ms/step - loss: 1.0785 -  
accuracy: 0.6490 - auc: 0.7043 - val_loss: 1.0866 - val_accuracy: 0.6335 -  
val_auc: 0.6839  
Epoch 40/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0755 -  
accuracy: 0.6463 - auc: 0.7050 - val_loss: 1.0848 - val_accuracy: 0.6296 -  
val_auc: 0.6804  
Epoch 41/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0706 -  
accuracy: 0.6452 - auc: 0.7047 - val_loss: 1.0823 - val_accuracy: 0.6273 -  
val_auc: 0.6800  
Epoch 42/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0678 -
```

```
accuracy: 0.6465 - auc: 0.7055 - val_loss: 1.0777 - val_accuracy: 0.6335 -  
val_auc: 0.6822  
Epoch 43/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.0623 -  
accuracy: 0.6534 - auc: 0.7075 - val_loss: 1.0753 - val_accuracy: 0.6281 -  
val_auc: 0.6800  
Epoch 44/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0599 -  
accuracy: 0.6505 - auc: 0.7069 - val_loss: 1.0696 - val_accuracy: 0.6296 -  
val_auc: 0.6854  
Epoch 45/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0590 -  
accuracy: 0.6453 - auc: 0.7032 - val_loss: 1.0666 - val_accuracy: 0.6289 -  
val_auc: 0.6841  
Epoch 46/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.0552 -  
accuracy: 0.6501 - auc: 0.7036 - val_loss: 1.0649 - val_accuracy: 0.6281 -  
val_auc: 0.6818  
Epoch 47/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0515 -  
accuracy: 0.6495 - auc: 0.7043 - val_loss: 1.0635 - val_accuracy: 0.6258 -  
val_auc: 0.6790  
Epoch 48/1000  
189/189 [=====] - 3s 18ms/step - loss: 1.0478 -  
accuracy: 0.6505 - auc: 0.7069 - val_loss: 1.0560 - val_accuracy: 0.6343 -  
val_auc: 0.6856  
Epoch 49/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0442 -  
accuracy: 0.6477 - auc: 0.7053 - val_loss: 1.0596 - val_accuracy: 0.6219 -  
val_auc: 0.6747  
Epoch 50/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0396 -  
accuracy: 0.6533 - auc: 0.7090 - val_loss: 1.0530 - val_accuracy: 0.6273 -  
val_auc: 0.6804  
Epoch 51/1000  
189/189 [=====] - 4s 21ms/step - loss: 1.0382 -  
accuracy: 0.6508 - auc: 0.7060 - val_loss: 1.0516 - val_accuracy: 0.6211 -  
val_auc: 0.6773  
Epoch 52/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0349 -  
accuracy: 0.6477 - auc: 0.7076 - val_loss: 1.0453 - val_accuracy: 0.6258 -  
val_auc: 0.6832  
Epoch 53/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0301 -  
accuracy: 0.6506 - auc: 0.7089 - val_loss: 1.0393 - val_accuracy: 0.6296 -  
val_auc: 0.6880  
Epoch 54/1000  
189/189 [=====] - 4s 20ms/step - loss: 1.0284 -
```

```
accuracy: 0.6495 - auc: 0.7064 - val_loss: 1.0369 - val_accuracy: 0.6412 -  
val_auc: 0.6882  
Epoch 55/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0251 -  
accuracy: 0.6508 - auc: 0.7069 - val_loss: 1.0363 - val_accuracy: 0.6258 -  
val_auc: 0.6834  
Epoch 56/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0222 -  
accuracy: 0.6541 - auc: 0.7091 - val_loss: 1.0299 - val_accuracy: 0.6373 -  
val_auc: 0.6891  
Epoch 57/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0193 -  
accuracy: 0.6460 - auc: 0.7069 - val_loss: 1.0293 - val_accuracy: 0.6281 -  
val_auc: 0.6855  
Epoch 58/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0155 -  
accuracy: 0.6548 - auc: 0.7099 - val_loss: 1.0220 - val_accuracy: 0.6397 -  
val_auc: 0.6920  
Epoch 59/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0123 -  
accuracy: 0.6500 - auc: 0.7091 - val_loss: 1.0242 - val_accuracy: 0.6250 -  
val_auc: 0.6834  
Epoch 60/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0098 -  
accuracy: 0.6498 - auc: 0.7092 - val_loss: 1.0177 - val_accuracy: 0.6397 -  
val_auc: 0.6906  
Epoch 61/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0073 -  
accuracy: 0.6559 - auc: 0.7094 - val_loss: 1.0158 - val_accuracy: 0.6381 -  
val_auc: 0.6886  
Epoch 62/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0047 -  
accuracy: 0.6506 - auc: 0.7073 - val_loss: 1.0131 - val_accuracy: 0.6327 -  
val_auc: 0.6880  
Epoch 63/1000  
189/189 [=====] - 4s 19ms/step - loss: 1.0006 -  
accuracy: 0.6544 - auc: 0.7106 - val_loss: 1.0130 - val_accuracy: 0.6273 -  
val_auc: 0.6843  
Epoch 64/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9988 -  
accuracy: 0.6485 - auc: 0.7077 - val_loss: 1.0102 - val_accuracy: 0.6296 -  
val_auc: 0.6837  
Epoch 65/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9956 -  
accuracy: 0.6526 - auc: 0.7085 - val_loss: 1.0039 - val_accuracy: 0.6373 -  
val_auc: 0.6901  
Epoch 66/1000  
189/189 [=====] - 3s 19ms/step - loss: 0.9918 -
```

```
accuracy: 0.6564 - auc: 0.7109 - val_loss: 1.0039 - val_accuracy: 0.6319 -  
val_auc: 0.6856  
Epoch 67/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9899 -  
accuracy: 0.6561 - auc: 0.7106 - val_loss: 1.0012 - val_accuracy: 0.6343 -  
val_auc: 0.6858  
Epoch 68/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9857 -  
accuracy: 0.6589 - auc: 0.7116 - val_loss: 0.9957 - val_accuracy: 0.6373 -  
val_auc: 0.6895  
Epoch 69/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9856 -  
accuracy: 0.6526 - auc: 0.7080 - val_loss: 0.9936 - val_accuracy: 0.6420 -  
val_auc: 0.6899  
Epoch 70/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9790 -  
accuracy: 0.6577 - auc: 0.7144 - val_loss: 0.9884 - val_accuracy: 0.6404 -  
val_auc: 0.6936  
Epoch 71/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.9802 -  
accuracy: 0.6553 - auc: 0.7083 - val_loss: 0.9893 - val_accuracy: 0.6335 -  
val_auc: 0.6875  
Epoch 72/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9761 -  
accuracy: 0.6515 - auc: 0.7105 - val_loss: 0.9852 - val_accuracy: 0.6319 -  
val_auc: 0.6896  
Epoch 73/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9736 -  
accuracy: 0.6548 - auc: 0.7108 - val_loss: 0.9835 - val_accuracy: 0.6350 -  
val_auc: 0.6885  
Epoch 74/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9699 -  
accuracy: 0.6524 - auc: 0.7111 - val_loss: 0.9786 - val_accuracy: 0.6389 -  
val_auc: 0.6938  
Epoch 75/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9672 -  
accuracy: 0.6558 - auc: 0.7136 - val_loss: 0.9755 - val_accuracy: 0.6373 -  
val_auc: 0.6939  
Epoch 76/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9657 -  
accuracy: 0.6553 - auc: 0.7117 - val_loss: 0.9736 - val_accuracy: 0.6366 -  
val_auc: 0.6925  
Epoch 77/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9637 -  
accuracy: 0.6556 - auc: 0.7091 - val_loss: 0.9725 - val_accuracy: 0.6373 -  
val_auc: 0.6898  
Epoch 78/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9599 -
```

```
accuracy: 0.6516 - auc: 0.7125 - val_loss: 0.9698 - val_accuracy: 0.6389 -  
val_auc: 0.6912  
Epoch 79/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9562 -  
accuracy: 0.6546 - auc: 0.7133 - val_loss: 0.9685 - val_accuracy: 0.6373 -  
val_auc: 0.6889  
Epoch 80/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9545 -  
accuracy: 0.6586 - auc: 0.7125 - val_loss: 0.9671 - val_accuracy: 0.6343 -  
val_auc: 0.6873  
Epoch 81/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9518 -  
accuracy: 0.6591 - auc: 0.7132 - val_loss: 0.9652 - val_accuracy: 0.6281 -  
val_auc: 0.6863  
Epoch 82/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9496 -  
accuracy: 0.6533 - auc: 0.7119 - val_loss: 0.9593 - val_accuracy: 0.6358 -  
val_auc: 0.6926  
Epoch 83/1000  
189/189 [=====] - 4s 18ms/step - loss: 0.9463 -  
accuracy: 0.6561 - auc: 0.7137 - val_loss: 0.9568 - val_accuracy: 0.6404 -  
val_auc: 0.6926  
Epoch 84/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9445 -  
accuracy: 0.6541 - auc: 0.7125 - val_loss: 0.9530 - val_accuracy: 0.6381 -  
val_auc: 0.6942  
Epoch 85/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9424 -  
accuracy: 0.6581 - auc: 0.7130 - val_loss: 0.9568 - val_accuracy: 0.6273 -  
val_auc: 0.6845  
Epoch 86/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9378 -  
accuracy: 0.6597 - auc: 0.7163 - val_loss: 0.9483 - val_accuracy: 0.6366 -  
val_auc: 0.6946  
Epoch 87/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9376 -  
accuracy: 0.6503 - auc: 0.7121 - val_loss: 0.9481 - val_accuracy: 0.6404 -  
val_auc: 0.6917  
Epoch 88/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9342 -  
accuracy: 0.6510 - auc: 0.7146 - val_loss: 0.9447 - val_accuracy: 0.6335 -  
val_auc: 0.6924  
Epoch 89/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9327 -  
accuracy: 0.6548 - auc: 0.7134 - val_loss: 0.9427 - val_accuracy: 0.6343 -  
val_auc: 0.6923  
Epoch 90/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9320 -
```

```
accuracy: 0.6543 - auc: 0.7110 - val_loss: 0.9406 - val_accuracy: 0.6389 -  
val_auc: 0.6926  
Epoch 91/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9279 -  
accuracy: 0.6541 - auc: 0.7151 - val_loss: 0.9392 - val_accuracy: 0.6304 -  
val_auc: 0.6905  
Epoch 92/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9266 -  
accuracy: 0.6566 - auc: 0.7129 - val_loss: 0.9332 - val_accuracy: 0.6420 -  
val_auc: 0.6976  
Epoch 93/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9237 -  
accuracy: 0.6577 - auc: 0.7135 - val_loss: 0.9329 - val_accuracy: 0.6350 -  
val_auc: 0.6938  
Epoch 94/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9219 -  
accuracy: 0.6569 - auc: 0.7122 - val_loss: 0.9305 - val_accuracy: 0.6350 -  
val_auc: 0.6937  
Epoch 95/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.9186 -  
accuracy: 0.6586 - auc: 0.7157 - val_loss: 0.9278 - val_accuracy: 0.6412 -  
val_auc: 0.6956  
Epoch 96/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9182 -  
accuracy: 0.6549 - auc: 0.7113 - val_loss: 0.9292 - val_accuracy: 0.6373 -  
val_auc: 0.6900  
Epoch 97/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9151 -  
accuracy: 0.6571 - auc: 0.7144 - val_loss: 0.9262 - val_accuracy: 0.6373 -  
val_auc: 0.6909  
Epoch 98/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9121 -  
accuracy: 0.6566 - auc: 0.7144 - val_loss: 0.9267 - val_accuracy: 0.6296 -  
val_auc: 0.6865  
Epoch 99/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9089 -  
accuracy: 0.6597 - auc: 0.7166 - val_loss: 0.9198 - val_accuracy: 0.6358 -  
val_auc: 0.6942  
Epoch 100/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9090 -  
accuracy: 0.6634 - auc: 0.7141 - val_loss: 0.9184 - val_accuracy: 0.6381 -  
val_auc: 0.6936  
Epoch 101/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9065 -  
accuracy: 0.6546 - auc: 0.7128 - val_loss: 0.9149 - val_accuracy: 0.6389 -  
val_auc: 0.6957  
Epoch 102/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.9040 -
```

```
accuracy: 0.6569 - auc: 0.7140 - val_loss: 0.9137 - val_accuracy: 0.6350 -  
val_auc: 0.6940  
Epoch 103/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.9011 -  
accuracy: 0.6564 - auc: 0.7148 - val_loss: 0.9101 - val_accuracy: 0.6389 -  
val_auc: 0.6974  
Epoch 104/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8987 -  
accuracy: 0.6558 - auc: 0.7157 - val_loss: 0.9095 - val_accuracy: 0.6451 -  
val_auc: 0.6955  
Epoch 105/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8965 -  
accuracy: 0.6596 - auc: 0.7169 - val_loss: 0.9064 - val_accuracy: 0.6397 -  
val_auc: 0.6966  
Epoch 106/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8939 -  
accuracy: 0.6615 - auc: 0.7170 - val_loss: 0.9076 - val_accuracy: 0.6358 -  
val_auc: 0.6915  
Epoch 107/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8929 -  
accuracy: 0.6574 - auc: 0.7156 - val_loss: 0.9011 - val_accuracy: 0.6451 -  
val_auc: 0.6996  
Epoch 108/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8893 -  
accuracy: 0.6597 - auc: 0.7189 - val_loss: 0.9007 - val_accuracy: 0.6381 -  
val_auc: 0.6961  
Epoch 109/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8888 -  
accuracy: 0.6572 - auc: 0.7155 - val_loss: 0.9000 - val_accuracy: 0.6397 -  
val_auc: 0.6948  
Epoch 110/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8874 -  
accuracy: 0.6579 - auc: 0.7154 - val_loss: 0.8985 - val_accuracy: 0.6350 -  
val_auc: 0.6931  
Epoch 111/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8855 -  
accuracy: 0.6617 - auc: 0.7159 - val_loss: 0.8956 - val_accuracy: 0.6404 -  
val_auc: 0.6952  
Epoch 112/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8836 -  
accuracy: 0.6551 - auc: 0.7154 - val_loss: 0.8952 - val_accuracy: 0.6366 -  
val_auc: 0.6931  
Epoch 113/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8792 -  
accuracy: 0.6602 - auc: 0.7192 - val_loss: 0.8903 - val_accuracy: 0.6451 -  
val_auc: 0.6983  
Epoch 114/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8800 -
```

```
accuracy: 0.6551 - auc: 0.7136 - val_loss: 0.8897 - val_accuracy: 0.6373 -  
val_auc: 0.6957  
Epoch 115/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8786 -  
accuracy: 0.6571 - auc: 0.7151 - val_loss: 0.8878 - val_accuracy: 0.6358 -  
val_auc: 0.6964  
Epoch 116/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8757 -  
accuracy: 0.6556 - auc: 0.7162 - val_loss: 0.8893 - val_accuracy: 0.6335 -  
val_auc: 0.6906  
Epoch 117/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8734 -  
accuracy: 0.6615 - auc: 0.7170 - val_loss: 0.8833 - val_accuracy: 0.6373 -  
val_auc: 0.6967  
Epoch 118/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8708 -  
accuracy: 0.6617 - auc: 0.7179 - val_loss: 0.8789 - val_accuracy: 0.6404 -  
val_auc: 0.7010  
Epoch 119/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8711 -  
accuracy: 0.6605 - auc: 0.7146 - val_loss: 0.8788 - val_accuracy: 0.6381 -  
val_auc: 0.6988  
Epoch 120/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8680 -  
accuracy: 0.6599 - auc: 0.7166 - val_loss: 0.8758 - val_accuracy: 0.6443 -  
val_auc: 0.7010  
Epoch 121/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8673 -  
accuracy: 0.6596 - auc: 0.7164 - val_loss: 0.8744 - val_accuracy: 0.6412 -  
val_auc: 0.7003  
Epoch 122/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8630 -  
accuracy: 0.6554 - auc: 0.7179 - val_loss: 0.8712 - val_accuracy: 0.6427 -  
val_auc: 0.7021  
Epoch 123/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8623 -  
accuracy: 0.6577 - auc: 0.7174 - val_loss: 0.8736 - val_accuracy: 0.6373 -  
val_auc: 0.6956  
Epoch 124/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.8604 -  
accuracy: 0.6615 - auc: 0.7170 - val_loss: 0.8709 - val_accuracy: 0.6389 -  
val_auc: 0.6975  
Epoch 125/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8587 -  
accuracy: 0.6637 - auc: 0.7177 - val_loss: 0.8659 - val_accuracy: 0.6427 -  
val_auc: 0.7028  
Epoch 126/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8576 -
```

```
accuracy: 0.6571 - auc: 0.7170 - val_loss: 0.8684 - val_accuracy: 0.6389 -  
val_auc: 0.6960  
Epoch 127/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8550 -  
accuracy: 0.6592 - auc: 0.7177 - val_loss: 0.8648 - val_accuracy: 0.6389 -  
val_auc: 0.6985  
Epoch 128/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8535 -  
accuracy: 0.6604 - auc: 0.7174 - val_loss: 0.8624 - val_accuracy: 0.6443 -  
val_auc: 0.7004  
Epoch 129/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8523 -  
accuracy: 0.6581 - auc: 0.7172 - val_loss: 0.8632 - val_accuracy: 0.6373 -  
val_auc: 0.6961  
Epoch 130/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8492 -  
accuracy: 0.6614 - auc: 0.7193 - val_loss: 0.8580 - val_accuracy: 0.6420 -  
val_auc: 0.7021  
Epoch 131/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8478 -  
accuracy: 0.6625 - auc: 0.7197 - val_loss: 0.8584 - val_accuracy: 0.6389 -  
val_auc: 0.6989  
Epoch 132/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.8479 -  
accuracy: 0.6627 - auc: 0.7174 - val_loss: 0.8563 - val_accuracy: 0.6420 -  
val_auc: 0.7003  
Epoch 133/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8453 -  
accuracy: 0.6566 - auc: 0.7178 - val_loss: 0.8574 - val_accuracy: 0.6412 -  
val_auc: 0.6941  
Epoch 134/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8437 -  
accuracy: 0.6640 - auc: 0.7185 - val_loss: 0.8549 - val_accuracy: 0.6420 -  
val_auc: 0.6963  
Epoch 135/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8419 -  
accuracy: 0.6556 - auc: 0.7178 - val_loss: 0.8509 - val_accuracy: 0.6420 -  
val_auc: 0.7005  
Epoch 136/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8388 -  
accuracy: 0.6587 - auc: 0.7201 - val_loss: 0.8505 - val_accuracy: 0.6404 -  
val_auc: 0.6990  
Epoch 137/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8385 -  
accuracy: 0.6587 - auc: 0.7184 - val_loss: 0.8509 - val_accuracy: 0.6404 -  
val_auc: 0.6957  
Epoch 138/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8354 -
```

```
accuracy: 0.6640 - auc: 0.7216 - val_loss: 0.8460 - val_accuracy: 0.6412 -  
val_auc: 0.7003  
Epoch 139/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.8351 -  
accuracy: 0.6612 - auc: 0.7196 - val_loss: 0.8456 - val_accuracy: 0.6366 -  
val_auc: 0.6985  
Epoch 140/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.8343 -  
accuracy: 0.6612 - auc: 0.7166 - val_loss: 0.8423 - val_accuracy: 0.6420 -  
val_auc: 0.7018  
Epoch 141/1000  
189/189 [=====] - 3s 18ms/step - loss: 0.8321 -  
accuracy: 0.6604 - auc: 0.7192 - val_loss: 0.8472 - val_accuracy: 0.6404 -  
val_auc: 0.6915  
Epoch 142/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8296 -  
accuracy: 0.6630 - auc: 0.7205 - val_loss: 0.8397 - val_accuracy: 0.6427 -  
val_auc: 0.7018  
Epoch 143/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8288 -  
accuracy: 0.6627 - auc: 0.7201 - val_loss: 0.8382 - val_accuracy: 0.6420 -  
val_auc: 0.7011  
Epoch 144/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8274 -  
accuracy: 0.6564 - auc: 0.7176 - val_loss: 0.8348 - val_accuracy: 0.6435 -  
val_auc: 0.7046  
Epoch 145/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8240 -  
accuracy: 0.6579 - auc: 0.7215 - val_loss: 0.8353 - val_accuracy: 0.6443 -  
val_auc: 0.7015  
Epoch 146/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8236 -  
accuracy: 0.6625 - auc: 0.7192 - val_loss: 0.8304 - val_accuracy: 0.6458 -  
val_auc: 0.7068  
Epoch 147/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8233 -  
accuracy: 0.6566 - auc: 0.7170 - val_loss: 0.8336 - val_accuracy: 0.6420 -  
val_auc: 0.6988  
Epoch 148/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8217 -  
accuracy: 0.6619 - auc: 0.7195 - val_loss: 0.8306 - val_accuracy: 0.6420 -  
val_auc: 0.7021  
Epoch 149/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8198 -  
accuracy: 0.6586 - auc: 0.7190 - val_loss: 0.8297 - val_accuracy: 0.6397 -  
val_auc: 0.7014  
Epoch 150/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8174 -
```

```
accuracy: 0.6619 - auc: 0.7218 - val_loss: 0.8274 - val_accuracy: 0.6443 -  
val_auc: 0.7025  
Epoch 151/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8160 -  
accuracy: 0.6634 - auc: 0.7214 - val_loss: 0.8282 - val_accuracy: 0.6420 -  
val_auc: 0.6983  
Epoch 152/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8164 -  
accuracy: 0.6592 - auc: 0.7182 - val_loss: 0.8268 - val_accuracy: 0.6404 -  
val_auc: 0.6987  
Epoch 153/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8123 -  
accuracy: 0.6632 - auc: 0.7227 - val_loss: 0.8230 - val_accuracy: 0.6443 -  
val_auc: 0.7029  
Epoch 154/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8120 -  
accuracy: 0.6609 - auc: 0.7197 - val_loss: 0.8224 - val_accuracy: 0.6427 -  
val_auc: 0.7019  
Epoch 155/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8117 -  
accuracy: 0.6576 - auc: 0.7201 - val_loss: 0.8192 - val_accuracy: 0.6412 -  
val_auc: 0.7046  
Epoch 156/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8094 -  
accuracy: 0.6619 - auc: 0.7198 - val_loss: 0.8195 - val_accuracy: 0.6412 -  
val_auc: 0.7019  
Epoch 157/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8079 -  
accuracy: 0.6609 - auc: 0.7210 - val_loss: 0.8157 - val_accuracy: 0.6435 -  
val_auc: 0.7063  
Epoch 158/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8061 -  
accuracy: 0.6635 - auc: 0.7211 - val_loss: 0.8175 - val_accuracy: 0.6404 -  
val_auc: 0.7012  
Epoch 159/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8052 -  
accuracy: 0.6634 - auc: 0.7207 - val_loss: 0.8145 - val_accuracy: 0.6404 -  
val_auc: 0.7035  
Epoch 160/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8048 -  
accuracy: 0.6566 - auc: 0.7198 - val_loss: 0.8118 - val_accuracy: 0.6373 -  
val_auc: 0.7059  
Epoch 161/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8015 -  
accuracy: 0.6634 - auc: 0.7236 - val_loss: 0.8084 - val_accuracy: 0.6505 -  
val_auc: 0.7094  
Epoch 162/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.8024 -
```

```
accuracy: 0.6571 - auc: 0.7187 - val_loss: 0.8139 - val_accuracy: 0.6389 -  
val_auc: 0.6983  
Epoch 163/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7998 -  
accuracy: 0.6609 - auc: 0.7207 - val_loss: 0.8099 - val_accuracy: 0.6412 -  
val_auc: 0.7027  
Epoch 164/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7990 -  
accuracy: 0.6604 - auc: 0.7194 - val_loss: 0.8089 - val_accuracy: 0.6381 -  
val_auc: 0.7029  
Epoch 165/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7965 -  
accuracy: 0.6620 - auc: 0.7230 - val_loss: 0.8072 - val_accuracy: 0.6381 -  
val_auc: 0.7026  
Epoch 166/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7955 -  
accuracy: 0.6610 - auc: 0.7217 - val_loss: 0.8101 - val_accuracy: 0.6366 -  
val_auc: 0.6962  
Epoch 167/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7938 -  
accuracy: 0.6582 - auc: 0.7205 - val_loss: 0.8078 - val_accuracy: 0.6389 -  
val_auc: 0.6984  
Epoch 168/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7939 -  
accuracy: 0.6572 - auc: 0.7217 - val_loss: 0.8066 - val_accuracy: 0.6404 -  
val_auc: 0.6982  
Epoch 169/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7924 -  
accuracy: 0.6645 - auc: 0.7209 - val_loss: 0.8034 - val_accuracy: 0.6420 -  
val_auc: 0.7018  
Epoch 170/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7892 -  
accuracy: 0.6663 - auc: 0.7239 - val_loss: 0.8000 - val_accuracy: 0.6420 -  
val_auc: 0.7047  
Epoch 171/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7896 -  
accuracy: 0.6605 - auc: 0.7217 - val_loss: 0.8006 - val_accuracy: 0.6427 -  
val_auc: 0.7019  
Epoch 172/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7890 -  
accuracy: 0.6572 - auc: 0.7203 - val_loss: 0.7964 - val_accuracy: 0.6389 -  
val_auc: 0.7069  
Epoch 173/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7868 -  
accuracy: 0.6622 - auc: 0.7230 - val_loss: 0.7976 - val_accuracy: 0.6427 -  
val_auc: 0.7027  
Epoch 174/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7854 -
```

```
accuracy: 0.6648 - auc: 0.7225 - val_loss: 0.7965 - val_accuracy: 0.6427 -  
val_auc: 0.7023  
Epoch 175/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7835 -  
accuracy: 0.6662 - auc: 0.7231 - val_loss: 0.7932 - val_accuracy: 0.6451 -  
val_auc: 0.7055  
Epoch 176/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7825 -  
accuracy: 0.6597 - auc: 0.7232 - val_loss: 0.7932 - val_accuracy: 0.6427 -  
val_auc: 0.7043  
Epoch 177/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7831 -  
accuracy: 0.6584 - auc: 0.7212 - val_loss: 0.7904 - val_accuracy: 0.6420 -  
val_auc: 0.7077  
Epoch 178/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7822 -  
accuracy: 0.6602 - auc: 0.7199 - val_loss: 0.7927 - val_accuracy: 0.6404 -  
val_auc: 0.7015  
Epoch 179/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7788 -  
accuracy: 0.6640 - auc: 0.7241 - val_loss: 0.7883 - val_accuracy: 0.6466 -  
val_auc: 0.7071  
Epoch 180/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7784 -  
accuracy: 0.6614 - auc: 0.7223 - val_loss: 0.7919 - val_accuracy: 0.6420 -  
val_auc: 0.6995  
Epoch 181/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7771 -  
accuracy: 0.6660 - auc: 0.7228 - val_loss: 0.7833 - val_accuracy: 0.6435 -  
val_auc: 0.7109  
Epoch 182/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7750 -  
accuracy: 0.6667 - auc: 0.7241 - val_loss: 0.7867 - val_accuracy: 0.6420 -  
val_auc: 0.7039  
Epoch 183/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7748 -  
accuracy: 0.6624 - auc: 0.7219 - val_loss: 0.7841 - val_accuracy: 0.6435 -  
val_auc: 0.7067  
Epoch 184/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7750 -  
accuracy: 0.6629 - auc: 0.7214 - val_loss: 0.7838 - val_accuracy: 0.6397 -  
val_auc: 0.7051  
Epoch 185/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7716 -  
accuracy: 0.6663 - auc: 0.7248 - val_loss: 0.7835 - val_accuracy: 0.6397 -  
val_auc: 0.7042  
Epoch 186/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7729 -
```

```
accuracy: 0.6648 - auc: 0.7211 - val_loss: 0.7813 - val_accuracy: 0.6412 -  
val_auc: 0.7056  
Epoch 187/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7696 -  
accuracy: 0.6637 - auc: 0.7247 - val_loss: 0.7795 - val_accuracy: 0.6420 -  
val_auc: 0.7069  
Epoch 188/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7695 -  
accuracy: 0.6635 - auc: 0.7239 - val_loss: 0.7789 - val_accuracy: 0.6427 -  
val_auc: 0.7067  
Epoch 189/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7670 -  
accuracy: 0.6634 - auc: 0.7243 - val_loss: 0.7781 - val_accuracy: 0.6435 -  
val_auc: 0.7063  
Epoch 190/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7671 -  
accuracy: 0.6663 - auc: 0.7236 - val_loss: 0.7781 - val_accuracy: 0.6435 -  
val_auc: 0.7039  
Epoch 191/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7669 -  
accuracy: 0.6629 - auc: 0.7217 - val_loss: 0.7770 - val_accuracy: 0.6404 -  
val_auc: 0.7041  
Epoch 192/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7660 -  
accuracy: 0.6610 - auc: 0.7213 - val_loss: 0.7768 - val_accuracy: 0.6404 -  
val_auc: 0.7029  
Epoch 193/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7647 -  
accuracy: 0.6619 - auc: 0.7223 - val_loss: 0.7716 - val_accuracy: 0.6458 -  
val_auc: 0.7094  
Epoch 194/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7646 -  
accuracy: 0.6639 - auc: 0.7204 - val_loss: 0.7739 - val_accuracy: 0.6427 -  
val_auc: 0.7039  
Epoch 195/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7602 -  
accuracy: 0.6673 - auc: 0.7265 - val_loss: 0.7745 - val_accuracy: 0.6404 -  
val_auc: 0.7014  
Epoch 196/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7603 -  
accuracy: 0.6645 - auc: 0.7247 - val_loss: 0.7707 - val_accuracy: 0.6435 -  
val_auc: 0.7061  
Epoch 197/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7607 -  
accuracy: 0.6604 - auc: 0.7208 - val_loss: 0.7707 - val_accuracy: 0.6451 -  
val_auc: 0.7042  
Epoch 198/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7566 -
```

```
accuracy: 0.6639 - auc: 0.7261 - val_loss: 0.7688 - val_accuracy: 0.6420 -  
val_auc: 0.7063  
Epoch 199/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7577 -  
accuracy: 0.6653 - auc: 0.7245 - val_loss: 0.7711 - val_accuracy: 0.6435 -  
val_auc: 0.7008  
Epoch 200/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7561 -  
accuracy: 0.6622 - auc: 0.7247 - val_loss: 0.7655 - val_accuracy: 0.6458 -  
val_auc: 0.7084  
Epoch 201/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7558 -  
accuracy: 0.6584 - auc: 0.7227 - val_loss: 0.7654 - val_accuracy: 0.6427 -  
val_auc: 0.7068  
Epoch 202/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7543 -  
accuracy: 0.6650 - auc: 0.7244 - val_loss: 0.7685 - val_accuracy: 0.6466 -  
val_auc: 0.7006  
Epoch 203/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7542 -  
accuracy: 0.6640 - auc: 0.7240 - val_loss: 0.7614 - val_accuracy: 0.6427 -  
val_auc: 0.7100  
Epoch 204/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7533 -  
accuracy: 0.6605 - auc: 0.7229 - val_loss: 0.7609 - val_accuracy: 0.6397 -  
val_auc: 0.7092  
Epoch 205/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7522 -  
accuracy: 0.6615 - auc: 0.7224 - val_loss: 0.7634 - val_accuracy: 0.6420 -  
val_auc: 0.7043  
Epoch 206/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7506 -  
accuracy: 0.6610 - auc: 0.7244 - val_loss: 0.7599 - val_accuracy: 0.6427 -  
val_auc: 0.7081  
Epoch 207/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7514 -  
accuracy: 0.6632 - auc: 0.7210 - val_loss: 0.7603 - val_accuracy: 0.6435 -  
val_auc: 0.7056  
Epoch 208/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7491 -  
accuracy: 0.6663 - auc: 0.7245 - val_loss: 0.7596 - val_accuracy: 0.6451 -  
val_auc: 0.7072  
Epoch 209/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7477 -  
accuracy: 0.6652 - auc: 0.7243 - val_loss: 0.7564 - val_accuracy: 0.6481 -  
val_auc: 0.7101  
Epoch 210/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7468 -
```

```
accuracy: 0.6650 - auc: 0.7255 - val_loss: 0.7611 - val_accuracy: 0.6420 -  
val_auc: 0.7004  
Epoch 211/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7440 -  
accuracy: 0.6687 - auc: 0.7280 - val_loss: 0.7548 - val_accuracy: 0.6427 -  
val_auc: 0.7091  
Epoch 212/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7471 -  
accuracy: 0.6640 - auc: 0.7221 - val_loss: 0.7567 - val_accuracy: 0.6474 -  
val_auc: 0.7046  
Epoch 213/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7445 -  
accuracy: 0.6627 - auc: 0.7229 - val_loss: 0.7516 - val_accuracy: 0.6505 -  
val_auc: 0.7124  
Epoch 214/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7428 -  
accuracy: 0.6655 - auc: 0.7261 - val_loss: 0.7540 - val_accuracy: 0.6389 -  
val_auc: 0.7065  
Epoch 215/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7417 -  
accuracy: 0.6639 - auc: 0.7248 - val_loss: 0.7543 - val_accuracy: 0.6466 -  
val_auc: 0.7041  
Epoch 216/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7423 -  
accuracy: 0.6622 - auc: 0.7240 - val_loss: 0.7495 - val_accuracy: 0.6451 -  
val_auc: 0.7107  
Epoch 217/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7409 -  
accuracy: 0.6610 - auc: 0.7246 - val_loss: 0.7476 - val_accuracy: 0.6443 -  
val_auc: 0.7124  
Epoch 218/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7395 -  
accuracy: 0.6680 - auc: 0.7252 - val_loss: 0.7505 - val_accuracy: 0.6420 -  
val_auc: 0.7065  
Epoch 219/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7378 -  
accuracy: 0.6663 - auc: 0.7265 - val_loss: 0.7474 - val_accuracy: 0.6427 -  
val_auc: 0.7099  
Epoch 220/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7381 -  
accuracy: 0.6665 - auc: 0.7249 - val_loss: 0.7469 - val_accuracy: 0.6435 -  
val_auc: 0.7095  
Epoch 221/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7362 -  
accuracy: 0.6665 - auc: 0.7270 - val_loss: 0.7475 - val_accuracy: 0.6458 -  
val_auc: 0.7078  
Epoch 222/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7369 -
```

```
accuracy: 0.6612 - auc: 0.7242 - val_loss: 0.7466 - val_accuracy: 0.6404 -  
val_auc: 0.7074  
Epoch 223/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7357 -  
accuracy: 0.6678 - auc: 0.7253 - val_loss: 0.7466 - val_accuracy: 0.6443 -  
val_auc: 0.7060  
Epoch 224/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7337 -  
accuracy: 0.6673 - auc: 0.7260 - val_loss: 0.7457 - val_accuracy: 0.6397 -  
val_auc: 0.7061  
Epoch 225/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7334 -  
accuracy: 0.6635 - auc: 0.7258 - val_loss: 0.7476 - val_accuracy: 0.6451 -  
val_auc: 0.7018  
Epoch 226/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7321 -  
accuracy: 0.6632 - auc: 0.7254 - val_loss: 0.7440 - val_accuracy: 0.6404 -  
val_auc: 0.7066  
Epoch 227/1000  
189/189 [=====] - 4s 21ms/step - loss: 0.7320 -  
accuracy: 0.6637 - auc: 0.7249 - val_loss: 0.7427 - val_accuracy: 0.6451 -  
val_auc: 0.7071  
Epoch 228/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7297 -  
accuracy: 0.6667 - auc: 0.7290 - val_loss: 0.7419 - val_accuracy: 0.6451 -  
val_auc: 0.7072  
Epoch 229/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7299 -  
accuracy: 0.6665 - auc: 0.7258 - val_loss: 0.7398 - val_accuracy: 0.6420 -  
val_auc: 0.7093  
Epoch 230/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7286 -  
accuracy: 0.6701 - auc: 0.7281 - val_loss: 0.7407 - val_accuracy: 0.6412 -  
val_auc: 0.7072  
Epoch 231/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7295 -  
accuracy: 0.6652 - auc: 0.7247 - val_loss: 0.7397 - val_accuracy: 0.6458 -  
val_auc: 0.7075  
Epoch 232/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7261 -  
accuracy: 0.6665 - auc: 0.7282 - val_loss: 0.7420 - val_accuracy: 0.6451 -  
val_auc: 0.7024  
Epoch 233/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7269 -  
accuracy: 0.6658 - auc: 0.7267 - val_loss: 0.7375 - val_accuracy: 0.6389 -  
val_auc: 0.7083  
Epoch 234/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7261 -
```

```
accuracy: 0.6650 - auc: 0.7265 - val_loss: 0.7356 - val_accuracy: 0.6404 -  
val_auc: 0.7102  
Epoch 235/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7258 -  
accuracy: 0.6658 - auc: 0.7261 - val_loss: 0.7377 - val_accuracy: 0.6466 -  
val_auc: 0.7056  
Epoch 236/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7250 -  
accuracy: 0.6627 - auc: 0.7260 - val_loss: 0.7343 - val_accuracy: 0.6458 -  
val_auc: 0.7105  
Epoch 237/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7238 -  
accuracy: 0.6667 - auc: 0.7260 - val_loss: 0.7356 - val_accuracy: 0.6443 -  
val_auc: 0.7071  
Epoch 238/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.7226 -  
accuracy: 0.6682 - auc: 0.7275 - val_loss: 0.7326 - val_accuracy: 0.6427 -  
val_auc: 0.7101  
Epoch 239/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7228 -  
accuracy: 0.6644 - auc: 0.7265 - val_loss: 0.7335 - val_accuracy: 0.6458 -  
val_auc: 0.7079  
Epoch 240/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7200 -  
accuracy: 0.6668 - auc: 0.7285 - val_loss: 0.7365 - val_accuracy: 0.6435 -  
val_auc: 0.7021  
Epoch 241/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7220 -  
accuracy: 0.6630 - auc: 0.7252 - val_loss: 0.7308 - val_accuracy: 0.6420 -  
val_auc: 0.7101  
Epoch 242/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7186 -  
accuracy: 0.6647 - auc: 0.7288 - val_loss: 0.7302 - val_accuracy: 0.6397 -  
val_auc: 0.7101  
Epoch 243/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7185 -  
accuracy: 0.6665 - auc: 0.7291 - val_loss: 0.7276 - val_accuracy: 0.6458 -  
val_auc: 0.7127  
Epoch 244/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7189 -  
accuracy: 0.6673 - auc: 0.7263 - val_loss: 0.7292 - val_accuracy: 0.6412 -  
val_auc: 0.7082  
Epoch 245/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7190 -  
accuracy: 0.6645 - auc: 0.7248 - val_loss: 0.7274 - val_accuracy: 0.6435 -  
val_auc: 0.7110  
Epoch 246/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7172 -
```

```
accuracy: 0.6672 - auc: 0.7266 - val_loss: 0.7298 - val_accuracy: 0.6435 -  
val_auc: 0.7061  
Epoch 247/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7174 -  
accuracy: 0.6622 - auc: 0.7258 - val_loss: 0.7261 - val_accuracy: 0.6427 -  
val_auc: 0.7111  
Epoch 248/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7166 -  
accuracy: 0.6655 - auc: 0.7259 - val_loss: 0.7286 - val_accuracy: 0.6474 -  
val_auc: 0.7055  
Epoch 249/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7139 -  
accuracy: 0.6703 - auc: 0.7290 - val_loss: 0.7248 - val_accuracy: 0.6427 -  
val_auc: 0.7108  
Epoch 250/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7155 -  
accuracy: 0.6678 - auc: 0.7257 - val_loss: 0.7251 - val_accuracy: 0.6481 -  
val_auc: 0.7086  
Epoch 251/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7140 -  
accuracy: 0.6663 - auc: 0.7265 - val_loss: 0.7202 - val_accuracy: 0.6505 -  
val_auc: 0.7153  
Epoch 252/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7122 -  
accuracy: 0.6680 - auc: 0.7278 - val_loss: 0.7237 - val_accuracy: 0.6466 -  
val_auc: 0.7092  
Epoch 253/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7130 -  
accuracy: 0.6696 - auc: 0.7266 - val_loss: 0.7209 - val_accuracy: 0.6520 -  
val_auc: 0.7120  
Epoch 254/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7116 -  
accuracy: 0.6690 - auc: 0.7277 - val_loss: 0.7188 - val_accuracy: 0.6489 -  
val_auc: 0.7145  
Epoch 255/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7123 -  
accuracy: 0.6700 - auc: 0.7259 - val_loss: 0.7187 - val_accuracy: 0.6458 -  
val_auc: 0.7141  
Epoch 256/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7105 -  
accuracy: 0.6642 - auc: 0.7266 - val_loss: 0.7190 - val_accuracy: 0.6443 -  
val_auc: 0.7131  
Epoch 257/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7099 -  
accuracy: 0.6658 - auc: 0.7273 - val_loss: 0.7214 - val_accuracy: 0.6420 -  
val_auc: 0.7081  
Epoch 258/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7083 -
```

```
accuracy: 0.6690 - auc: 0.7286 - val_loss: 0.7226 - val_accuracy: 0.6474 -  
val_auc: 0.7056  
Epoch 259/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7076 -  
accuracy: 0.6663 - auc: 0.7277 - val_loss: 0.7190 - val_accuracy: 0.6427 -  
val_auc: 0.7102  
Epoch 260/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7064 -  
accuracy: 0.6678 - auc: 0.7293 - val_loss: 0.7168 - val_accuracy: 0.6443 -  
val_auc: 0.7123  
Epoch 261/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7067 -  
accuracy: 0.6680 - auc: 0.7276 - val_loss: 0.7190 - val_accuracy: 0.6443 -  
val_auc: 0.7079  
Epoch 262/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7066 -  
accuracy: 0.6642 - auc: 0.7284 - val_loss: 0.7168 - val_accuracy: 0.6427 -  
val_auc: 0.7108  
Epoch 263/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7046 -  
accuracy: 0.6632 - auc: 0.7300 - val_loss: 0.7184 - val_accuracy: 0.6451 -  
val_auc: 0.7074  
Epoch 264/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7038 -  
accuracy: 0.6698 - auc: 0.7288 - val_loss: 0.7179 - val_accuracy: 0.6451 -  
val_auc: 0.7074  
Epoch 265/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7031 -  
accuracy: 0.6688 - auc: 0.7297 - val_loss: 0.7135 - val_accuracy: 0.6466 -  
val_auc: 0.7134  
Epoch 266/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7030 -  
accuracy: 0.6660 - auc: 0.7287 - val_loss: 0.7140 - val_accuracy: 0.6466 -  
val_auc: 0.7113  
Epoch 267/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7040 -  
accuracy: 0.6667 - auc: 0.7266 - val_loss: 0.7130 - val_accuracy: 0.6451 -  
val_auc: 0.7121  
Epoch 268/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7019 -  
accuracy: 0.6678 - auc: 0.7298 - val_loss: 0.7142 - val_accuracy: 0.6451 -  
val_auc: 0.7093  
Epoch 269/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7027 -  
accuracy: 0.6658 - auc: 0.7268 - val_loss: 0.7112 - val_accuracy: 0.6404 -  
val_auc: 0.7127  
Epoch 270/1000  
189/189 [=====] - 4s 21ms/step - loss: 0.7023 -
```

```
accuracy: 0.6639 - auc: 0.7267 - val_loss: 0.7135 - val_accuracy: 0.6505 -  
val_auc: 0.7089  
Epoch 271/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6999 -  
accuracy: 0.6744 - auc: 0.7304 - val_loss: 0.7113 - val_accuracy: 0.6451 -  
val_auc: 0.7111  
Epoch 272/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.7002 -  
accuracy: 0.6693 - auc: 0.7284 - val_loss: 0.7117 - val_accuracy: 0.6481 -  
val_auc: 0.7095  
Epoch 273/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6987 -  
accuracy: 0.6705 - auc: 0.7296 - val_loss: 0.7063 - val_accuracy: 0.6489 -  
val_auc: 0.7168  
Epoch 274/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6976 -  
accuracy: 0.6734 - auc: 0.7304 - val_loss: 0.7071 - val_accuracy: 0.6474 -  
val_auc: 0.7147  
Epoch 275/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6993 -  
accuracy: 0.6635 - auc: 0.7278 - val_loss: 0.7050 - val_accuracy: 0.6481 -  
val_auc: 0.7172  
Epoch 276/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6976 -  
accuracy: 0.6701 - auc: 0.7289 - val_loss: 0.7049 - val_accuracy: 0.6497 -  
val_auc: 0.7169  
Epoch 277/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6967 -  
accuracy: 0.6660 - auc: 0.7295 - val_loss: 0.7071 - val_accuracy: 0.6451 -  
val_auc: 0.7121  
Epoch 278/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6968 -  
accuracy: 0.6688 - auc: 0.7281 - val_loss: 0.7067 - val_accuracy: 0.6481 -  
val_auc: 0.7118  
Epoch 279/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6966 -  
accuracy: 0.6672 - auc: 0.7273 - val_loss: 0.7048 - val_accuracy: 0.6497 -  
val_auc: 0.7138  
Epoch 280/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6936 -  
accuracy: 0.6687 - auc: 0.7317 - val_loss: 0.7066 - val_accuracy: 0.6474 -  
val_auc: 0.7112  
Epoch 281/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6948 -  
accuracy: 0.6639 - auc: 0.7287 - val_loss: 0.7065 - val_accuracy: 0.6512 -  
val_auc: 0.7100  
Epoch 282/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6940 -
```

```
accuracy: 0.6713 - auc: 0.7303 - val_loss: 0.7027 - val_accuracy: 0.6489 -  
val_auc: 0.7150  
Epoch 283/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6950 -  
accuracy: 0.6644 - auc: 0.7282 - val_loss: 0.7049 - val_accuracy: 0.6466 -  
val_auc: 0.7106  
Epoch 284/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6915 -  
accuracy: 0.6683 - auc: 0.7309 - val_loss: 0.7013 - val_accuracy: 0.6497 -  
val_auc: 0.7157  
Epoch 285/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6925 -  
accuracy: 0.6720 - auc: 0.7299 - val_loss: 0.7035 - val_accuracy: 0.6489 -  
val_auc: 0.7116  
Epoch 286/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6927 -  
accuracy: 0.6663 - auc: 0.7279 - val_loss: 0.7053 - val_accuracy: 0.6489 -  
val_auc: 0.7080  
Epoch 287/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6906 -  
accuracy: 0.6691 - auc: 0.7306 - val_loss: 0.7021 - val_accuracy: 0.6458 -  
val_auc: 0.7121  
Epoch 288/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6899 -  
accuracy: 0.6652 - auc: 0.7307 - val_loss: 0.7003 - val_accuracy: 0.6505 -  
val_auc: 0.7142  
Epoch 289/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6902 -  
accuracy: 0.6663 - auc: 0.7296 - val_loss: 0.7016 - val_accuracy: 0.6481 -  
val_auc: 0.7118  
Epoch 290/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6902 -  
accuracy: 0.6658 - auc: 0.7297 - val_loss: 0.6997 - val_accuracy: 0.6481 -  
val_auc: 0.7139  
Epoch 291/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6896 -  
accuracy: 0.6688 - auc: 0.7294 - val_loss: 0.6998 - val_accuracy: 0.6474 -  
val_auc: 0.7125  
Epoch 292/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6880 -  
accuracy: 0.6693 - auc: 0.7313 - val_loss: 0.6973 - val_accuracy: 0.6489 -  
val_auc: 0.7161  
Epoch 293/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6879 -  
accuracy: 0.6690 - auc: 0.7299 - val_loss: 0.6983 - val_accuracy: 0.6443 -  
val_auc: 0.7136  
Epoch 294/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6862 -
```

```
accuracy: 0.6713 - auc: 0.7318 - val_loss: 0.6982 - val_accuracy: 0.6404 -  
val_auc: 0.7132  
Epoch 295/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6872 -  
accuracy: 0.6637 - auc: 0.7299 - val_loss: 0.6937 - val_accuracy: 0.6528 -  
val_auc: 0.7193  
Epoch 296/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6872 -  
accuracy: 0.6617 - auc: 0.7295 - val_loss: 0.6969 - val_accuracy: 0.6474 -  
val_auc: 0.7137  
Epoch 297/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6862 -  
accuracy: 0.6622 - auc: 0.7303 - val_loss: 0.6972 - val_accuracy: 0.6443 -  
val_auc: 0.7121  
Epoch 298/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6857 -  
accuracy: 0.6687 - auc: 0.7298 - val_loss: 0.6946 - val_accuracy: 0.6443 -  
val_auc: 0.7157  
Epoch 299/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6861 -  
accuracy: 0.6673 - auc: 0.7301 - val_loss: 0.6943 - val_accuracy: 0.6435 -  
val_auc: 0.7153  
Epoch 300/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6842 -  
accuracy: 0.6667 - auc: 0.7306 - val_loss: 0.6955 - val_accuracy: 0.6458 -  
val_auc: 0.7127  
Epoch 301/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6846 -  
accuracy: 0.6682 - auc: 0.7292 - val_loss: 0.6926 - val_accuracy: 0.6497 -  
val_auc: 0.7169  
Epoch 302/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6830 -  
accuracy: 0.6677 - auc: 0.7313 - val_loss: 0.6964 - val_accuracy: 0.6427 -  
val_auc: 0.7100  
Epoch 303/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6831 -  
accuracy: 0.6733 - auc: 0.7314 - val_loss: 0.6912 - val_accuracy: 0.6528 -  
val_auc: 0.7178  
Epoch 304/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6829 -  
accuracy: 0.6683 - auc: 0.7311 - val_loss: 0.6911 - val_accuracy: 0.6489 -  
val_auc: 0.7169  
Epoch 305/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6821 -  
accuracy: 0.6700 - auc: 0.7306 - val_loss: 0.6913 - val_accuracy: 0.6520 -  
val_auc: 0.7159  
Epoch 306/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6812 -
```

```
accuracy: 0.6695 - auc: 0.7316 - val_loss: 0.6918 - val_accuracy: 0.6512 -  
val_auc: 0.7148  
Epoch 307/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6804 -  
accuracy: 0.6653 - auc: 0.7311 - val_loss: 0.6907 - val_accuracy: 0.6458 -  
val_auc: 0.7159  
Epoch 308/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6824 -  
accuracy: 0.6637 - auc: 0.7289 - val_loss: 0.6917 - val_accuracy: 0.6481 -  
val_auc: 0.7143  
Epoch 309/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6796 -  
accuracy: 0.6687 - auc: 0.7324 - val_loss: 0.6897 - val_accuracy: 0.6489 -  
val_auc: 0.7153  
Epoch 310/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6785 -  
accuracy: 0.6701 - auc: 0.7332 - val_loss: 0.6896 - val_accuracy: 0.6489 -  
val_auc: 0.7158  
Epoch 311/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6786 -  
accuracy: 0.6660 - auc: 0.7324 - val_loss: 0.6891 - val_accuracy: 0.6451 -  
val_auc: 0.7152  
Epoch 312/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6781 -  
accuracy: 0.6662 - auc: 0.7319 - val_loss: 0.6912 - val_accuracy: 0.6481 -  
val_auc: 0.7120  
Epoch 313/1000  
189/189 [=====] - 4s 21ms/step - loss: 0.6790 -  
accuracy: 0.6677 - auc: 0.7310 - val_loss: 0.6898 - val_accuracy: 0.6489 -  
val_auc: 0.7128  
Epoch 314/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6776 -  
accuracy: 0.6655 - auc: 0.7311 - val_loss: 0.6899 - val_accuracy: 0.6481 -  
val_auc: 0.7127  
Epoch 315/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6792 -  
accuracy: 0.6690 - auc: 0.7291 - val_loss: 0.6892 - val_accuracy: 0.6466 -  
val_auc: 0.7130  
Epoch 316/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6784 -  
accuracy: 0.6690 - auc: 0.7303 - val_loss: 0.6868 - val_accuracy: 0.6505 -  
val_auc: 0.7159  
Epoch 317/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6760 -  
accuracy: 0.6703 - auc: 0.7315 - val_loss: 0.6861 - val_accuracy: 0.6512 -  
val_auc: 0.7172  
Epoch 318/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6752 -
```

```
accuracy: 0.6685 - auc: 0.7336 - val_loss: 0.6851 - val_accuracy: 0.6489 -  
val_auc: 0.7172  
Epoch 319/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6753 -  
accuracy: 0.6736 - auc: 0.7326 - val_loss: 0.6843 - val_accuracy: 0.6512 -  
val_auc: 0.7186  
Epoch 320/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6763 -  
accuracy: 0.6698 - auc: 0.7307 - val_loss: 0.6855 - val_accuracy: 0.6481 -  
val_auc: 0.7160  
Epoch 321/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6741 -  
accuracy: 0.6711 - auc: 0.7336 - val_loss: 0.6865 - val_accuracy: 0.6489 -  
val_auc: 0.7136  
Epoch 322/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6747 -  
accuracy: 0.6685 - auc: 0.7313 - val_loss: 0.6837 - val_accuracy: 0.6497 -  
val_auc: 0.7174  
Epoch 323/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6729 -  
accuracy: 0.6723 - auc: 0.7343 - val_loss: 0.6818 - val_accuracy: 0.6528 -  
val_auc: 0.7195  
Epoch 324/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6737 -  
accuracy: 0.6733 - auc: 0.7322 - val_loss: 0.6836 - val_accuracy: 0.6497 -  
val_auc: 0.7162  
Epoch 325/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6724 -  
accuracy: 0.6728 - auc: 0.7332 - val_loss: 0.6836 - val_accuracy: 0.6481 -  
val_auc: 0.7154  
Epoch 326/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6703 -  
accuracy: 0.6729 - auc: 0.7357 - val_loss: 0.6852 - val_accuracy: 0.6466 -  
val_auc: 0.7122  
Epoch 327/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6739 -  
accuracy: 0.6662 - auc: 0.7307 - val_loss: 0.6851 - val_accuracy: 0.6474 -  
val_auc: 0.7118  
Epoch 328/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6721 -  
accuracy: 0.6690 - auc: 0.7316 - val_loss: 0.6841 - val_accuracy: 0.6466 -  
val_auc: 0.7125  
Epoch 329/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6699 -  
accuracy: 0.6698 - auc: 0.7341 - val_loss: 0.6780 - val_accuracy: 0.6535 -  
val_auc: 0.7219  
Epoch 330/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6712 -
```

```
accuracy: 0.6729 - auc: 0.7332 - val_loss: 0.6808 - val_accuracy: 0.6520 -  
val_auc: 0.7170  
Epoch 331/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6703 -  
accuracy: 0.6716 - auc: 0.7328 - val_loss: 0.6793 - val_accuracy: 0.6497 -  
val_auc: 0.7182  
Epoch 332/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6689 -  
accuracy: 0.6700 - auc: 0.7334 - val_loss: 0.6852 - val_accuracy: 0.6489 -  
val_auc: 0.7089  
Epoch 333/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6693 -  
accuracy: 0.6703 - auc: 0.7339 - val_loss: 0.6795 - val_accuracy: 0.6512 -  
val_auc: 0.7172  
Epoch 334/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6692 -  
accuracy: 0.6682 - auc: 0.7331 - val_loss: 0.6805 - val_accuracy: 0.6497 -  
val_auc: 0.7153  
Epoch 335/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6675 -  
accuracy: 0.6723 - auc: 0.7342 - val_loss: 0.6827 - val_accuracy: 0.6497 -  
val_auc: 0.7116  
Epoch 336/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6687 -  
accuracy: 0.6720 - auc: 0.7329 - val_loss: 0.6814 - val_accuracy: 0.6528 -  
val_auc: 0.7131  
Epoch 337/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6674 -  
accuracy: 0.6698 - auc: 0.7333 - val_loss: 0.6779 - val_accuracy: 0.6481 -  
val_auc: 0.7181  
Epoch 338/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6686 -  
accuracy: 0.6682 - auc: 0.7315 - val_loss: 0.6763 - val_accuracy: 0.6535 -  
val_auc: 0.7200  
Epoch 339/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6676 -  
accuracy: 0.6721 - auc: 0.7343 - val_loss: 0.6778 - val_accuracy: 0.6451 -  
val_auc: 0.7166  
Epoch 340/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6674 -  
accuracy: 0.6690 - auc: 0.7329 - val_loss: 0.6808 - val_accuracy: 0.6451 -  
val_auc: 0.7120  
Epoch 341/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6662 -  
accuracy: 0.6715 - auc: 0.7337 - val_loss: 0.6760 - val_accuracy: 0.6497 -  
val_auc: 0.7187  
Epoch 342/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6657 -
```

```
accuracy: 0.6711 - auc: 0.7348 - val_loss: 0.6766 - val_accuracy: 0.6535 -  
val_auc: 0.7181  
Epoch 343/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6651 -  
accuracy: 0.6682 - auc: 0.7339 - val_loss: 0.6771 - val_accuracy: 0.6497 -  
val_auc: 0.7158  
Epoch 344/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6667 -  
accuracy: 0.6725 - auc: 0.7328 - val_loss: 0.6752 - val_accuracy: 0.6505 -  
val_auc: 0.7186  
Epoch 345/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6654 -  
accuracy: 0.6700 - auc: 0.7331 - val_loss: 0.6758 - val_accuracy: 0.6497 -  
val_auc: 0.7169  
Epoch 346/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6657 -  
accuracy: 0.6713 - auc: 0.7334 - val_loss: 0.6739 - val_accuracy: 0.6497 -  
val_auc: 0.7191  
Epoch 347/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6655 -  
accuracy: 0.6672 - auc: 0.7319 - val_loss: 0.6722 - val_accuracy: 0.6543 -  
val_auc: 0.7212  
Epoch 348/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6640 -  
accuracy: 0.6736 - auc: 0.7342 - val_loss: 0.6769 - val_accuracy: 0.6458 -  
val_auc: 0.7135  
Epoch 349/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6640 -  
accuracy: 0.6691 - auc: 0.7330 - val_loss: 0.6741 - val_accuracy: 0.6512 -  
val_auc: 0.7183  
Epoch 350/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6622 -  
accuracy: 0.6713 - auc: 0.7347 - val_loss: 0.6739 - val_accuracy: 0.6489 -  
val_auc: 0.7176  
Epoch 351/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6644 -  
accuracy: 0.6713 - auc: 0.7326 - val_loss: 0.6722 - val_accuracy: 0.6497 -  
val_auc: 0.7196  
Epoch 352/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6632 -  
accuracy: 0.6680 - auc: 0.7334 - val_loss: 0.6711 - val_accuracy: 0.6505 -  
val_auc: 0.7201  
Epoch 353/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6617 -  
accuracy: 0.6733 - auc: 0.7352 - val_loss: 0.6777 - val_accuracy: 0.6458 -  
val_auc: 0.7103  
Epoch 354/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6607 -
```

```
accuracy: 0.6721 - auc: 0.7349 - val_loss: 0.6736 - val_accuracy: 0.6505 -  
val_auc: 0.7162  
Epoch 355/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6608 -  
accuracy: 0.6687 - auc: 0.7352 - val_loss: 0.6717 - val_accuracy: 0.6466 -  
val_auc: 0.7183  
Epoch 356/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6610 -  
accuracy: 0.6688 - auc: 0.7346 - val_loss: 0.6706 - val_accuracy: 0.6489 -  
val_auc: 0.7199  
Epoch 357/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6597 -  
accuracy: 0.6744 - auc: 0.7361 - val_loss: 0.6704 - val_accuracy: 0.6512 -  
val_auc: 0.7199  
Epoch 358/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6598 -  
accuracy: 0.6729 - auc: 0.7360 - val_loss: 0.6711 - val_accuracy: 0.6489 -  
val_auc: 0.7181  
Epoch 359/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6602 -  
accuracy: 0.6696 - auc: 0.7346 - val_loss: 0.6701 - val_accuracy: 0.6481 -  
val_auc: 0.7195  
Epoch 360/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6612 -  
accuracy: 0.6715 - auc: 0.7343 - val_loss: 0.6727 - val_accuracy: 0.6481 -  
val_auc: 0.7152  
Epoch 361/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6593 -  
accuracy: 0.6701 - auc: 0.7347 - val_loss: 0.6689 - val_accuracy: 0.6528 -  
val_auc: 0.7207  
Epoch 362/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6603 -  
accuracy: 0.6703 - auc: 0.7335 - val_loss: 0.6688 - val_accuracy: 0.6497 -  
val_auc: 0.7202  
Epoch 363/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6583 -  
accuracy: 0.6754 - auc: 0.7364 - val_loss: 0.6682 - val_accuracy: 0.6497 -  
val_auc: 0.7202  
Epoch 364/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6590 -  
accuracy: 0.6734 - auc: 0.7349 - val_loss: 0.6717 - val_accuracy: 0.6451 -  
val_auc: 0.7151  
Epoch 365/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6587 -  
accuracy: 0.6696 - auc: 0.7350 - val_loss: 0.6687 - val_accuracy: 0.6505 -  
val_auc: 0.7186  
Epoch 366/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6582 -
```

```
accuracy: 0.6706 - auc: 0.7340 - val_loss: 0.6661 - val_accuracy: 0.6543 -  
val_auc: 0.7221  
Epoch 367/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6575 -  
accuracy: 0.6710 - auc: 0.7349 - val_loss: 0.6647 - val_accuracy: 0.6551 -  
val_auc: 0.7245  
Epoch 368/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6575 -  
accuracy: 0.6749 - auc: 0.7350 - val_loss: 0.6708 - val_accuracy: 0.6497 -  
val_auc: 0.7146  
Epoch 369/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6560 -  
accuracy: 0.6728 - auc: 0.7367 - val_loss: 0.6692 - val_accuracy: 0.6489 -  
val_auc: 0.7166  
Epoch 370/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6557 -  
accuracy: 0.6721 - auc: 0.7372 - val_loss: 0.6679 - val_accuracy: 0.6512 -  
val_auc: 0.7179  
Epoch 371/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6574 -  
accuracy: 0.6652 - auc: 0.7328 - val_loss: 0.6657 - val_accuracy: 0.6505 -  
val_auc: 0.7206  
Epoch 372/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6571 -  
accuracy: 0.6725 - auc: 0.7349 - val_loss: 0.6637 - val_accuracy: 0.6528 -  
val_auc: 0.7237  
Epoch 373/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6562 -  
accuracy: 0.6716 - auc: 0.7352 - val_loss: 0.6696 - val_accuracy: 0.6489 -  
val_auc: 0.7143  
Epoch 374/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6563 -  
accuracy: 0.6691 - auc: 0.7346 - val_loss: 0.6645 - val_accuracy: 0.6512 -  
val_auc: 0.7215  
Epoch 375/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6551 -  
accuracy: 0.6728 - auc: 0.7358 - val_loss: 0.6632 - val_accuracy: 0.6528 -  
val_auc: 0.7234  
Epoch 376/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6550 -  
accuracy: 0.6710 - auc: 0.7357 - val_loss: 0.6659 - val_accuracy: 0.6489 -  
val_auc: 0.7191  
Epoch 377/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6528 -  
accuracy: 0.6761 - auc: 0.7383 - val_loss: 0.6624 - val_accuracy: 0.6512 -  
val_auc: 0.7236  
Epoch 378/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6541 -
```

```
accuracy: 0.6711 - auc: 0.7366 - val_loss: 0.6641 - val_accuracy: 0.6535 -  
val_auc: 0.7207  
Epoch 379/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6535 -  
accuracy: 0.6711 - auc: 0.7367 - val_loss: 0.6659 - val_accuracy: 0.6505 -  
val_auc: 0.7177  
Epoch 380/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6543 -  
accuracy: 0.6731 - auc: 0.7351 - val_loss: 0.6621 - val_accuracy: 0.6520 -  
val_auc: 0.7234  
Epoch 381/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6545 -  
accuracy: 0.6716 - auc: 0.7356 - val_loss: 0.6645 - val_accuracy: 0.6481 -  
val_auc: 0.7192  
Epoch 382/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6513 -  
accuracy: 0.6759 - auc: 0.7380 - val_loss: 0.6623 - val_accuracy: 0.6481 -  
val_auc: 0.7222  
Epoch 383/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6527 -  
accuracy: 0.6700 - auc: 0.7369 - val_loss: 0.6637 - val_accuracy: 0.6474 -  
val_auc: 0.7195  
Epoch 384/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6513 -  
accuracy: 0.6731 - auc: 0.7379 - val_loss: 0.6627 - val_accuracy: 0.6497 -  
val_auc: 0.7203  
Epoch 385/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6544 -  
accuracy: 0.6688 - auc: 0.7331 - val_loss: 0.6659 - val_accuracy: 0.6512 -  
val_auc: 0.7157  
Epoch 386/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6516 -  
accuracy: 0.6705 - auc: 0.7370 - val_loss: 0.6592 - val_accuracy: 0.6528 -  
val_auc: 0.7255  
Epoch 387/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6517 -  
accuracy: 0.6690 - auc: 0.7366 - val_loss: 0.6620 - val_accuracy: 0.6528 -  
val_auc: 0.7201  
Epoch 388/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6504 -  
accuracy: 0.6729 - auc: 0.7377 - val_loss: 0.6668 - val_accuracy: 0.6466 -  
val_auc: 0.7131  
Epoch 389/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6504 -  
accuracy: 0.6729 - auc: 0.7379 - val_loss: 0.6578 - val_accuracy: 0.6512 -  
val_auc: 0.7266  
Epoch 390/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6523 -
```

```
accuracy: 0.6695 - auc: 0.7353 - val_loss: 0.6612 - val_accuracy: 0.6489 -  
val_auc: 0.7206  
Epoch 391/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6518 -  
accuracy: 0.6738 - auc: 0.7344 - val_loss: 0.6613 - val_accuracy: 0.6505 -  
val_auc: 0.7201  
Epoch 392/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6509 -  
accuracy: 0.6695 - auc: 0.7359 - val_loss: 0.6593 - val_accuracy: 0.6528 -  
val_auc: 0.7231  
Epoch 393/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6502 -  
accuracy: 0.6753 - auc: 0.7371 - val_loss: 0.6604 - val_accuracy: 0.6497 -  
val_auc: 0.7219  
Epoch 394/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6470 -  
accuracy: 0.6768 - auc: 0.7412 - val_loss: 0.6579 - val_accuracy: 0.6528 -  
val_auc: 0.7246  
Epoch 395/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6476 -  
accuracy: 0.6766 - auc: 0.7397 - val_loss: 0.6621 - val_accuracy: 0.6505 -  
val_auc: 0.7174  
Epoch 396/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6489 -  
accuracy: 0.6753 - auc: 0.7371 - val_loss: 0.6611 - val_accuracy: 0.6481 -  
val_auc: 0.7191  
Epoch 397/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6501 -  
accuracy: 0.6701 - auc: 0.7346 - val_loss: 0.6606 - val_accuracy: 0.6497 -  
val_auc: 0.7193  
Epoch 398/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6496 -  
accuracy: 0.6758 - auc: 0.7364 - val_loss: 0.6595 - val_accuracy: 0.6497 -  
val_auc: 0.7209  
Epoch 399/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6485 -  
accuracy: 0.6720 - auc: 0.7371 - val_loss: 0.6571 - val_accuracy: 0.6466 -  
val_auc: 0.7238  
Epoch 400/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6472 -  
accuracy: 0.6729 - auc: 0.7396 - val_loss: 0.6568 - val_accuracy: 0.6528 -  
val_auc: 0.7242  
Epoch 401/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6485 -  
accuracy: 0.6725 - auc: 0.7364 - val_loss: 0.6594 - val_accuracy: 0.6497 -  
val_auc: 0.7203  
Epoch 402/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6486 -
```

```
accuracy: 0.6738 - auc: 0.7361 - val_loss: 0.6562 - val_accuracy: 0.6505 -  
val_auc: 0.7241  
Epoch 403/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6479 -  
accuracy: 0.6733 - auc: 0.7379 - val_loss: 0.6548 - val_accuracy: 0.6535 -  
val_auc: 0.7261  
Epoch 404/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6468 -  
accuracy: 0.6763 - auc: 0.7378 - val_loss: 0.6550 - val_accuracy: 0.6543 -  
val_auc: 0.7258  
Epoch 405/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6467 -  
accuracy: 0.6728 - auc: 0.7383 - val_loss: 0.6559 - val_accuracy: 0.6497 -  
val_auc: 0.7243  
Epoch 406/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6471 -  
accuracy: 0.6749 - auc: 0.7378 - val_loss: 0.6570 - val_accuracy: 0.6474 -  
val_auc: 0.7224  
Epoch 407/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6465 -  
accuracy: 0.6743 - auc: 0.7380 - val_loss: 0.6560 - val_accuracy: 0.6520 -  
val_auc: 0.7228  
Epoch 408/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6448 -  
accuracy: 0.6796 - auc: 0.7404 - val_loss: 0.6562 - val_accuracy: 0.6520 -  
val_auc: 0.7227  
Epoch 409/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6461 -  
accuracy: 0.6744 - auc: 0.7394 - val_loss: 0.6558 - val_accuracy: 0.6497 -  
val_auc: 0.7232  
Epoch 410/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6466 -  
accuracy: 0.6731 - auc: 0.7369 - val_loss: 0.6564 - val_accuracy: 0.6520 -  
val_auc: 0.7217  
Epoch 411/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6454 -  
accuracy: 0.6779 - auc: 0.7397 - val_loss: 0.6531 - val_accuracy: 0.6497 -  
val_auc: 0.7261  
Epoch 412/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6450 -  
accuracy: 0.6761 - auc: 0.7379 - val_loss: 0.6523 - val_accuracy: 0.6497 -  
val_auc: 0.7272  
Epoch 413/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6453 -  
accuracy: 0.6721 - auc: 0.7381 - val_loss: 0.6535 - val_accuracy: 0.6489 -  
val_auc: 0.7252  
Epoch 414/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6452 -
```

```
accuracy: 0.6729 - auc: 0.7372 - val_loss: 0.6524 - val_accuracy: 0.6520 -  
val_auc: 0.7267  
Epoch 415/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6455 -  
accuracy: 0.6739 - auc: 0.7371 - val_loss: 0.6530 - val_accuracy: 0.6489 -  
val_auc: 0.7247  
Epoch 416/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6435 -  
accuracy: 0.6743 - auc: 0.7394 - val_loss: 0.6550 - val_accuracy: 0.6505 -  
val_auc: 0.7222  
Epoch 417/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6450 -  
accuracy: 0.6701 - auc: 0.7370 - val_loss: 0.6541 - val_accuracy: 0.6497 -  
val_auc: 0.7233  
Epoch 418/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6427 -  
accuracy: 0.6817 - auc: 0.7415 - val_loss: 0.6550 - val_accuracy: 0.6505 -  
val_auc: 0.7215  
Epoch 419/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6440 -  
accuracy: 0.6682 - auc: 0.7384 - val_loss: 0.6542 - val_accuracy: 0.6481 -  
val_auc: 0.7219  
Epoch 420/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6438 -  
accuracy: 0.6766 - auc: 0.7387 - val_loss: 0.6563 - val_accuracy: 0.6497 -  
val_auc: 0.7186  
Epoch 421/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6440 -  
accuracy: 0.6763 - auc: 0.7382 - val_loss: 0.6541 - val_accuracy: 0.6481 -  
val_auc: 0.7216  
Epoch 422/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6424 -  
accuracy: 0.6751 - auc: 0.7383 - val_loss: 0.6556 - val_accuracy: 0.6520 -  
val_auc: 0.7199  
Epoch 423/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6426 -  
accuracy: 0.6744 - auc: 0.7393 - val_loss: 0.6536 - val_accuracy: 0.6481 -  
val_auc: 0.7224  
Epoch 424/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6413 -  
accuracy: 0.6753 - auc: 0.7404 - val_loss: 0.6534 - val_accuracy: 0.6512 -  
val_auc: 0.7225  
Epoch 425/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6413 -  
accuracy: 0.6769 - auc: 0.7407 - val_loss: 0.6514 - val_accuracy: 0.6520 -  
val_auc: 0.7251  
Epoch 426/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6428 -
```

```
accuracy: 0.6756 - auc: 0.7387 - val_loss: 0.6517 - val_accuracy: 0.6520 -  
val_auc: 0.7244  
Epoch 427/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6423 -  
accuracy: 0.6736 - auc: 0.7385 - val_loss: 0.6511 - val_accuracy: 0.6481 -  
val_auc: 0.7243  
Epoch 428/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6395 -  
accuracy: 0.6820 - auc: 0.7428 - val_loss: 0.6528 - val_accuracy: 0.6481 -  
val_auc: 0.7220  
Epoch 429/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6413 -  
accuracy: 0.6787 - auc: 0.7398 - val_loss: 0.6518 - val_accuracy: 0.6497 -  
val_auc: 0.7229  
Epoch 430/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6407 -  
accuracy: 0.6761 - auc: 0.7393 - val_loss: 0.6536 - val_accuracy: 0.6497 -  
val_auc: 0.7204  
Epoch 431/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6405 -  
accuracy: 0.6781 - auc: 0.7406 - val_loss: 0.6508 - val_accuracy: 0.6481 -  
val_auc: 0.7238  
Epoch 432/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6397 -  
accuracy: 0.6746 - auc: 0.7402 - val_loss: 0.6496 - val_accuracy: 0.6481 -  
val_auc: 0.7256  
Epoch 433/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6412 -  
accuracy: 0.6779 - auc: 0.7389 - val_loss: 0.6483 - val_accuracy: 0.6512 -  
val_auc: 0.7280  
Epoch 434/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6382 -  
accuracy: 0.6733 - auc: 0.7420 - val_loss: 0.6508 - val_accuracy: 0.6505 -  
val_auc: 0.7235  
Epoch 435/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6385 -  
accuracy: 0.6777 - auc: 0.7413 - val_loss: 0.6498 - val_accuracy: 0.6528 -  
val_auc: 0.7246  
Epoch 436/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6412 -  
accuracy: 0.6764 - auc: 0.7382 - val_loss: 0.6523 - val_accuracy: 0.6512 -  
val_auc: 0.7207  
Epoch 437/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6390 -  
accuracy: 0.6758 - auc: 0.7398 - val_loss: 0.6517 - val_accuracy: 0.6505 -  
val_auc: 0.7217  
Epoch 438/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6406 -
```

```
accuracy: 0.6763 - auc: 0.7392 - val_loss: 0.6496 - val_accuracy: 0.6497 -  
val_auc: 0.7245  
Epoch 439/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6411 -  
accuracy: 0.6706 - auc: 0.7368 - val_loss: 0.6502 - val_accuracy: 0.6497 -  
val_auc: 0.7233  
Epoch 440/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6390 -  
accuracy: 0.6753 - auc: 0.7409 - val_loss: 0.6532 - val_accuracy: 0.6481 -  
val_auc: 0.7186  
Epoch 441/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6377 -  
accuracy: 0.6771 - auc: 0.7421 - val_loss: 0.6494 - val_accuracy: 0.6474 -  
val_auc: 0.7238  
Epoch 442/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6394 -  
accuracy: 0.6741 - auc: 0.7390 - val_loss: 0.6497 - val_accuracy: 0.6497 -  
val_auc: 0.7235  
Epoch 443/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6391 -  
accuracy: 0.6792 - auc: 0.7404 - val_loss: 0.6474 - val_accuracy: 0.6481 -  
val_auc: 0.7264  
Epoch 444/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6382 -  
accuracy: 0.6771 - auc: 0.7410 - val_loss: 0.6479 - val_accuracy: 0.6466 -  
val_auc: 0.7257  
Epoch 445/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6382 -  
accuracy: 0.6728 - auc: 0.7393 - val_loss: 0.6457 - val_accuracy: 0.6520 -  
val_auc: 0.7290  
Epoch 446/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6371 -  
accuracy: 0.6756 - auc: 0.7414 - val_loss: 0.6493 - val_accuracy: 0.6474 -  
val_auc: 0.7227  
Epoch 447/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6385 -  
accuracy: 0.6759 - auc: 0.7393 - val_loss: 0.6474 - val_accuracy: 0.6505 -  
val_auc: 0.7258  
Epoch 448/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6370 -  
accuracy: 0.6776 - auc: 0.7417 - val_loss: 0.6471 - val_accuracy: 0.6466 -  
val_auc: 0.7256  
Epoch 449/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6361 -  
accuracy: 0.6738 - auc: 0.7421 - val_loss: 0.6479 - val_accuracy: 0.6497 -  
val_auc: 0.7247  
Epoch 450/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6374 -
```

```
accuracy: 0.6741 - auc: 0.7398 - val_loss: 0.6447 - val_accuracy: 0.6535 -  
val_auc: 0.7290  
Epoch 451/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6365 -  
accuracy: 0.6782 - auc: 0.7420 - val_loss: 0.6449 - val_accuracy: 0.6535 -  
val_auc: 0.7288  
Epoch 452/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6363 -  
accuracy: 0.6787 - auc: 0.7412 - val_loss: 0.6462 - val_accuracy: 0.6497 -  
val_auc: 0.7261  
Epoch 453/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6385 -  
accuracy: 0.6756 - auc: 0.7393 - val_loss: 0.6451 - val_accuracy: 0.6512 -  
val_auc: 0.7279  
Epoch 454/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6368 -  
accuracy: 0.6796 - auc: 0.7395 - val_loss: 0.6512 - val_accuracy: 0.6458 -  
val_auc: 0.7187  
Epoch 455/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6351 -  
accuracy: 0.6801 - auc: 0.7433 - val_loss: 0.6499 - val_accuracy: 0.6489 -  
val_auc: 0.7205  
Epoch 456/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6356 -  
accuracy: 0.6733 - auc: 0.7410 - val_loss: 0.6465 - val_accuracy: 0.6497 -  
val_auc: 0.7250  
Epoch 457/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6353 -  
accuracy: 0.6776 - auc: 0.7414 - val_loss: 0.6461 - val_accuracy: 0.6481 -  
val_auc: 0.7251  
Epoch 458/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6352 -  
accuracy: 0.6802 - auc: 0.7414 - val_loss: 0.6474 - val_accuracy: 0.6497 -  
val_auc: 0.7234  
Epoch 459/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6347 -  
accuracy: 0.6718 - auc: 0.7424 - val_loss: 0.6489 - val_accuracy: 0.6489 -  
val_auc: 0.7205  
Epoch 460/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6361 -  
accuracy: 0.6782 - auc: 0.7401 - val_loss: 0.6443 - val_accuracy: 0.6512 -  
val_auc: 0.7277  
Epoch 461/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6350 -  
accuracy: 0.6784 - auc: 0.7424 - val_loss: 0.6435 - val_accuracy: 0.6551 -  
val_auc: 0.7288  
Epoch 462/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6344 -
```

```
accuracy: 0.6739 - auc: 0.7419 - val_loss: 0.6451 - val_accuracy: 0.6458 -  
val_auc: 0.7253  
Epoch 463/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6341 -  
accuracy: 0.6772 - auc: 0.7425 - val_loss: 0.6462 - val_accuracy: 0.6505 -  
val_auc: 0.7239  
Epoch 464/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6347 -  
accuracy: 0.6749 - auc: 0.7414 - val_loss: 0.6439 - val_accuracy: 0.6512 -  
val_auc: 0.7271  
Epoch 465/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6320 -  
accuracy: 0.6792 - auc: 0.7444 - val_loss: 0.6417 - val_accuracy: 0.6505 -  
val_auc: 0.7296  
Epoch 466/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6354 -  
accuracy: 0.6771 - auc: 0.7406 - val_loss: 0.6450 - val_accuracy: 0.6505 -  
val_auc: 0.7255  
Epoch 467/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6340 -  
accuracy: 0.6781 - auc: 0.7418 - val_loss: 0.6419 - val_accuracy: 0.6543 -  
val_auc: 0.7296  
Epoch 468/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6327 -  
accuracy: 0.6751 - auc: 0.7436 - val_loss: 0.6423 - val_accuracy: 0.6512 -  
val_auc: 0.7284  
Epoch 469/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6345 -  
accuracy: 0.6766 - auc: 0.7405 - val_loss: 0.6446 - val_accuracy: 0.6458 -  
val_auc: 0.7253  
Epoch 470/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6321 -  
accuracy: 0.6772 - auc: 0.7448 - val_loss: 0.6435 - val_accuracy: 0.6528 -  
val_auc: 0.7270  
Epoch 471/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6325 -  
accuracy: 0.6787 - auc: 0.7425 - val_loss: 0.6428 - val_accuracy: 0.6489 -  
val_auc: 0.7275  
Epoch 472/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6340 -  
accuracy: 0.6779 - auc: 0.7416 - val_loss: 0.6446 - val_accuracy: 0.6535 -  
val_auc: 0.7252  
Epoch 473/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6312 -  
accuracy: 0.6832 - auc: 0.7452 - val_loss: 0.6425 - val_accuracy: 0.6543 -  
val_auc: 0.7281  
Epoch 474/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6354 -
```

```
accuracy: 0.6744 - auc: 0.7387 - val_loss: 0.6450 - val_accuracy: 0.6474 -  
val_auc: 0.7242  
Epoch 475/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6318 -  
accuracy: 0.6827 - auc: 0.7434 - val_loss: 0.6443 - val_accuracy: 0.6497 -  
val_auc: 0.7256  
Epoch 476/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6323 -  
accuracy: 0.6754 - auc: 0.7422 - val_loss: 0.6442 - val_accuracy: 0.6481 -  
val_auc: 0.7249  
Epoch 477/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6315 -  
accuracy: 0.6799 - auc: 0.7424 - val_loss: 0.6438 - val_accuracy: 0.6466 -  
val_auc: 0.7255  
Epoch 478/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6311 -  
accuracy: 0.6791 - auc: 0.7438 - val_loss: 0.6432 - val_accuracy: 0.6489 -  
val_auc: 0.7262  
Epoch 479/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6305 -  
accuracy: 0.6784 - auc: 0.7451 - val_loss: 0.6433 - val_accuracy: 0.6505 -  
val_auc: 0.7259  
Epoch 480/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6304 -  
accuracy: 0.6779 - auc: 0.7449 - val_loss: 0.6437 - val_accuracy: 0.6543 -  
val_auc: 0.7252  
Epoch 481/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6318 -  
accuracy: 0.6784 - auc: 0.7430 - val_loss: 0.6432 - val_accuracy: 0.6551 -  
val_auc: 0.7258  
Epoch 482/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6309 -  
accuracy: 0.6834 - auc: 0.7439 - val_loss: 0.6414 - val_accuracy: 0.6520 -  
val_auc: 0.7280  
Epoch 483/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6317 -  
accuracy: 0.6746 - auc: 0.7416 - val_loss: 0.6441 - val_accuracy: 0.6458 -  
val_auc: 0.7240  
Epoch 484/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6316 -  
accuracy: 0.6741 - auc: 0.7426 - val_loss: 0.6461 - val_accuracy: 0.6474 -  
val_auc: 0.7212  
Epoch 485/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6302 -  
accuracy: 0.6771 - auc: 0.7444 - val_loss: 0.6393 - val_accuracy: 0.6505 -  
val_auc: 0.7302  
Epoch 486/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6310 -
```

```
accuracy: 0.6777 - auc: 0.7427 - val_loss: 0.6422 - val_accuracy: 0.6497 -  
val_auc: 0.7260  
Epoch 487/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6317 -  
accuracy: 0.6739 - auc: 0.7419 - val_loss: 0.6397 - val_accuracy: 0.6520 -  
val_auc: 0.7297  
Epoch 488/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6296 -  
accuracy: 0.6771 - auc: 0.7444 - val_loss: 0.6420 - val_accuracy: 0.6481 -  
val_auc: 0.7260  
Epoch 489/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6314 -  
accuracy: 0.6774 - auc: 0.7420 - val_loss: 0.6399 - val_accuracy: 0.6551 -  
val_auc: 0.7293  
Epoch 490/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6297 -  
accuracy: 0.6787 - auc: 0.7434 - val_loss: 0.6430 - val_accuracy: 0.6435 -  
val_auc: 0.7243  
Epoch 491/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6309 -  
accuracy: 0.6753 - auc: 0.7426 - val_loss: 0.6394 - val_accuracy: 0.6535 -  
val_auc: 0.7295  
Epoch 492/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6289 -  
accuracy: 0.6825 - auc: 0.7457 - val_loss: 0.6439 - val_accuracy: 0.6481 -  
val_auc: 0.7225  
Epoch 493/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6299 -  
accuracy: 0.6751 - auc: 0.7434 - val_loss: 0.6434 - val_accuracy: 0.6474 -  
val_auc: 0.7232  
Epoch 494/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6294 -  
accuracy: 0.6779 - auc: 0.7438 - val_loss: 0.6404 - val_accuracy: 0.6497 -  
val_auc: 0.7273  
Epoch 495/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6300 -  
accuracy: 0.6766 - auc: 0.7427 - val_loss: 0.6359 - val_accuracy: 0.6590 -  
val_auc: 0.7333  
Epoch 496/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6303 -  
accuracy: 0.6751 - auc: 0.7422 - val_loss: 0.6420 - val_accuracy: 0.6489 -  
val_auc: 0.7249  
Epoch 497/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6284 -  
accuracy: 0.6794 - auc: 0.7452 - val_loss: 0.6404 - val_accuracy: 0.6458 -  
val_auc: 0.7264  
Epoch 498/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6280 -
```

```
accuracy: 0.6781 - auc: 0.7459 - val_loss: 0.6398 - val_accuracy: 0.6535 -  
val_auc: 0.7280  
Epoch 499/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6298 -  
accuracy: 0.6806 - auc: 0.7435 - val_loss: 0.6410 - val_accuracy: 0.6528 -  
val_auc: 0.7261  
Epoch 500/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6294 -  
accuracy: 0.6822 - auc: 0.7429 - val_loss: 0.6373 - val_accuracy: 0.6559 -  
val_auc: 0.7311  
Epoch 501/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6284 -  
accuracy: 0.6754 - auc: 0.7434 - val_loss: 0.6398 - val_accuracy: 0.6489 -  
val_auc: 0.7274  
Epoch 502/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6269 -  
accuracy: 0.6802 - auc: 0.7459 - val_loss: 0.6360 - val_accuracy: 0.6597 -  
val_auc: 0.7327  
Epoch 503/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6263 -  
accuracy: 0.6794 - auc: 0.7473 - val_loss: 0.6413 - val_accuracy: 0.6458 -  
val_auc: 0.7247  
Epoch 504/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6281 -  
accuracy: 0.6809 - auc: 0.7442 - val_loss: 0.6373 - val_accuracy: 0.6574 -  
val_auc: 0.7305  
Epoch 505/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6277 -  
accuracy: 0.6776 - auc: 0.7450 - val_loss: 0.6373 - val_accuracy: 0.6551 -  
val_auc: 0.7306  
Epoch 506/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6266 -  
accuracy: 0.6758 - auc: 0.7455 - val_loss: 0.6393 - val_accuracy: 0.6559 -  
val_auc: 0.7272  
Epoch 507/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6271 -  
accuracy: 0.6794 - auc: 0.7454 - val_loss: 0.6411 - val_accuracy: 0.6481 -  
val_auc: 0.7242  
Epoch 508/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6274 -  
accuracy: 0.6802 - auc: 0.7453 - val_loss: 0.6357 - val_accuracy: 0.6574 -  
val_auc: 0.7323  
Epoch 509/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6287 -  
accuracy: 0.6758 - auc: 0.7422 - val_loss: 0.6400 - val_accuracy: 0.6489 -  
val_auc: 0.7258  
Epoch 510/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6268 -
```

```
accuracy: 0.6787 - auc: 0.7455 - val_loss: 0.6382 - val_accuracy: 0.6520 -  
val_auc: 0.7284  
Epoch 511/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6279 -  
accuracy: 0.6774 - auc: 0.7439 - val_loss: 0.6391 - val_accuracy: 0.6489 -  
val_auc: 0.7269  
Epoch 512/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6276 -  
accuracy: 0.6825 - auc: 0.7444 - val_loss: 0.6376 - val_accuracy: 0.6520 -  
val_auc: 0.7290  
Epoch 513/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6266 -  
accuracy: 0.6824 - auc: 0.7462 - val_loss: 0.6338 - val_accuracy: 0.6613 -  
val_auc: 0.7344  
Epoch 514/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6276 -  
accuracy: 0.6748 - auc: 0.7441 - val_loss: 0.6334 - val_accuracy: 0.6566 -  
val_auc: 0.7346  
Epoch 515/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6264 -  
accuracy: 0.6758 - auc: 0.7448 - val_loss: 0.6365 - val_accuracy: 0.6535 -  
val_auc: 0.7300  
Epoch 516/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6267 -  
accuracy: 0.6792 - auc: 0.7461 - val_loss: 0.6347 - val_accuracy: 0.6605 -  
val_auc: 0.7329  
Epoch 517/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6250 -  
accuracy: 0.6834 - auc: 0.7471 - val_loss: 0.6376 - val_accuracy: 0.6466 -  
val_auc: 0.7280  
Epoch 518/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6266 -  
accuracy: 0.6820 - auc: 0.7449 - val_loss: 0.6350 - val_accuracy: 0.6535 -  
val_auc: 0.7315  
Epoch 519/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6262 -  
accuracy: 0.6812 - auc: 0.7457 - val_loss: 0.6360 - val_accuracy: 0.6528 -  
val_auc: 0.7301  
Epoch 520/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6250 -  
accuracy: 0.6761 - auc: 0.7461 - val_loss: 0.6359 - val_accuracy: 0.6582 -  
val_auc: 0.7308  
Epoch 521/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6265 -  
accuracy: 0.6809 - auc: 0.7457 - val_loss: 0.6340 - val_accuracy: 0.6551 -  
val_auc: 0.7326  
Epoch 522/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6261 -
```

```
accuracy: 0.6815 - auc: 0.7443 - val_loss: 0.6356 - val_accuracy: 0.6528 -  
val_auc: 0.7307  
Epoch 523/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6244 -  
accuracy: 0.6842 - auc: 0.7471 - val_loss: 0.6362 - val_accuracy: 0.6543 -  
val_auc: 0.7300  
Epoch 524/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6258 -  
accuracy: 0.6804 - auc: 0.7454 - val_loss: 0.6340 - val_accuracy: 0.6566 -  
val_auc: 0.7325  
Epoch 525/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6269 -  
accuracy: 0.6772 - auc: 0.7453 - val_loss: 0.6333 - val_accuracy: 0.6582 -  
val_auc: 0.7331  
Epoch 526/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6251 -  
accuracy: 0.6799 - auc: 0.7454 - val_loss: 0.6369 - val_accuracy: 0.6505 -  
val_auc: 0.7282  
Epoch 527/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6238 -  
accuracy: 0.6812 - auc: 0.7462 - val_loss: 0.6341 - val_accuracy: 0.6566 -  
val_auc: 0.7321  
Epoch 528/1000  
189/189 [=====] - 4s 21ms/step - loss: 0.6230 -  
accuracy: 0.6802 - auc: 0.7490 - val_loss: 0.6330 - val_accuracy: 0.6566 -  
val_auc: 0.7333  
Epoch 529/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6231 -  
accuracy: 0.6898 - auc: 0.7493 - val_loss: 0.6325 - val_accuracy: 0.6590 -  
val_auc: 0.7341  
Epoch 530/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6233 -  
accuracy: 0.6786 - auc: 0.7476 - val_loss: 0.6334 - val_accuracy: 0.6582 -  
val_auc: 0.7331  
Epoch 531/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6227 -  
accuracy: 0.6837 - auc: 0.7480 - val_loss: 0.6357 - val_accuracy: 0.6535 -  
val_auc: 0.7293  
Epoch 532/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6248 -  
accuracy: 0.6799 - auc: 0.7466 - val_loss: 0.6332 - val_accuracy: 0.6543 -  
val_auc: 0.7327  
Epoch 533/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6232 -  
accuracy: 0.6799 - auc: 0.7470 - val_loss: 0.6366 - val_accuracy: 0.6497 -  
val_auc: 0.7278  
Epoch 534/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6256 -
```

```
accuracy: 0.6743 - auc: 0.7445 - val_loss: 0.6342 - val_accuracy: 0.6574 -  
val_auc: 0.7315  
Epoch 535/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6236 -  
accuracy: 0.6835 - auc: 0.7463 - val_loss: 0.6338 - val_accuracy: 0.6559 -  
val_auc: 0.7319  
Epoch 536/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6245 -  
accuracy: 0.6796 - auc: 0.7458 - val_loss: 0.6358 - val_accuracy: 0.6528 -  
val_auc: 0.7291  
Epoch 537/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6231 -  
accuracy: 0.6820 - auc: 0.7465 - val_loss: 0.6337 - val_accuracy: 0.6566 -  
val_auc: 0.7317  
Epoch 538/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6233 -  
accuracy: 0.6796 - auc: 0.7470 - val_loss: 0.6387 - val_accuracy: 0.6497 -  
val_auc: 0.7240  
Epoch 539/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6228 -  
accuracy: 0.6769 - auc: 0.7464 - val_loss: 0.6344 - val_accuracy: 0.6543 -  
val_auc: 0.7308  
Epoch 540/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6225 -  
accuracy: 0.6844 - auc: 0.7484 - val_loss: 0.6331 - val_accuracy: 0.6559 -  
val_auc: 0.7323  
Epoch 541/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6242 -  
accuracy: 0.6789 - auc: 0.7451 - val_loss: 0.6359 - val_accuracy: 0.6497 -  
val_auc: 0.7276  
Epoch 542/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6226 -  
accuracy: 0.6830 - auc: 0.7480 - val_loss: 0.6334 - val_accuracy: 0.6535 -  
val_auc: 0.7313  
Epoch 543/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6220 -  
accuracy: 0.6817 - auc: 0.7485 - val_loss: 0.6343 - val_accuracy: 0.6512 -  
val_auc: 0.7293  
Epoch 544/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6227 -  
accuracy: 0.6806 - auc: 0.7461 - val_loss: 0.6320 - val_accuracy: 0.6582 -  
val_auc: 0.7333  
Epoch 545/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6214 -  
accuracy: 0.6769 - auc: 0.7481 - val_loss: 0.6344 - val_accuracy: 0.6559 -  
val_auc: 0.7300  
Epoch 546/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6236 -
```

```
accuracy: 0.6832 - auc: 0.7458 - val_loss: 0.6313 - val_accuracy: 0.6574 -  
val_auc: 0.7341  
Epoch 547/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6242 -  
accuracy: 0.6784 - auc: 0.7461 - val_loss: 0.6343 - val_accuracy: 0.6505 -  
val_auc: 0.7290  
Epoch 548/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6219 -  
accuracy: 0.6811 - auc: 0.7469 - val_loss: 0.6327 - val_accuracy: 0.6551 -  
val_auc: 0.7317  
Epoch 549/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6240 -  
accuracy: 0.6807 - auc: 0.7455 - val_loss: 0.6319 - val_accuracy: 0.6520 -  
val_auc: 0.7325  
Epoch 550/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6234 -  
accuracy: 0.6768 - auc: 0.7453 - val_loss: 0.6338 - val_accuracy: 0.6505 -  
val_auc: 0.7304  
Epoch 551/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6224 -  
accuracy: 0.6837 - auc: 0.7477 - val_loss: 0.6342 - val_accuracy: 0.6458 -  
val_auc: 0.7290  
Epoch 552/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6227 -  
accuracy: 0.6806 - auc: 0.7460 - val_loss: 0.6336 - val_accuracy: 0.6520 -  
val_auc: 0.7304  
Epoch 553/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6212 -  
accuracy: 0.6844 - auc: 0.7491 - val_loss: 0.6305 - val_accuracy: 0.6590 -  
val_auc: 0.7348  
Epoch 554/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6225 -  
accuracy: 0.6840 - auc: 0.7468 - val_loss: 0.6332 - val_accuracy: 0.6528 -  
val_auc: 0.7306  
Epoch 555/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6221 -  
accuracy: 0.6776 - auc: 0.7465 - val_loss: 0.6340 - val_accuracy: 0.6489 -  
val_auc: 0.7286  
Epoch 556/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6214 -  
accuracy: 0.6847 - auc: 0.7472 - val_loss: 0.6305 - val_accuracy: 0.6535 -  
val_auc: 0.7338  
Epoch 557/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6203 -  
accuracy: 0.6829 - auc: 0.7492 - val_loss: 0.6323 - val_accuracy: 0.6543 -  
val_auc: 0.7315  
Epoch 558/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6207 -
```

```
accuracy: 0.6837 - auc: 0.7490 - val_loss: 0.6315 - val_accuracy: 0.6566 -  
val_auc: 0.7327  
Epoch 559/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6217 -  
accuracy: 0.6834 - auc: 0.7473 - val_loss: 0.6310 - val_accuracy: 0.6543 -  
val_auc: 0.7328  
Epoch 560/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6217 -  
accuracy: 0.6802 - auc: 0.7473 - val_loss: 0.6324 - val_accuracy: 0.6528 -  
val_auc: 0.7308  
Epoch 561/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6203 -  
accuracy: 0.6845 - auc: 0.7483 - val_loss: 0.6323 - val_accuracy: 0.6543 -  
val_auc: 0.7312  
Epoch 562/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6206 -  
accuracy: 0.6844 - auc: 0.7491 - val_loss: 0.6348 - val_accuracy: 0.6559 -  
val_auc: 0.7282  
Epoch 563/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6208 -  
accuracy: 0.6834 - auc: 0.7487 - val_loss: 0.6305 - val_accuracy: 0.6559 -  
val_auc: 0.7341  
Epoch 564/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6213 -  
accuracy: 0.6776 - auc: 0.7475 - val_loss: 0.6322 - val_accuracy: 0.6559 -  
val_auc: 0.7310  
Epoch 565/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6204 -  
accuracy: 0.6863 - auc: 0.7494 - val_loss: 0.6303 - val_accuracy: 0.6582 -  
val_auc: 0.7338  
Epoch 566/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6207 -  
accuracy: 0.6815 - auc: 0.7470 - val_loss: 0.6311 - val_accuracy: 0.6528 -  
val_auc: 0.7329  
Epoch 567/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6206 -  
accuracy: 0.6870 - auc: 0.7499 - val_loss: 0.6348 - val_accuracy: 0.6489 -  
val_auc: 0.7272  
Epoch 568/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6202 -  
accuracy: 0.6860 - auc: 0.7491 - val_loss: 0.6329 - val_accuracy: 0.6528 -  
val_auc: 0.7299  
Epoch 569/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6199 -  
accuracy: 0.6845 - auc: 0.7480 - val_loss: 0.6367 - val_accuracy: 0.6528 -  
val_auc: 0.7239  
Epoch 570/1000  
189/189 [=====] - 4s 21ms/step - loss: 0.6192 -
```

```
accuracy: 0.6827 - auc: 0.7497 - val_loss: 0.6303 - val_accuracy: 0.6574 -  
val_auc: 0.7337  
Epoch 571/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6197 -  
accuracy: 0.6820 - auc: 0.7489 - val_loss: 0.6297 - val_accuracy: 0.6574 -  
val_auc: 0.7337  
Epoch 572/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6199 -  
accuracy: 0.6802 - auc: 0.7468 - val_loss: 0.6321 - val_accuracy: 0.6559 -  
val_auc: 0.7312  
Epoch 573/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6189 -  
accuracy: 0.6830 - auc: 0.7502 - val_loss: 0.6302 - val_accuracy: 0.6559 -  
val_auc: 0.7331  
Epoch 574/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6212 -  
accuracy: 0.6804 - auc: 0.7476 - val_loss: 0.6297 - val_accuracy: 0.6528 -  
val_auc: 0.7330  
Epoch 575/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6210 -  
accuracy: 0.6799 - auc: 0.7456 - val_loss: 0.6335 - val_accuracy: 0.6505 -  
val_auc: 0.7283  
Epoch 576/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6199 -  
accuracy: 0.6840 - auc: 0.7488 - val_loss: 0.6343 - val_accuracy: 0.6481 -  
val_auc: 0.7268  
Epoch 577/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6202 -  
accuracy: 0.6832 - auc: 0.7478 - val_loss: 0.6271 - val_accuracy: 0.6597 -  
val_auc: 0.7367  
Epoch 578/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6185 -  
accuracy: 0.6860 - auc: 0.7493 - val_loss: 0.6335 - val_accuracy: 0.6497 -  
val_auc: 0.7284  
Epoch 579/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6186 -  
accuracy: 0.6822 - auc: 0.7497 - val_loss: 0.6300 - val_accuracy: 0.6566 -  
val_auc: 0.7333  
Epoch 580/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6197 -  
accuracy: 0.6835 - auc: 0.7490 - val_loss: 0.6308 - val_accuracy: 0.6543 -  
val_auc: 0.7316  
Epoch 581/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6181 -  
accuracy: 0.6860 - auc: 0.7504 - val_loss: 0.6304 - val_accuracy: 0.6566 -  
val_auc: 0.7323  
Epoch 582/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6195 -
```

```
accuracy: 0.6814 - auc: 0.7470 - val_loss: 0.6288 - val_accuracy: 0.6590 -  
val_auc: 0.7344  
Epoch 583/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6198 -  
accuracy: 0.6845 - auc: 0.7475 - val_loss: 0.6284 - val_accuracy: 0.6582 -  
val_auc: 0.7347  
Epoch 584/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6182 -  
accuracy: 0.6839 - auc: 0.7497 - val_loss: 0.6282 - val_accuracy: 0.6597 -  
val_auc: 0.7350  
Epoch 585/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6174 -  
accuracy: 0.6822 - auc: 0.7521 - val_loss: 0.6294 - val_accuracy: 0.6574 -  
val_auc: 0.7334  
Epoch 586/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6186 -  
accuracy: 0.6829 - auc: 0.7488 - val_loss: 0.6301 - val_accuracy: 0.6574 -  
val_auc: 0.7322  
Epoch 587/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6183 -  
accuracy: 0.6854 - auc: 0.7492 - val_loss: 0.6298 - val_accuracy: 0.6582 -  
val_auc: 0.7317  
Epoch 588/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6176 -  
accuracy: 0.6817 - auc: 0.7495 - val_loss: 0.6272 - val_accuracy: 0.6590 -  
val_auc: 0.7361  
Epoch 589/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6182 -  
accuracy: 0.6875 - auc: 0.7492 - val_loss: 0.6320 - val_accuracy: 0.6551 -  
val_auc: 0.7294  
Epoch 590/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6179 -  
accuracy: 0.6829 - auc: 0.7503 - val_loss: 0.6275 - val_accuracy: 0.6590 -  
val_auc: 0.7359  
Epoch 591/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6168 -  
accuracy: 0.6865 - auc: 0.7506 - val_loss: 0.6316 - val_accuracy: 0.6505 -  
val_auc: 0.7296  
Epoch 592/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6174 -  
accuracy: 0.6837 - auc: 0.7503 - val_loss: 0.6261 - val_accuracy: 0.6566 -  
val_auc: 0.7372  
Epoch 593/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6177 -  
accuracy: 0.6845 - auc: 0.7502 - val_loss: 0.6297 - val_accuracy: 0.6559 -  
val_auc: 0.7319  
Epoch 594/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6179 -
```

```
accuracy: 0.6822 - auc: 0.7495 - val_loss: 0.6287 - val_accuracy: 0.6566 -
val_auc: 0.7328
Epoch 595/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6175 -
accuracy: 0.6811 - auc: 0.7498 - val_loss: 0.6273 - val_accuracy: 0.6597 -
val_auc: 0.7356
Epoch 596/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6167 -
accuracy: 0.6877 - auc: 0.7514 - val_loss: 0.6278 - val_accuracy: 0.6597 -
val_auc: 0.7347
Epoch 597/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6183 -
accuracy: 0.6829 - auc: 0.7495 - val_loss: 0.6265 - val_accuracy: 0.6590 -
val_auc: 0.7366
Epoch 598/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6171 -
accuracy: 0.6819 - auc: 0.7507 - val_loss: 0.6282 - val_accuracy: 0.6582 -
val_auc: 0.7340
Epoch 599/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6166 -
accuracy: 0.6844 - auc: 0.7505 - val_loss: 0.6282 - val_accuracy: 0.6559 -
val_auc: 0.7334
Epoch 600/1000
189/189 [=====] - 4s 19ms/step - loss: 0.6152 -
accuracy: 0.6855 - auc: 0.7522 - val_loss: 0.6308 - val_accuracy: 0.6566 -
val_auc: 0.7300
Epoch 601/1000
189/189 [=====] - 4s 19ms/step - loss: 0.6158 -
accuracy: 0.6850 - auc: 0.7523 - val_loss: 0.6294 - val_accuracy: 0.6559 -
val_auc: 0.7319
Epoch 602/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6165 -
accuracy: 0.6835 - auc: 0.7506 - val_loss: 0.6256 - val_accuracy: 0.6590 -
val_auc: 0.7373
Epoch 603/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6169 -
accuracy: 0.6849 - auc: 0.7496 - val_loss: 0.6271 - val_accuracy: 0.6590 -
val_auc: 0.7360
Epoch 604/1000
189/189 [=====] - 4s 19ms/step - loss: 0.6163 -
accuracy: 0.6873 - auc: 0.7512 - val_loss: 0.6293 - val_accuracy: 0.6543 -
val_auc: 0.7322
Epoch 605/1000
189/189 [=====] - 4s 19ms/step - loss: 0.6152 -
accuracy: 0.6898 - auc: 0.7538 - val_loss: 0.6290 - val_accuracy: 0.6559 -
val_auc: 0.7322
Epoch 606/1000
189/189 [=====] - 4s 19ms/step - loss: 0.6145 -
```

```
accuracy: 0.6875 - auc: 0.7532 - val_loss: 0.6273 - val_accuracy: 0.6574 -  
val_auc: 0.7346  
Epoch 607/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6172 -  
accuracy: 0.6832 - auc: 0.7495 - val_loss: 0.6239 - val_accuracy: 0.6590 -  
val_auc: 0.7392  
Epoch 608/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6161 -  
accuracy: 0.6870 - auc: 0.7506 - val_loss: 0.6314 - val_accuracy: 0.6574 -  
val_auc: 0.7287  
Epoch 609/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6166 -  
accuracy: 0.6815 - auc: 0.7510 - val_loss: 0.6295 - val_accuracy: 0.6551 -  
val_auc: 0.7314  
Epoch 610/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6177 -  
accuracy: 0.6825 - auc: 0.7487 - val_loss: 0.6301 - val_accuracy: 0.6535 -  
val_auc: 0.7302  
Epoch 611/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6162 -  
accuracy: 0.6858 - auc: 0.7503 - val_loss: 0.6250 - val_accuracy: 0.6559 -  
val_auc: 0.7372  
Epoch 612/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6161 -  
accuracy: 0.6903 - auc: 0.7507 - val_loss: 0.6276 - val_accuracy: 0.6574 -  
val_auc: 0.7341  
Epoch 613/1000  
189/189 [=====] - 4s 21ms/step - loss: 0.6169 -  
accuracy: 0.6824 - auc: 0.7504 - val_loss: 0.6274 - val_accuracy: 0.6559 -  
val_auc: 0.7334  
Epoch 614/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6181 -  
accuracy: 0.6822 - auc: 0.7477 - val_loss: 0.6283 - val_accuracy: 0.6551 -  
val_auc: 0.7327  
Epoch 615/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6152 -  
accuracy: 0.6867 - auc: 0.7520 - val_loss: 0.6267 - val_accuracy: 0.6566 -  
val_auc: 0.7351  
Epoch 616/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6156 -  
accuracy: 0.6860 - auc: 0.7522 - val_loss: 0.6241 - val_accuracy: 0.6582 -  
val_auc: 0.7389  
Epoch 617/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6150 -  
accuracy: 0.6840 - auc: 0.7522 - val_loss: 0.6276 - val_accuracy: 0.6543 -  
val_auc: 0.7334  
Epoch 618/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6151 -
```

```
accuracy: 0.6825 - auc: 0.7512 - val_loss: 0.6268 - val_accuracy: 0.6582 -  
val_auc: 0.7347  
Epoch 619/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6146 -  
accuracy: 0.6873 - auc: 0.7520 - val_loss: 0.6284 - val_accuracy: 0.6528 -  
val_auc: 0.7324  
Epoch 620/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6147 -  
accuracy: 0.6850 - auc: 0.7514 - val_loss: 0.6252 - val_accuracy: 0.6574 -  
val_auc: 0.7364  
Epoch 621/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6127 -  
accuracy: 0.6837 - auc: 0.7548 - val_loss: 0.6275 - val_accuracy: 0.6551 -  
val_auc: 0.7332  
Epoch 622/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6174 -  
accuracy: 0.6830 - auc: 0.7489 - val_loss: 0.6293 - val_accuracy: 0.6535 -  
val_auc: 0.7308  
Epoch 623/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6148 -  
accuracy: 0.6882 - auc: 0.7527 - val_loss: 0.6251 - val_accuracy: 0.6605 -  
val_auc: 0.7370  
Epoch 624/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6151 -  
accuracy: 0.6824 - auc: 0.7504 - val_loss: 0.6259 - val_accuracy: 0.6574 -  
val_auc: 0.7354  
Epoch 625/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6163 -  
accuracy: 0.6862 - auc: 0.7505 - val_loss: 0.6263 - val_accuracy: 0.6628 -  
val_auc: 0.7354  
Epoch 626/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6153 -  
accuracy: 0.6883 - auc: 0.7524 - val_loss: 0.6258 - val_accuracy: 0.6590 -  
val_auc: 0.7352  
Epoch 627/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6159 -  
accuracy: 0.6844 - auc: 0.7505 - val_loss: 0.6256 - val_accuracy: 0.6574 -  
val_auc: 0.7360  
Epoch 628/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6154 -  
accuracy: 0.6896 - auc: 0.7518 - val_loss: 0.6294 - val_accuracy: 0.6551 -  
val_auc: 0.7302  
Epoch 629/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6142 -  
accuracy: 0.6827 - auc: 0.7507 - val_loss: 0.6255 - val_accuracy: 0.6590 -  
val_auc: 0.7358  
Epoch 630/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6152 -
```

```
accuracy: 0.6877 - auc: 0.7527 - val_loss: 0.6233 - val_accuracy: 0.6551 -  
val_auc: 0.7386  
Epoch 631/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6156 -  
accuracy: 0.6835 - auc: 0.7500 - val_loss: 0.6246 - val_accuracy: 0.6613 -  
val_auc: 0.7378  
Epoch 632/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6153 -  
accuracy: 0.6860 - auc: 0.7507 - val_loss: 0.6282 - val_accuracy: 0.6590 -  
val_auc: 0.7313  
Epoch 633/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6148 -  
accuracy: 0.6844 - auc: 0.7518 - val_loss: 0.6249 - val_accuracy: 0.6597 -  
val_auc: 0.7367  
Epoch 634/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6147 -  
accuracy: 0.6847 - auc: 0.7513 - val_loss: 0.6291 - val_accuracy: 0.6582 -  
val_auc: 0.7305  
Epoch 635/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6147 -  
accuracy: 0.6905 - auc: 0.7512 - val_loss: 0.6258 - val_accuracy: 0.6590 -  
val_auc: 0.7355  
Epoch 636/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6149 -  
accuracy: 0.6863 - auc: 0.7514 - val_loss: 0.6258 - val_accuracy: 0.6597 -  
val_auc: 0.7349  
Epoch 637/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6152 -  
accuracy: 0.6835 - auc: 0.7509 - val_loss: 0.6235 - val_accuracy: 0.6566 -  
val_auc: 0.7380  
Epoch 638/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6133 -  
accuracy: 0.6872 - auc: 0.7528 - val_loss: 0.6278 - val_accuracy: 0.6543 -  
val_auc: 0.7323  
Epoch 639/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6159 -  
accuracy: 0.6862 - auc: 0.7507 - val_loss: 0.6258 - val_accuracy: 0.6559 -  
val_auc: 0.7346  
Epoch 640/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6152 -  
accuracy: 0.6863 - auc: 0.7512 - val_loss: 0.6236 - val_accuracy: 0.6597 -  
val_auc: 0.7378  
Epoch 641/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6137 -  
accuracy: 0.6854 - auc: 0.7518 - val_loss: 0.6237 - val_accuracy: 0.6559 -  
val_auc: 0.7372  
Epoch 642/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6155 -
```

```
accuracy: 0.6845 - auc: 0.7507 - val_loss: 0.6266 - val_accuracy: 0.6574 -  
val_auc: 0.7340  
Epoch 643/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6130 -  
accuracy: 0.6865 - auc: 0.7536 - val_loss: 0.6264 - val_accuracy: 0.6590 -  
val_auc: 0.7341  
Epoch 644/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6139 -  
accuracy: 0.6877 - auc: 0.7521 - val_loss: 0.6246 - val_accuracy: 0.6605 -  
val_auc: 0.7362  
Epoch 645/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6116 -  
accuracy: 0.6887 - auc: 0.7564 - val_loss: 0.6257 - val_accuracy: 0.6590 -  
val_auc: 0.7351  
Epoch 646/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6131 -  
accuracy: 0.6883 - auc: 0.7534 - val_loss: 0.6245 - val_accuracy: 0.6597 -  
val_auc: 0.7364  
Epoch 647/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6126 -  
accuracy: 0.6893 - auc: 0.7538 - val_loss: 0.6265 - val_accuracy: 0.6582 -  
val_auc: 0.7337  
Epoch 648/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6132 -  
accuracy: 0.6870 - auc: 0.7528 - val_loss: 0.6241 - val_accuracy: 0.6590 -  
val_auc: 0.7364  
Epoch 649/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6127 -  
accuracy: 0.6895 - auc: 0.7527 - val_loss: 0.6245 - val_accuracy: 0.6590 -  
val_auc: 0.7360  
Epoch 650/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6129 -  
accuracy: 0.6873 - auc: 0.7536 - val_loss: 0.6250 - val_accuracy: 0.6636 -  
val_auc: 0.7358  
Epoch 651/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6133 -  
accuracy: 0.6893 - auc: 0.7522 - val_loss: 0.6236 - val_accuracy: 0.6620 -  
val_auc: 0.7375  
Epoch 652/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6137 -  
accuracy: 0.6822 - auc: 0.7526 - val_loss: 0.6283 - val_accuracy: 0.6574 -  
val_auc: 0.7313  
Epoch 653/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6125 -  
accuracy: 0.6820 - auc: 0.7535 - val_loss: 0.6235 - val_accuracy: 0.6613 -  
val_auc: 0.7368  
Epoch 654/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6157 -
```

```
accuracy: 0.6807 - auc: 0.7497 - val_loss: 0.6257 - val_accuracy: 0.6620 -  
val_auc: 0.7338  
Epoch 655/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6130 -  
accuracy: 0.6820 - auc: 0.7529 - val_loss: 0.6254 - val_accuracy: 0.6582 -  
val_auc: 0.7350  
Epoch 656/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6131 -  
accuracy: 0.6888 - auc: 0.7523 - val_loss: 0.6288 - val_accuracy: 0.6520 -  
val_auc: 0.7306  
Epoch 657/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6129 -  
accuracy: 0.6862 - auc: 0.7523 - val_loss: 0.6267 - val_accuracy: 0.6590 -  
val_auc: 0.7334  
Epoch 658/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6136 -  
accuracy: 0.6863 - auc: 0.7521 - val_loss: 0.6258 - val_accuracy: 0.6620 -  
val_auc: 0.7342  
Epoch 659/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6144 -  
accuracy: 0.6887 - auc: 0.7514 - val_loss: 0.6252 - val_accuracy: 0.6574 -  
val_auc: 0.7350  
Epoch 660/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6117 -  
accuracy: 0.6885 - auc: 0.7552 - val_loss: 0.6252 - val_accuracy: 0.6582 -  
val_auc: 0.7347  
Epoch 661/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6134 -  
accuracy: 0.6872 - auc: 0.7519 - val_loss: 0.6253 - val_accuracy: 0.6597 -  
val_auc: 0.7349  
Epoch 662/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6143 -  
accuracy: 0.6849 - auc: 0.7508 - val_loss: 0.6290 - val_accuracy: 0.6582 -  
val_auc: 0.7296  
Epoch 663/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6111 -  
accuracy: 0.6852 - auc: 0.7541 - val_loss: 0.6296 - val_accuracy: 0.6566 -  
val_auc: 0.7286  
Epoch 664/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6131 -  
accuracy: 0.6825 - auc: 0.7523 - val_loss: 0.6246 - val_accuracy: 0.6597 -  
val_auc: 0.7357  
Epoch 665/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6133 -  
accuracy: 0.6832 - auc: 0.7501 - val_loss: 0.6220 - val_accuracy: 0.6620 -  
val_auc: 0.7397  
Epoch 666/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6125 -
```

```
accuracy: 0.6835 - auc: 0.7540 - val_loss: 0.6242 - val_accuracy: 0.6590 -  
val_auc: 0.7361  
Epoch 667/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6128 -  
accuracy: 0.6872 - auc: 0.7519 - val_loss: 0.6210 - val_accuracy: 0.6566 -  
val_auc: 0.7401  
Epoch 668/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6112 -  
accuracy: 0.6817 - auc: 0.7542 - val_loss: 0.6245 - val_accuracy: 0.6574 -  
val_auc: 0.7353  
Epoch 669/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6109 -  
accuracy: 0.6892 - auc: 0.7550 - val_loss: 0.6264 - val_accuracy: 0.6559 -  
val_auc: 0.7331  
Epoch 670/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6120 -  
accuracy: 0.6849 - auc: 0.7534 - val_loss: 0.6204 - val_accuracy: 0.6651 -  
val_auc: 0.7416  
Epoch 671/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6123 -  
accuracy: 0.6857 - auc: 0.7537 - val_loss: 0.6216 - val_accuracy: 0.6566 -  
val_auc: 0.7388  
Epoch 672/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6104 -  
accuracy: 0.6878 - auc: 0.7549 - val_loss: 0.6253 - val_accuracy: 0.6574 -  
val_auc: 0.7339  
Epoch 673/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6119 -  
accuracy: 0.6870 - auc: 0.7533 - val_loss: 0.6229 - val_accuracy: 0.6512 -  
val_auc: 0.7367  
Epoch 674/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6119 -  
accuracy: 0.6867 - auc: 0.7541 - val_loss: 0.6255 - val_accuracy: 0.6582 -  
val_auc: 0.7344  
Epoch 675/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6111 -  
accuracy: 0.6845 - auc: 0.7544 - val_loss: 0.6229 - val_accuracy: 0.6605 -  
val_auc: 0.7375  
Epoch 676/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6148 -  
accuracy: 0.6830 - auc: 0.7505 - val_loss: 0.6249 - val_accuracy: 0.6566 -  
val_auc: 0.7350  
Epoch 677/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6110 -  
accuracy: 0.6890 - auc: 0.7550 - val_loss: 0.6248 - val_accuracy: 0.6605 -  
val_auc: 0.7346  
Epoch 678/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6123 -
```

```
accuracy: 0.6832 - auc: 0.7531 - val_loss: 0.6236 - val_accuracy: 0.6605 -  
val_auc: 0.7364  
Epoch 679/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6126 -  
accuracy: 0.6827 - auc: 0.7523 - val_loss: 0.6237 - val_accuracy: 0.6574 -  
val_auc: 0.7363  
Epoch 680/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6125 -  
accuracy: 0.6867 - auc: 0.7523 - val_loss: 0.6230 - val_accuracy: 0.6620 -  
val_auc: 0.7369  
Epoch 681/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6124 -  
accuracy: 0.6862 - auc: 0.7520 - val_loss: 0.6253 - val_accuracy: 0.6566 -  
val_auc: 0.7340  
Epoch 682/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6111 -  
accuracy: 0.6865 - auc: 0.7537 - val_loss: 0.6240 - val_accuracy: 0.6574 -  
val_auc: 0.7357  
Epoch 683/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6138 -  
accuracy: 0.6842 - auc: 0.7509 - val_loss: 0.6251 - val_accuracy: 0.6566 -  
val_auc: 0.7342  
Epoch 684/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6118 -  
accuracy: 0.6863 - auc: 0.7535 - val_loss: 0.6229 - val_accuracy: 0.6551 -  
val_auc: 0.7369  
Epoch 685/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6128 -  
accuracy: 0.6860 - auc: 0.7517 - val_loss: 0.6273 - val_accuracy: 0.6566 -  
val_auc: 0.7313  
Epoch 686/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6103 -  
accuracy: 0.6883 - auc: 0.7554 - val_loss: 0.6230 - val_accuracy: 0.6636 -  
val_auc: 0.7374  
Epoch 687/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6127 -  
accuracy: 0.6870 - auc: 0.7511 - val_loss: 0.6223 - val_accuracy: 0.6566 -  
val_auc: 0.7375  
Epoch 688/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6143 -  
accuracy: 0.6840 - auc: 0.7518 - val_loss: 0.6194 - val_accuracy: 0.6590 -  
val_auc: 0.7412  
Epoch 689/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6111 -  
accuracy: 0.6870 - auc: 0.7541 - val_loss: 0.6275 - val_accuracy: 0.6551 -  
val_auc: 0.7305  
Epoch 690/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6102 -
```

```
accuracy: 0.6896 - auc: 0.7553 - val_loss: 0.6227 - val_accuracy: 0.6605 -  
val_auc: 0.7369  
Epoch 691/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6124 -  
accuracy: 0.6852 - auc: 0.7533 - val_loss: 0.6255 - val_accuracy: 0.6613 -  
val_auc: 0.7334  
Epoch 692/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6129 -  
accuracy: 0.6880 - auc: 0.7514 - val_loss: 0.6252 - val_accuracy: 0.6605 -  
val_auc: 0.7331  
Epoch 693/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6121 -  
accuracy: 0.6882 - auc: 0.7530 - val_loss: 0.6234 - val_accuracy: 0.6590 -  
val_auc: 0.7363  
Epoch 694/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6107 -  
accuracy: 0.6858 - auc: 0.7542 - val_loss: 0.6220 - val_accuracy: 0.6566 -  
val_auc: 0.7376  
Epoch 695/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6117 -  
accuracy: 0.6903 - auc: 0.7542 - val_loss: 0.6232 - val_accuracy: 0.6597 -  
val_auc: 0.7361  
Epoch 696/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6109 -  
accuracy: 0.6832 - auc: 0.7535 - val_loss: 0.6240 - val_accuracy: 0.6590 -  
val_auc: 0.7350  
Epoch 697/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6108 -  
accuracy: 0.6918 - auc: 0.7547 - val_loss: 0.6218 - val_accuracy: 0.6597 -  
val_auc: 0.7379  
Epoch 698/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6117 -  
accuracy: 0.6865 - auc: 0.7519 - val_loss: 0.6209 - val_accuracy: 0.6582 -  
val_auc: 0.7389  
Epoch 699/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6138 -  
accuracy: 0.6842 - auc: 0.7495 - val_loss: 0.6238 - val_accuracy: 0.6597 -  
val_auc: 0.7350  
Epoch 700/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6117 -  
accuracy: 0.6890 - auc: 0.7538 - val_loss: 0.6234 - val_accuracy: 0.6590 -  
val_auc: 0.7358  
Epoch 701/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6116 -  
accuracy: 0.6829 - auc: 0.7515 - val_loss: 0.6211 - val_accuracy: 0.6543 -  
val_auc: 0.7383  
Epoch 702/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6113 -
```

```
accuracy: 0.6890 - auc: 0.7535 - val_loss: 0.6214 - val_accuracy: 0.6582 -  
val_auc: 0.7385  
Epoch 703/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6108 -  
accuracy: 0.6877 - auc: 0.7548 - val_loss: 0.6236 - val_accuracy: 0.6574 -  
val_auc: 0.7353  
Epoch 704/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6107 -  
accuracy: 0.6857 - auc: 0.7537 - val_loss: 0.6213 - val_accuracy: 0.6613 -  
val_auc: 0.7387  
Epoch 705/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6112 -  
accuracy: 0.6885 - auc: 0.7523 - val_loss: 0.6210 - val_accuracy: 0.6574 -  
val_auc: 0.7387  
Epoch 706/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6094 -  
accuracy: 0.6865 - auc: 0.7558 - val_loss: 0.6194 - val_accuracy: 0.6582 -  
val_auc: 0.7410  
Epoch 707/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6106 -  
accuracy: 0.6854 - auc: 0.7537 - val_loss: 0.6226 - val_accuracy: 0.6582 -  
val_auc: 0.7364  
Epoch 708/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6127 -  
accuracy: 0.6857 - auc: 0.7512 - val_loss: 0.6196 - val_accuracy: 0.6559 -  
val_auc: 0.7397  
Epoch 709/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6123 -  
accuracy: 0.6849 - auc: 0.7513 - val_loss: 0.6220 - val_accuracy: 0.6605 -  
val_auc: 0.7373  
Epoch 710/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6113 -  
accuracy: 0.6863 - auc: 0.7534 - val_loss: 0.6234 - val_accuracy: 0.6574 -  
val_auc: 0.7352  
Epoch 711/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6106 -  
accuracy: 0.6898 - auc: 0.7543 - val_loss: 0.6231 - val_accuracy: 0.6566 -  
val_auc: 0.7358  
Epoch 712/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6121 -  
accuracy: 0.6880 - auc: 0.7523 - val_loss: 0.6227 - val_accuracy: 0.6566 -  
val_auc: 0.7360  
Epoch 713/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6096 -  
accuracy: 0.6858 - auc: 0.7552 - val_loss: 0.6195 - val_accuracy: 0.6566 -  
val_auc: 0.7404  
Epoch 714/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6104 -
```

```
accuracy: 0.6888 - auc: 0.7550 - val_loss: 0.6206 - val_accuracy: 0.6605 -  
val_auc: 0.7391  
Epoch 715/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6105 -  
accuracy: 0.6852 - auc: 0.7532 - val_loss: 0.6221 - val_accuracy: 0.6597 -  
val_auc: 0.7365  
Epoch 716/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6086 -  
accuracy: 0.6928 - auc: 0.7567 - val_loss: 0.6213 - val_accuracy: 0.6528 -  
val_auc: 0.7377  
Epoch 717/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6096 -  
accuracy: 0.6888 - auc: 0.7553 - val_loss: 0.6188 - val_accuracy: 0.6566 -  
val_auc: 0.7409  
Epoch 718/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6107 -  
accuracy: 0.6872 - auc: 0.7535 - val_loss: 0.6255 - val_accuracy: 0.6535 -  
val_auc: 0.7325  
Epoch 719/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6097 -  
accuracy: 0.6887 - auc: 0.7556 - val_loss: 0.6220 - val_accuracy: 0.6636 -  
val_auc: 0.7372  
Epoch 720/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6086 -  
accuracy: 0.6888 - auc: 0.7563 - val_loss: 0.6189 - val_accuracy: 0.6613 -  
val_auc: 0.7416  
Epoch 721/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6110 -  
accuracy: 0.6860 - auc: 0.7532 - val_loss: 0.6201 - val_accuracy: 0.6597 -  
val_auc: 0.7398  
Epoch 722/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6086 -  
accuracy: 0.6900 - auc: 0.7564 - val_loss: 0.6218 - val_accuracy: 0.6590 -  
val_auc: 0.7371  
Epoch 723/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6128 -  
accuracy: 0.6875 - auc: 0.7508 - val_loss: 0.6217 - val_accuracy: 0.6597 -  
val_auc: 0.7375  
Epoch 724/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6106 -  
accuracy: 0.6857 - auc: 0.7544 - val_loss: 0.6194 - val_accuracy: 0.6574 -  
val_auc: 0.7406  
Epoch 725/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6080 -  
accuracy: 0.6887 - auc: 0.7573 - val_loss: 0.6193 - val_accuracy: 0.6582 -  
val_auc: 0.7402  
Epoch 726/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6113 -
```

```
accuracy: 0.6862 - auc: 0.7522 - val_loss: 0.6213 - val_accuracy: 0.6582 -  
val_auc: 0.7377  
Epoch 727/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6116 -  
accuracy: 0.6825 - auc: 0.7526 - val_loss: 0.6202 - val_accuracy: 0.6582 -  
val_auc: 0.7393  
Epoch 728/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6091 -  
accuracy: 0.6830 - auc: 0.7546 - val_loss: 0.6201 - val_accuracy: 0.6566 -  
val_auc: 0.7390  
Epoch 729/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6087 -  
accuracy: 0.6870 - auc: 0.7560 - val_loss: 0.6223 - val_accuracy: 0.6620 -  
val_auc: 0.7365  
Epoch 730/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6088 -  
accuracy: 0.6910 - auc: 0.7559 - val_loss: 0.6193 - val_accuracy: 0.6613 -  
val_auc: 0.7400  
Epoch 731/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6105 -  
accuracy: 0.6892 - auc: 0.7548 - val_loss: 0.6228 - val_accuracy: 0.6574 -  
val_auc: 0.7350  
Epoch 732/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6111 -  
accuracy: 0.6858 - auc: 0.7522 - val_loss: 0.6205 - val_accuracy: 0.6551 -  
val_auc: 0.7386  
Epoch 733/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6101 -  
accuracy: 0.6877 - auc: 0.7543 - val_loss: 0.6261 - val_accuracy: 0.6582 -  
val_auc: 0.7311  
Epoch 734/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6081 -  
accuracy: 0.6926 - auc: 0.7568 - val_loss: 0.6185 - val_accuracy: 0.6559 -  
val_auc: 0.7405  
Epoch 735/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6089 -  
accuracy: 0.6873 - auc: 0.7557 - val_loss: 0.6214 - val_accuracy: 0.6590 -  
val_auc: 0.7374  
Epoch 736/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6108 -  
accuracy: 0.6847 - auc: 0.7528 - val_loss: 0.6212 - val_accuracy: 0.6528 -  
val_auc: 0.7370  
Epoch 737/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6107 -  
accuracy: 0.6849 - auc: 0.7536 - val_loss: 0.6177 - val_accuracy: 0.6605 -  
val_auc: 0.7419  
Epoch 738/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6102 -
```

```
accuracy: 0.6862 - auc: 0.7547 - val_loss: 0.6201 - val_accuracy: 0.6574 -  
val_auc: 0.7393  
Epoch 739/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6100 -  
accuracy: 0.6878 - auc: 0.7540 - val_loss: 0.6204 - val_accuracy: 0.6574 -  
val_auc: 0.7385  
Epoch 740/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6105 -  
accuracy: 0.6877 - auc: 0.7538 - val_loss: 0.6200 - val_accuracy: 0.6597 -  
val_auc: 0.7391  
Epoch 741/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6099 -  
accuracy: 0.6865 - auc: 0.7542 - val_loss: 0.6222 - val_accuracy: 0.6582 -  
val_auc: 0.7364  
Epoch 742/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6095 -  
accuracy: 0.6863 - auc: 0.7549 - val_loss: 0.6217 - val_accuracy: 0.6590 -  
val_auc: 0.7366  
Epoch 743/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6092 -  
accuracy: 0.6877 - auc: 0.7554 - val_loss: 0.6214 - val_accuracy: 0.6574 -  
val_auc: 0.7371  
Epoch 744/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6078 -  
accuracy: 0.6906 - auc: 0.7571 - val_loss: 0.6259 - val_accuracy: 0.6590 -  
val_auc: 0.7311  
Epoch 745/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6112 -  
accuracy: 0.6880 - auc: 0.7528 - val_loss: 0.6185 - val_accuracy: 0.6582 -  
val_auc: 0.7409  
Epoch 746/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6091 -  
accuracy: 0.6863 - auc: 0.7554 - val_loss: 0.6218 - val_accuracy: 0.6574 -  
val_auc: 0.7363  
Epoch 747/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6083 -  
accuracy: 0.6890 - auc: 0.7556 - val_loss: 0.6213 - val_accuracy: 0.6620 -  
val_auc: 0.7376  
Epoch 748/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6086 -  
accuracy: 0.6908 - auc: 0.7554 - val_loss: 0.6198 - val_accuracy: 0.6566 -  
val_auc: 0.7397  
Epoch 749/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6082 -  
accuracy: 0.6931 - auc: 0.7567 - val_loss: 0.6165 - val_accuracy: 0.6582 -  
val_auc: 0.7428  
Epoch 750/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6092 -
```

```
accuracy: 0.6893 - auc: 0.7550 - val_loss: 0.6195 - val_accuracy: 0.6597 -  
val_auc: 0.7397  
Epoch 751/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6081 -  
accuracy: 0.6887 - auc: 0.7564 - val_loss: 0.6210 - val_accuracy: 0.6605 -  
val_auc: 0.7374  
Epoch 752/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6100 -  
accuracy: 0.6882 - auc: 0.7543 - val_loss: 0.6224 - val_accuracy: 0.6590 -  
val_auc: 0.7356  
Epoch 753/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6121 -  
accuracy: 0.6910 - auc: 0.7516 - val_loss: 0.6193 - val_accuracy: 0.6559 -  
val_auc: 0.7396  
Epoch 754/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6087 -  
accuracy: 0.6906 - auc: 0.7551 - val_loss: 0.6253 - val_accuracy: 0.6566 -  
val_auc: 0.7314  
Epoch 755/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6098 -  
accuracy: 0.6863 - auc: 0.7538 - val_loss: 0.6200 - val_accuracy: 0.6574 -  
val_auc: 0.7386  
Epoch 756/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6077 -  
accuracy: 0.6900 - auc: 0.7568 - val_loss: 0.6202 - val_accuracy: 0.6574 -  
val_auc: 0.7383  
Epoch 757/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6104 -  
accuracy: 0.6887 - auc: 0.7542 - val_loss: 0.6205 - val_accuracy: 0.6597 -  
val_auc: 0.7375  
Epoch 758/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6103 -  
accuracy: 0.6872 - auc: 0.7537 - val_loss: 0.6262 - val_accuracy: 0.6574 -  
val_auc: 0.7308  
Epoch 759/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6101 -  
accuracy: 0.6887 - auc: 0.7531 - val_loss: 0.6204 - val_accuracy: 0.6590 -  
val_auc: 0.7380  
Epoch 760/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6082 -  
accuracy: 0.6849 - auc: 0.7564 - val_loss: 0.6206 - val_accuracy: 0.6613 -  
val_auc: 0.7382  
Epoch 761/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6090 -  
accuracy: 0.6883 - auc: 0.7552 - val_loss: 0.6214 - val_accuracy: 0.6590 -  
val_auc: 0.7368  
Epoch 762/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6088 -
```

```
accuracy: 0.6887 - auc: 0.7547 - val_loss: 0.6209 - val_accuracy: 0.6582 -  
val_auc: 0.7373  
Epoch 763/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6089 -  
accuracy: 0.6905 - auc: 0.7553 - val_loss: 0.6211 - val_accuracy: 0.6574 -  
val_auc: 0.7375  
Epoch 764/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6085 -  
accuracy: 0.6896 - auc: 0.7557 - val_loss: 0.6185 - val_accuracy: 0.6605 -  
val_auc: 0.7411  
Epoch 765/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6079 -  
accuracy: 0.6868 - auc: 0.7566 - val_loss: 0.6188 - val_accuracy: 0.6574 -  
val_auc: 0.7404  
Epoch 766/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6075 -  
accuracy: 0.6858 - auc: 0.7564 - val_loss: 0.6209 - val_accuracy: 0.6582 -  
val_auc: 0.7374  
Epoch 767/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6077 -  
accuracy: 0.6873 - auc: 0.7567 - val_loss: 0.6227 - val_accuracy: 0.6636 -  
val_auc: 0.7354  
Epoch 768/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6093 -  
accuracy: 0.6858 - auc: 0.7540 - val_loss: 0.6208 - val_accuracy: 0.6620 -  
val_auc: 0.7372  
Epoch 769/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6102 -  
accuracy: 0.6900 - auc: 0.7534 - val_loss: 0.6191 - val_accuracy: 0.6582 -  
val_auc: 0.7394  
Epoch 770/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6070 -  
accuracy: 0.6916 - auc: 0.7581 - val_loss: 0.6218 - val_accuracy: 0.6620 -  
val_auc: 0.7359  
Epoch 771/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6100 -  
accuracy: 0.6905 - auc: 0.7539 - val_loss: 0.6194 - val_accuracy: 0.6628 -  
val_auc: 0.7402  
Epoch 772/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6090 -  
accuracy: 0.6880 - auc: 0.7551 - val_loss: 0.6220 - val_accuracy: 0.6636 -  
val_auc: 0.7358  
Epoch 773/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6080 -  
accuracy: 0.6863 - auc: 0.7560 - val_loss: 0.6176 - val_accuracy: 0.6551 -  
val_auc: 0.7419  
Epoch 774/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6078 -
```

```
accuracy: 0.6953 - auc: 0.7572 - val_loss: 0.6209 - val_accuracy: 0.6597 -  
val_auc: 0.7376  
Epoch 775/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6083 -  
accuracy: 0.6855 - auc: 0.7546 - val_loss: 0.6207 - val_accuracy: 0.6597 -  
val_auc: 0.7373  
Epoch 776/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6085 -  
accuracy: 0.6913 - auc: 0.7558 - val_loss: 0.6197 - val_accuracy: 0.6597 -  
val_auc: 0.7382  
Epoch 777/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6097 -  
accuracy: 0.6854 - auc: 0.7533 - val_loss: 0.6189 - val_accuracy: 0.6582 -  
val_auc: 0.7395  
Epoch 778/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6098 -  
accuracy: 0.6885 - auc: 0.7548 - val_loss: 0.6205 - val_accuracy: 0.6613 -  
val_auc: 0.7376  
Epoch 779/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6087 -  
accuracy: 0.6880 - auc: 0.7556 - val_loss: 0.6206 - val_accuracy: 0.6590 -  
val_auc: 0.7372  
Epoch 780/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6065 -  
accuracy: 0.6913 - auc: 0.7577 - val_loss: 0.6207 - val_accuracy: 0.6605 -  
val_auc: 0.7371  
Epoch 781/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6070 -  
accuracy: 0.6936 - auc: 0.7569 - val_loss: 0.6237 - val_accuracy: 0.6636 -  
val_auc: 0.7338  
Epoch 782/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6106 -  
accuracy: 0.6873 - auc: 0.7533 - val_loss: 0.6181 - val_accuracy: 0.6574 -  
val_auc: 0.7406  
Epoch 783/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6069 -  
accuracy: 0.6892 - auc: 0.7573 - val_loss: 0.6205 - val_accuracy: 0.6605 -  
val_auc: 0.7376  
Epoch 784/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6085 -  
accuracy: 0.6882 - auc: 0.7554 - val_loss: 0.6172 - val_accuracy: 0.6582 -  
val_auc: 0.7414  
Epoch 785/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6069 -  
accuracy: 0.6893 - auc: 0.7564 - val_loss: 0.6237 - val_accuracy: 0.6590 -  
val_auc: 0.7333  
Epoch 786/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6069 -
```

```
accuracy: 0.6925 - auc: 0.7564 - val_loss: 0.6209 - val_accuracy: 0.6605 -  
val_auc: 0.7366  
Epoch 787/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6080 -  
accuracy: 0.6877 - auc: 0.7555 - val_loss: 0.6178 - val_accuracy: 0.6566 -  
val_auc: 0.7405  
Epoch 788/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6082 -  
accuracy: 0.6916 - auc: 0.7556 - val_loss: 0.6205 - val_accuracy: 0.6597 -  
val_auc: 0.7373  
Epoch 789/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6070 -  
accuracy: 0.6908 - auc: 0.7568 - val_loss: 0.6217 - val_accuracy: 0.6636 -  
val_auc: 0.7365  
Epoch 790/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6086 -  
accuracy: 0.6920 - auc: 0.7555 - val_loss: 0.6214 - val_accuracy: 0.6620 -  
val_auc: 0.7362  
Epoch 791/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6073 -  
accuracy: 0.6880 - auc: 0.7549 - val_loss: 0.6190 - val_accuracy: 0.6582 -  
val_auc: 0.7392  
Epoch 792/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6079 -  
accuracy: 0.6896 - auc: 0.7560 - val_loss: 0.6212 - val_accuracy: 0.6628 -  
val_auc: 0.7359  
Epoch 793/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6080 -  
accuracy: 0.6906 - auc: 0.7560 - val_loss: 0.6196 - val_accuracy: 0.6605 -  
val_auc: 0.7389  
Epoch 794/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6083 -  
accuracy: 0.6900 - auc: 0.7554 - val_loss: 0.6204 - val_accuracy: 0.6628 -  
val_auc: 0.7377  
Epoch 795/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6075 -  
accuracy: 0.6925 - auc: 0.7563 - val_loss: 0.6177 - val_accuracy: 0.6582 -  
val_auc: 0.7411  
Epoch 796/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6086 -  
accuracy: 0.6825 - auc: 0.7534 - val_loss: 0.6210 - val_accuracy: 0.6590 -  
val_auc: 0.7366  
Epoch 797/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6078 -  
accuracy: 0.6906 - auc: 0.7565 - val_loss: 0.6181 - val_accuracy: 0.6605 -  
val_auc: 0.7403  
Epoch 798/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6080 -
```

```
accuracy: 0.6862 - auc: 0.7551 - val_loss: 0.6192 - val_accuracy: 0.6582 -  
val_auc: 0.7387  
Epoch 799/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6074 -  
accuracy: 0.6926 - auc: 0.7566 - val_loss: 0.6198 - val_accuracy: 0.6628 -  
val_auc: 0.7379  
Epoch 800/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6067 -  
accuracy: 0.6920 - auc: 0.7593 - val_loss: 0.6201 - val_accuracy: 0.6605 -  
val_auc: 0.7377  
Epoch 801/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6080 -  
accuracy: 0.6872 - auc: 0.7546 - val_loss: 0.6221 - val_accuracy: 0.6605 -  
val_auc: 0.7349  
Epoch 802/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6088 -  
accuracy: 0.6941 - auc: 0.7548 - val_loss: 0.6169 - val_accuracy: 0.6559 -  
val_auc: 0.7409  
Epoch 803/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6059 -  
accuracy: 0.6893 - auc: 0.7588 - val_loss: 0.6181 - val_accuracy: 0.6566 -  
val_auc: 0.7401  
Epoch 804/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6062 -  
accuracy: 0.6936 - auc: 0.7584 - val_loss: 0.6186 - val_accuracy: 0.6582 -  
val_auc: 0.7394  
Epoch 805/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6059 -  
accuracy: 0.6910 - auc: 0.7570 - val_loss: 0.6182 - val_accuracy: 0.6566 -  
val_auc: 0.7398  
Epoch 806/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6069 -  
accuracy: 0.6920 - auc: 0.7577 - val_loss: 0.6191 - val_accuracy: 0.6597 -  
val_auc: 0.7386  
Epoch 807/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6056 -  
accuracy: 0.6916 - auc: 0.7584 - val_loss: 0.6202 - val_accuracy: 0.6597 -  
val_auc: 0.7366  
Epoch 808/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6068 -  
accuracy: 0.6880 - auc: 0.7560 - val_loss: 0.6206 - val_accuracy: 0.6582 -  
val_auc: 0.7368  
Epoch 809/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6062 -  
accuracy: 0.6908 - auc: 0.7571 - val_loss: 0.6178 - val_accuracy: 0.6597 -  
val_auc: 0.7403  
Epoch 810/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6067 -
```

```
accuracy: 0.6911 - auc: 0.7573 - val_loss: 0.6183 - val_accuracy: 0.6590 -  
val_auc: 0.7396  
Epoch 811/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6071 -  
accuracy: 0.6926 - auc: 0.7568 - val_loss: 0.6227 - val_accuracy: 0.6597 -  
val_auc: 0.7341  
Epoch 812/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6074 -  
accuracy: 0.6887 - auc: 0.7561 - val_loss: 0.6199 - val_accuracy: 0.6574 -  
val_auc: 0.7378  
Epoch 813/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6068 -  
accuracy: 0.6885 - auc: 0.7573 - val_loss: 0.6186 - val_accuracy: 0.6605 -  
val_auc: 0.7391  
Epoch 814/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6049 -  
accuracy: 0.6893 - auc: 0.7589 - val_loss: 0.6219 - val_accuracy: 0.6597 -  
val_auc: 0.7349  
Epoch 815/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -  
accuracy: 0.6926 - auc: 0.7579 - val_loss: 0.6194 - val_accuracy: 0.6613 -  
val_auc: 0.7383  
Epoch 816/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6080 -  
accuracy: 0.6885 - auc: 0.7553 - val_loss: 0.6179 - val_accuracy: 0.6605 -  
val_auc: 0.7400  
Epoch 817/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6080 -  
accuracy: 0.6883 - auc: 0.7559 - val_loss: 0.6195 - val_accuracy: 0.6628 -  
val_auc: 0.7386  
Epoch 818/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6069 -  
accuracy: 0.6949 - auc: 0.7569 - val_loss: 0.6195 - val_accuracy: 0.6605 -  
val_auc: 0.7386  
Epoch 819/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6075 -  
accuracy: 0.6910 - auc: 0.7553 - val_loss: 0.6209 - val_accuracy: 0.6613 -  
val_auc: 0.7363  
Epoch 820/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6062 -  
accuracy: 0.6948 - auc: 0.7581 - val_loss: 0.6188 - val_accuracy: 0.6620 -  
val_auc: 0.7394  
Epoch 821/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6067 -  
accuracy: 0.6888 - auc: 0.7562 - val_loss: 0.6192 - val_accuracy: 0.6574 -  
val_auc: 0.7386  
Epoch 822/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6064 -
```

```
accuracy: 0.6910 - auc: 0.7573 - val_loss: 0.6179 - val_accuracy: 0.6590 -  
val_auc: 0.7398  
Epoch 823/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6071 -  
accuracy: 0.6916 - auc: 0.7567 - val_loss: 0.6159 - val_accuracy: 0.6582 -  
val_auc: 0.7433  
Epoch 824/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6070 -  
accuracy: 0.6898 - auc: 0.7567 - val_loss: 0.6167 - val_accuracy: 0.6551 -  
val_auc: 0.7424  
Epoch 825/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6087 -  
accuracy: 0.6903 - auc: 0.7553 - val_loss: 0.6193 - val_accuracy: 0.6628 -  
val_auc: 0.7384  
Epoch 826/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6066 -  
accuracy: 0.6877 - auc: 0.7567 - val_loss: 0.6184 - val_accuracy: 0.6620 -  
val_auc: 0.7395  
Epoch 827/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6066 -  
accuracy: 0.6887 - auc: 0.7574 - val_loss: 0.6197 - val_accuracy: 0.6620 -  
val_auc: 0.7378  
Epoch 828/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6074 -  
accuracy: 0.6921 - auc: 0.7563 - val_loss: 0.6208 - val_accuracy: 0.6651 -  
val_auc: 0.7364  
Epoch 829/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6062 -  
accuracy: 0.6905 - auc: 0.7581 - val_loss: 0.6187 - val_accuracy: 0.6574 -  
val_auc: 0.7393  
Epoch 830/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6058 -  
accuracy: 0.6885 - auc: 0.7569 - val_loss: 0.6185 - val_accuracy: 0.6597 -  
val_auc: 0.7400  
Epoch 831/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6072 -  
accuracy: 0.6860 - auc: 0.7554 - val_loss: 0.6173 - val_accuracy: 0.6605 -  
val_auc: 0.7412  
Epoch 832/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6063 -  
accuracy: 0.6933 - auc: 0.7591 - val_loss: 0.6178 - val_accuracy: 0.6574 -  
val_auc: 0.7405  
Epoch 833/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6071 -  
accuracy: 0.6900 - auc: 0.7557 - val_loss: 0.6214 - val_accuracy: 0.6636 -  
val_auc: 0.7360  
Epoch 834/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6056 -
```

```
accuracy: 0.6862 - auc: 0.7578 - val_loss: 0.6181 - val_accuracy: 0.6597 -  
val_auc: 0.7402  
Epoch 835/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6066 -  
accuracy: 0.6901 - auc: 0.7572 - val_loss: 0.6185 - val_accuracy: 0.6605 -  
val_auc: 0.7395  
Epoch 836/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6046 -  
accuracy: 0.6931 - auc: 0.7601 - val_loss: 0.6186 - val_accuracy: 0.6597 -  
val_auc: 0.7391  
Epoch 837/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6051 -  
accuracy: 0.6926 - auc: 0.7589 - val_loss: 0.6196 - val_accuracy: 0.6628 -  
val_auc: 0.7375  
Epoch 838/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6063 -  
accuracy: 0.6913 - auc: 0.7582 - val_loss: 0.6171 - val_accuracy: 0.6590 -  
val_auc: 0.7412  
Epoch 839/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6074 -  
accuracy: 0.6863 - auc: 0.7550 - val_loss: 0.6182 - val_accuracy: 0.6605 -  
val_auc: 0.7394  
Epoch 840/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6072 -  
accuracy: 0.6948 - auc: 0.7566 - val_loss: 0.6187 - val_accuracy: 0.6590 -  
val_auc: 0.7394  
Epoch 841/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6054 -  
accuracy: 0.6954 - auc: 0.7586 - val_loss: 0.6173 - val_accuracy: 0.6590 -  
val_auc: 0.7413  
Epoch 842/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -  
accuracy: 0.6880 - auc: 0.7588 - val_loss: 0.6197 - val_accuracy: 0.6636 -  
val_auc: 0.7379  
Epoch 843/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6061 -  
accuracy: 0.6895 - auc: 0.7572 - val_loss: 0.6179 - val_accuracy: 0.6605 -  
val_auc: 0.7398  
Epoch 844/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6062 -  
accuracy: 0.6913 - auc: 0.7584 - val_loss: 0.6231 - val_accuracy: 0.6613 -  
val_auc: 0.7330  
Epoch 845/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6056 -  
accuracy: 0.6931 - auc: 0.7575 - val_loss: 0.6163 - val_accuracy: 0.6566 -  
val_auc: 0.7421  
Epoch 846/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6062 -
```

```
accuracy: 0.6920 - auc: 0.7576 - val_loss: 0.6203 - val_accuracy: 0.6605 -  
val_auc: 0.7371  
Epoch 847/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6070 -  
accuracy: 0.6873 - auc: 0.7561 - val_loss: 0.6199 - val_accuracy: 0.6620 -  
val_auc: 0.7377  
Epoch 848/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6054 -  
accuracy: 0.6930 - auc: 0.7586 - val_loss: 0.6177 - val_accuracy: 0.6574 -  
val_auc: 0.7403  
Epoch 849/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6059 -  
accuracy: 0.6901 - auc: 0.7570 - val_loss: 0.6165 - val_accuracy: 0.6613 -  
val_auc: 0.7418  
Epoch 850/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6060 -  
accuracy: 0.6931 - auc: 0.7591 - val_loss: 0.6188 - val_accuracy: 0.6597 -  
val_auc: 0.7387  
Epoch 851/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6049 -  
accuracy: 0.6906 - auc: 0.7589 - val_loss: 0.6193 - val_accuracy: 0.6590 -  
val_auc: 0.7377  
Epoch 852/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6066 -  
accuracy: 0.6860 - auc: 0.7559 - val_loss: 0.6167 - val_accuracy: 0.6535 -  
val_auc: 0.7417  
Epoch 853/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6062 -  
accuracy: 0.6896 - auc: 0.7564 - val_loss: 0.6173 - val_accuracy: 0.6597 -  
val_auc: 0.7403  
Epoch 854/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -  
accuracy: 0.6930 - auc: 0.7590 - val_loss: 0.6189 - val_accuracy: 0.6597 -  
val_auc: 0.7386  
Epoch 855/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6057 -  
accuracy: 0.6910 - auc: 0.7574 - val_loss: 0.6193 - val_accuracy: 0.6582 -  
val_auc: 0.7380  
Epoch 856/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6046 -  
accuracy: 0.6885 - auc: 0.7584 - val_loss: 0.6189 - val_accuracy: 0.6613 -  
val_auc: 0.7385  
Epoch 857/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6083 -  
accuracy: 0.6854 - auc: 0.7537 - val_loss: 0.6194 - val_accuracy: 0.6628 -  
val_auc: 0.7381  
Epoch 858/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6052 -
```

```
accuracy: 0.6910 - auc: 0.7590 - val_loss: 0.6183 - val_accuracy: 0.6628 -
val_auc: 0.7389
Epoch 859/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6062 -
accuracy: 0.6926 - auc: 0.7569 - val_loss: 0.6192 - val_accuracy: 0.6620 -
val_auc: 0.7378
Epoch 860/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6072 -
accuracy: 0.6880 - auc: 0.7558 - val_loss: 0.6194 - val_accuracy: 0.6613 -
val_auc: 0.7373
Epoch 861/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -
accuracy: 0.6908 - auc: 0.7583 - val_loss: 0.6177 - val_accuracy: 0.6590 -
val_auc: 0.7405
Epoch 862/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -
accuracy: 0.6921 - auc: 0.7577 - val_loss: 0.6151 - val_accuracy: 0.6574 -
val_auc: 0.7433
Epoch 863/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6054 -
accuracy: 0.6906 - auc: 0.7592 - val_loss: 0.6178 - val_accuracy: 0.6628 -
val_auc: 0.7396
Epoch 864/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6058 -
accuracy: 0.6926 - auc: 0.7569 - val_loss: 0.6189 - val_accuracy: 0.6620 -
val_auc: 0.7383
Epoch 865/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6066 -
accuracy: 0.6887 - auc: 0.7565 - val_loss: 0.6169 - val_accuracy: 0.6620 -
val_auc: 0.7409
Epoch 866/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6058 -
accuracy: 0.6915 - auc: 0.7577 - val_loss: 0.6177 - val_accuracy: 0.6644 -
val_auc: 0.7398
Epoch 867/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6045 -
accuracy: 0.6920 - auc: 0.7590 - val_loss: 0.6189 - val_accuracy: 0.6644 -
val_auc: 0.7386
Epoch 868/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6073 -
accuracy: 0.6872 - auc: 0.7557 - val_loss: 0.6172 - val_accuracy: 0.6582 -
val_auc: 0.7404
Epoch 869/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6057 -
accuracy: 0.6887 - auc: 0.7573 - val_loss: 0.6208 - val_accuracy: 0.6605 -
val_auc: 0.7360
Epoch 870/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6068 -
```

```
accuracy: 0.6844 - auc: 0.7563 - val_loss: 0.6197 - val_accuracy: 0.6605 -  
val_auc: 0.7376  
Epoch 871/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6073 -  
accuracy: 0.6862 - auc: 0.7550 - val_loss: 0.6188 - val_accuracy: 0.6597 -  
val_auc: 0.7385  
Epoch 872/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6063 -  
accuracy: 0.6875 - auc: 0.7565 - val_loss: 0.6205 - val_accuracy: 0.6582 -  
val_auc: 0.7363  
Epoch 873/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -  
accuracy: 0.6880 - auc: 0.7575 - val_loss: 0.6205 - val_accuracy: 0.6597 -  
val_auc: 0.7363  
Epoch 874/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6056 -  
accuracy: 0.6928 - auc: 0.7591 - val_loss: 0.6169 - val_accuracy: 0.6597 -  
val_auc: 0.7412  
Epoch 875/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6054 -  
accuracy: 0.6908 - auc: 0.7580 - val_loss: 0.6158 - val_accuracy: 0.6574 -  
val_auc: 0.7419  
Epoch 876/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6061 -  
accuracy: 0.6898 - auc: 0.7566 - val_loss: 0.6206 - val_accuracy: 0.6636 -  
val_auc: 0.7365  
Epoch 877/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6062 -  
accuracy: 0.6954 - auc: 0.7585 - val_loss: 0.6183 - val_accuracy: 0.6613 -  
val_auc: 0.7392  
Epoch 878/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6053 -  
accuracy: 0.6916 - auc: 0.7578 - val_loss: 0.6171 - val_accuracy: 0.6597 -  
val_auc: 0.7401  
Epoch 879/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -  
accuracy: 0.6870 - auc: 0.7568 - val_loss: 0.6220 - val_accuracy: 0.6535 -  
val_auc: 0.7342  
Epoch 880/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6042 -  
accuracy: 0.6941 - auc: 0.7599 - val_loss: 0.6183 - val_accuracy: 0.6590 -  
val_auc: 0.7391  
Epoch 881/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6046 -  
accuracy: 0.6931 - auc: 0.7589 - val_loss: 0.6166 - val_accuracy: 0.6605 -  
val_auc: 0.7406  
Epoch 882/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6065 -
```

```
accuracy: 0.6885 - auc: 0.7568 - val_loss: 0.6178 - val_accuracy: 0.6605 -  
val_auc: 0.7394  
Epoch 883/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6039 -  
accuracy: 0.6883 - auc: 0.7598 - val_loss: 0.6158 - val_accuracy: 0.6628 -  
val_auc: 0.7416  
Epoch 884/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6045 -  
accuracy: 0.6916 - auc: 0.7588 - val_loss: 0.6146 - val_accuracy: 0.6590 -  
val_auc: 0.7441  
Epoch 885/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6054 -  
accuracy: 0.6883 - auc: 0.7586 - val_loss: 0.6173 - val_accuracy: 0.6613 -  
val_auc: 0.7400  
Epoch 886/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6049 -  
accuracy: 0.6926 - auc: 0.7586 - val_loss: 0.6175 - val_accuracy: 0.6590 -  
val_auc: 0.7401  
Epoch 887/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6043 -  
accuracy: 0.6931 - auc: 0.7591 - val_loss: 0.6181 - val_accuracy: 0.6628 -  
val_auc: 0.7388  
Epoch 888/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6046 -  
accuracy: 0.6901 - auc: 0.7579 - val_loss: 0.6203 - val_accuracy: 0.6551 -  
val_auc: 0.7360  
Epoch 889/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6051 -  
accuracy: 0.6893 - auc: 0.7584 - val_loss: 0.6157 - val_accuracy: 0.6566 -  
val_auc: 0.7424  
Epoch 890/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6056 -  
accuracy: 0.6868 - auc: 0.7570 - val_loss: 0.6181 - val_accuracy: 0.6636 -  
val_auc: 0.7392  
Epoch 891/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6045 -  
accuracy: 0.6872 - auc: 0.7603 - val_loss: 0.6158 - val_accuracy: 0.6651 -  
val_auc: 0.7428  
Epoch 892/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -  
accuracy: 0.6928 - auc: 0.7581 - val_loss: 0.6172 - val_accuracy: 0.6597 -  
val_auc: 0.7400  
Epoch 893/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6040 -  
accuracy: 0.6915 - auc: 0.7592 - val_loss: 0.6159 - val_accuracy: 0.6582 -  
val_auc: 0.7423  
Epoch 894/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6057 -
```

```
accuracy: 0.6901 - auc: 0.7573 - val_loss: 0.6132 - val_accuracy: 0.6659 -  
val_auc: 0.7454  
Epoch 895/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6064 -  
accuracy: 0.6901 - auc: 0.7575 - val_loss: 0.6153 - val_accuracy: 0.6590 -  
val_auc: 0.7430  
Epoch 896/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6029 -  
accuracy: 0.6978 - auc: 0.7614 - val_loss: 0.6162 - val_accuracy: 0.6605 -  
val_auc: 0.7415  
Epoch 897/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6046 -  
accuracy: 0.6925 - auc: 0.7586 - val_loss: 0.6158 - val_accuracy: 0.6559 -  
val_auc: 0.7417  
Epoch 898/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -  
accuracy: 0.6880 - auc: 0.7575 - val_loss: 0.6179 - val_accuracy: 0.6620 -  
val_auc: 0.7394  
Epoch 899/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6037 -  
accuracy: 0.6906 - auc: 0.7593 - val_loss: 0.6174 - val_accuracy: 0.6613 -  
val_auc: 0.7403  
Epoch 900/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6064 -  
accuracy: 0.6892 - auc: 0.7572 - val_loss: 0.6155 - val_accuracy: 0.6590 -  
val_auc: 0.7427  
Epoch 901/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6039 -  
accuracy: 0.6887 - auc: 0.7594 - val_loss: 0.6175 - val_accuracy: 0.6620 -  
val_auc: 0.7402  
Epoch 902/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6051 -  
accuracy: 0.6921 - auc: 0.7574 - val_loss: 0.6158 - val_accuracy: 0.6597 -  
val_auc: 0.7418  
Epoch 903/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6052 -  
accuracy: 0.6890 - auc: 0.7575 - val_loss: 0.6148 - val_accuracy: 0.6605 -  
val_auc: 0.7434  
Epoch 904/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6044 -  
accuracy: 0.6906 - auc: 0.7589 - val_loss: 0.6188 - val_accuracy: 0.6644 -  
val_auc: 0.7384  
Epoch 905/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6045 -  
accuracy: 0.6880 - auc: 0.7585 - val_loss: 0.6160 - val_accuracy: 0.6613 -  
val_auc: 0.7413  
Epoch 906/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6046 -
```

```
accuracy: 0.6865 - auc: 0.7583 - val_loss: 0.6181 - val_accuracy: 0.6620 -  
val_auc: 0.7392  
Epoch 907/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6033 -  
accuracy: 0.6951 - auc: 0.7610 - val_loss: 0.6137 - val_accuracy: 0.6574 -  
val_auc: 0.7448  
Epoch 908/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6060 -  
accuracy: 0.6878 - auc: 0.7565 - val_loss: 0.6226 - val_accuracy: 0.6543 -  
val_auc: 0.7332  
Epoch 909/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6049 -  
accuracy: 0.6933 - auc: 0.7601 - val_loss: 0.6180 - val_accuracy: 0.6613 -  
val_auc: 0.7386  
Epoch 910/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6048 -  
accuracy: 0.6918 - auc: 0.7578 - val_loss: 0.6189 - val_accuracy: 0.6636 -  
val_auc: 0.7376  
Epoch 911/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6051 -  
accuracy: 0.6898 - auc: 0.7578 - val_loss: 0.6149 - val_accuracy: 0.6628 -  
val_auc: 0.7433  
Epoch 912/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6066 -  
accuracy: 0.6883 - auc: 0.7564 - val_loss: 0.6166 - val_accuracy: 0.6574 -  
val_auc: 0.7412  
Epoch 913/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6047 -  
accuracy: 0.6943 - auc: 0.7590 - val_loss: 0.6193 - val_accuracy: 0.6644 -  
val_auc: 0.7380  
Epoch 914/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6058 -  
accuracy: 0.6896 - auc: 0.7579 - val_loss: 0.6141 - val_accuracy: 0.6574 -  
val_auc: 0.7438  
Epoch 915/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6046 -  
accuracy: 0.6928 - auc: 0.7582 - val_loss: 0.6172 - val_accuracy: 0.6597 -  
val_auc: 0.7405  
Epoch 916/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6040 -  
accuracy: 0.6903 - auc: 0.7594 - val_loss: 0.6181 - val_accuracy: 0.6605 -  
val_auc: 0.7394  
Epoch 917/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6037 -  
accuracy: 0.6915 - auc: 0.7589 - val_loss: 0.6190 - val_accuracy: 0.6613 -  
val_auc: 0.7376  
Epoch 918/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6036 -
```

```
accuracy: 0.6913 - auc: 0.7596 - val_loss: 0.6154 - val_accuracy: 0.6582 -  
val_auc: 0.7429  
Epoch 919/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6035 -  
accuracy: 0.6946 - auc: 0.7607 - val_loss: 0.6166 - val_accuracy: 0.6628 -  
val_auc: 0.7406  
Epoch 920/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6030 -  
accuracy: 0.6956 - auc: 0.7599 - val_loss: 0.6185 - val_accuracy: 0.6566 -  
val_auc: 0.7380  
Epoch 921/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6040 -  
accuracy: 0.6948 - auc: 0.7606 - val_loss: 0.6177 - val_accuracy: 0.6597 -  
val_auc: 0.7389  
Epoch 922/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6044 -  
accuracy: 0.6903 - auc: 0.7588 - val_loss: 0.6133 - val_accuracy: 0.6605 -  
val_auc: 0.7450  
Epoch 923/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6053 -  
accuracy: 0.6911 - auc: 0.7584 - val_loss: 0.6185 - val_accuracy: 0.6628 -  
val_auc: 0.7382  
Epoch 924/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6068 -  
accuracy: 0.6911 - auc: 0.7567 - val_loss: 0.6205 - val_accuracy: 0.6597 -  
val_auc: 0.7359  
Epoch 925/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6044 -  
accuracy: 0.6949 - auc: 0.7585 - val_loss: 0.6201 - val_accuracy: 0.6613 -  
val_auc: 0.7362  
Epoch 926/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6043 -  
accuracy: 0.6936 - auc: 0.7588 - val_loss: 0.6161 - val_accuracy: 0.6590 -  
val_auc: 0.7417  
Epoch 927/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6045 -  
accuracy: 0.6931 - auc: 0.7586 - val_loss: 0.6210 - val_accuracy: 0.6605 -  
val_auc: 0.7352  
Epoch 928/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6040 -  
accuracy: 0.6901 - auc: 0.7591 - val_loss: 0.6197 - val_accuracy: 0.6605 -  
val_auc: 0.7368  
Epoch 929/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6052 -  
accuracy: 0.6898 - auc: 0.7584 - val_loss: 0.6169 - val_accuracy: 0.6582 -  
val_auc: 0.7403  
Epoch 930/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6041 -
```

```
accuracy: 0.6923 - auc: 0.7585 - val_loss: 0.6178 - val_accuracy: 0.6613 -  
val_auc: 0.7392  
Epoch 931/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6051 -  
accuracy: 0.6892 - auc: 0.7586 - val_loss: 0.6165 - val_accuracy: 0.6651 -  
val_auc: 0.7409  
Epoch 932/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6030 -  
accuracy: 0.6939 - auc: 0.7599 - val_loss: 0.6183 - val_accuracy: 0.6644 -  
val_auc: 0.7385  
Epoch 933/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6029 -  
accuracy: 0.6941 - auc: 0.7604 - val_loss: 0.6178 - val_accuracy: 0.6636 -  
val_auc: 0.7393  
Epoch 934/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6051 -  
accuracy: 0.6931 - auc: 0.7592 - val_loss: 0.6190 - val_accuracy: 0.6605 -  
val_auc: 0.7375  
Epoch 935/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6023 -  
accuracy: 0.6888 - auc: 0.7600 - val_loss: 0.6171 - val_accuracy: 0.6636 -  
val_auc: 0.7405  
Epoch 936/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6043 -  
accuracy: 0.6918 - auc: 0.7595 - val_loss: 0.6158 - val_accuracy: 0.6597 -  
val_auc: 0.7422  
Epoch 937/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6060 -  
accuracy: 0.6920 - auc: 0.7569 - val_loss: 0.6179 - val_accuracy: 0.6590 -  
val_auc: 0.7391  
Epoch 938/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6035 -  
accuracy: 0.6958 - auc: 0.7593 - val_loss: 0.6172 - val_accuracy: 0.6628 -  
val_auc: 0.7397  
Epoch 939/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6038 -  
accuracy: 0.6910 - auc: 0.7603 - val_loss: 0.6183 - val_accuracy: 0.6597 -  
val_auc: 0.7381  
Epoch 940/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6050 -  
accuracy: 0.6877 - auc: 0.7572 - val_loss: 0.6150 - val_accuracy: 0.6574 -  
val_auc: 0.7428  
Epoch 941/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6037 -  
accuracy: 0.6867 - auc: 0.7593 - val_loss: 0.6124 - val_accuracy: 0.6605 -  
val_auc: 0.7458  
Epoch 942/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6022 -
```

```
accuracy: 0.6978 - auc: 0.7622 - val_loss: 0.6170 - val_accuracy: 0.6597 -  
val_auc: 0.7398  
Epoch 943/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6047 -  
accuracy: 0.6915 - auc: 0.7589 - val_loss: 0.6187 - val_accuracy: 0.6613 -  
val_auc: 0.7385  
Epoch 944/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6042 -  
accuracy: 0.6944 - auc: 0.7586 - val_loss: 0.6182 - val_accuracy: 0.6644 -  
val_auc: 0.7386  
Epoch 945/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6028 -  
accuracy: 0.6923 - auc: 0.7602 - val_loss: 0.6142 - val_accuracy: 0.6605 -  
val_auc: 0.7436  
Epoch 946/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6025 -  
accuracy: 0.6956 - auc: 0.7613 - val_loss: 0.6174 - val_accuracy: 0.6582 -  
val_auc: 0.7394  
Epoch 947/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6032 -  
accuracy: 0.6946 - auc: 0.7597 - val_loss: 0.6154 - val_accuracy: 0.6597 -  
val_auc: 0.7429  
Epoch 948/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6035 -  
accuracy: 0.6913 - auc: 0.7606 - val_loss: 0.6156 - val_accuracy: 0.6582 -  
val_auc: 0.7418  
Epoch 949/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6021 -  
accuracy: 0.6986 - auc: 0.7618 - val_loss: 0.6186 - val_accuracy: 0.6597 -  
val_auc: 0.7379  
Epoch 950/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6030 -  
accuracy: 0.6911 - auc: 0.7599 - val_loss: 0.6182 - val_accuracy: 0.6613 -  
val_auc: 0.7384  
Epoch 951/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6028 -  
accuracy: 0.6877 - auc: 0.7604 - val_loss: 0.6187 - val_accuracy: 0.6597 -  
val_auc: 0.7379  
Epoch 952/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6040 -  
accuracy: 0.6964 - auc: 0.7598 - val_loss: 0.6166 - val_accuracy: 0.6620 -  
val_auc: 0.7408  
Epoch 953/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6041 -  
accuracy: 0.6898 - auc: 0.7598 - val_loss: 0.6152 - val_accuracy: 0.6590 -  
val_auc: 0.7427  
Epoch 954/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6048 -
```

```
accuracy: 0.6916 - auc: 0.7581 - val_loss: 0.6187 - val_accuracy: 0.6620 -  
val_auc: 0.7377  
Epoch 955/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6037 -  
accuracy: 0.6911 - auc: 0.7587 - val_loss: 0.6193 - val_accuracy: 0.6605 -  
val_auc: 0.7370  
Epoch 956/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6055 -  
accuracy: 0.6911 - auc: 0.7586 - val_loss: 0.6160 - val_accuracy: 0.6651 -  
val_auc: 0.7412  
Epoch 957/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6044 -  
accuracy: 0.6905 - auc: 0.7590 - val_loss: 0.6202 - val_accuracy: 0.6590 -  
val_auc: 0.7359  
Epoch 958/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6034 -  
accuracy: 0.6887 - auc: 0.7598 - val_loss: 0.6183 - val_accuracy: 0.6605 -  
val_auc: 0.7382  
Epoch 959/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6030 -  
accuracy: 0.6928 - auc: 0.7608 - val_loss: 0.6172 - val_accuracy: 0.6628 -  
val_auc: 0.7394  
Epoch 960/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6031 -  
accuracy: 0.6913 - auc: 0.7605 - val_loss: 0.6163 - val_accuracy: 0.6620 -  
val_auc: 0.7413  
Epoch 961/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6045 -  
accuracy: 0.6887 - auc: 0.7579 - val_loss: 0.6145 - val_accuracy: 0.6590 -  
val_auc: 0.7434  
Epoch 962/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6046 -  
accuracy: 0.6923 - auc: 0.7592 - val_loss: 0.6181 - val_accuracy: 0.6636 -  
val_auc: 0.7384  
Epoch 963/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6022 -  
accuracy: 0.6939 - auc: 0.7612 - val_loss: 0.6167 - val_accuracy: 0.6605 -  
val_auc: 0.7408  
Epoch 964/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6039 -  
accuracy: 0.6880 - auc: 0.7594 - val_loss: 0.6172 - val_accuracy: 0.6613 -  
val_auc: 0.7404  
Epoch 965/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6033 -  
accuracy: 0.6939 - auc: 0.7608 - val_loss: 0.6177 - val_accuracy: 0.6628 -  
val_auc: 0.7391  
Epoch 966/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6023 -
```

```
accuracy: 0.6915 - auc: 0.7602 - val_loss: 0.6142 - val_accuracy: 0.6582 -  
val_auc: 0.7436  
Epoch 967/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6031 -  
accuracy: 0.6930 - auc: 0.7605 - val_loss: 0.6183 - val_accuracy: 0.6605 -  
val_auc: 0.7382  
Epoch 968/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6041 -  
accuracy: 0.6991 - auc: 0.7591 - val_loss: 0.6174 - val_accuracy: 0.6582 -  
val_auc: 0.7401  
Epoch 969/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6028 -  
accuracy: 0.6895 - auc: 0.7605 - val_loss: 0.6147 - val_accuracy: 0.6566 -  
val_auc: 0.7431  
Epoch 970/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6033 -  
accuracy: 0.6936 - auc: 0.7609 - val_loss: 0.6159 - val_accuracy: 0.6620 -  
val_auc: 0.7422  
Epoch 971/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6043 -  
accuracy: 0.6888 - auc: 0.7587 - val_loss: 0.6143 - val_accuracy: 0.6582 -  
val_auc: 0.7433  
Epoch 972/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6045 -  
accuracy: 0.6923 - auc: 0.7592 - val_loss: 0.6162 - val_accuracy: 0.6636 -  
val_auc: 0.7406  
Epoch 973/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6030 -  
accuracy: 0.6931 - auc: 0.7610 - val_loss: 0.6200 - val_accuracy: 0.6597 -  
val_auc: 0.7363  
Epoch 974/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6035 -  
accuracy: 0.6941 - auc: 0.7593 - val_loss: 0.6181 - val_accuracy: 0.6582 -  
val_auc: 0.7386  
Epoch 975/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6023 -  
accuracy: 0.6968 - auc: 0.7613 - val_loss: 0.6159 - val_accuracy: 0.6628 -  
val_auc: 0.7414  
Epoch 976/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6019 -  
accuracy: 0.6954 - auc: 0.7618 - val_loss: 0.6156 - val_accuracy: 0.6636 -  
val_auc: 0.7416  
Epoch 977/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6028 -  
accuracy: 0.6887 - auc: 0.7609 - val_loss: 0.6141 - val_accuracy: 0.6605 -  
val_auc: 0.7436  
Epoch 978/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6038 -
```

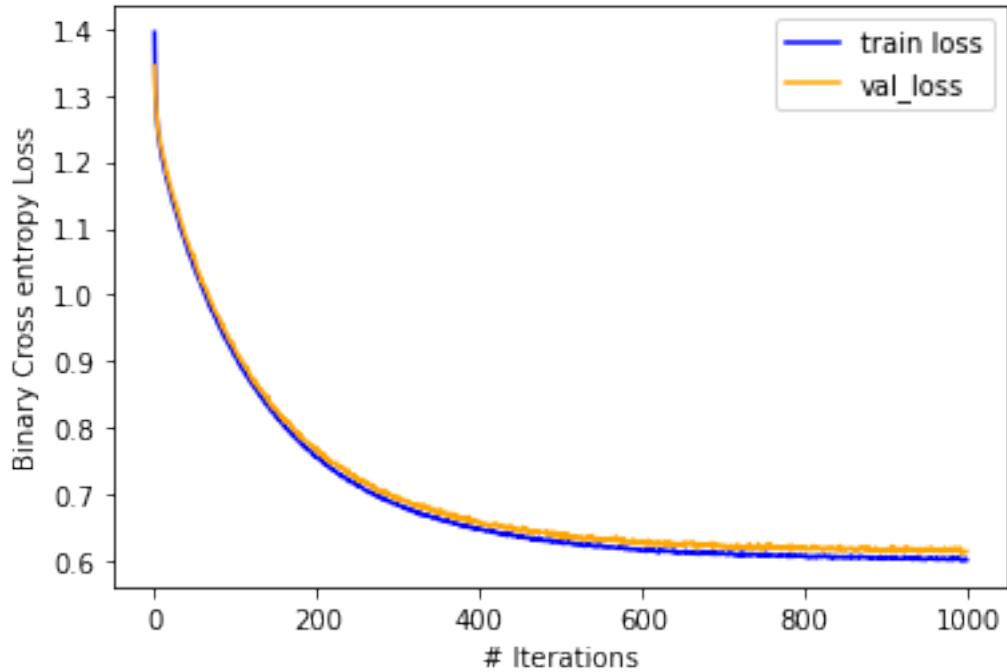
```
accuracy: 0.6910 - auc: 0.7596 - val_loss: 0.6154 - val_accuracy: 0.6628 -  
val_auc: 0.7417  
Epoch 979/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6026 -  
accuracy: 0.6939 - auc: 0.7608 - val_loss: 0.6171 - val_accuracy: 0.6597 -  
val_auc: 0.7395  
Epoch 980/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6032 -  
accuracy: 0.6895 - auc: 0.7596 - val_loss: 0.6190 - val_accuracy: 0.6535 -  
val_auc: 0.7369  
Epoch 981/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6025 -  
accuracy: 0.6958 - auc: 0.7609 - val_loss: 0.6150 - val_accuracy: 0.6613 -  
val_auc: 0.7426  
Epoch 982/1000  
189/189 [=====] - 4s 19ms/step - loss: 0.6053 -  
accuracy: 0.6925 - auc: 0.7576 - val_loss: 0.6181 - val_accuracy: 0.6574 -  
val_auc: 0.7387  
Epoch 983/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6020 -  
accuracy: 0.6951 - auc: 0.7621 - val_loss: 0.6139 - val_accuracy: 0.6620 -  
val_auc: 0.7439  
Epoch 984/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6037 -  
accuracy: 0.6956 - auc: 0.7602 - val_loss: 0.6196 - val_accuracy: 0.6590 -  
val_auc: 0.7367  
Epoch 985/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6030 -  
accuracy: 0.6923 - auc: 0.7602 - val_loss: 0.6166 - val_accuracy: 0.6620 -  
val_auc: 0.7407  
Epoch 986/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6031 -  
accuracy: 0.6883 - auc: 0.7604 - val_loss: 0.6189 - val_accuracy: 0.6582 -  
val_auc: 0.7372  
Epoch 987/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6047 -  
accuracy: 0.6900 - auc: 0.7583 - val_loss: 0.6195 - val_accuracy: 0.6566 -  
val_auc: 0.7366  
Epoch 988/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6026 -  
accuracy: 0.6923 - auc: 0.7611 - val_loss: 0.6151 - val_accuracy: 0.6644 -  
val_auc: 0.7428  
Epoch 989/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6046 -  
accuracy: 0.6949 - auc: 0.7588 - val_loss: 0.6187 - val_accuracy: 0.6605 -  
val_auc: 0.7377  
Epoch 990/1000  
189/189 [=====] - 4s 20ms/step - loss: 0.6030 -
```

```
accuracy: 0.6920 - auc: 0.7607 - val_loss: 0.6156 - val_accuracy: 0.6620 -
val_auc: 0.7422
Epoch 991/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6041 -
accuracy: 0.6898 - auc: 0.7595 - val_loss: 0.6172 - val_accuracy: 0.6620 -
val_auc: 0.7396
Epoch 992/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6038 -
accuracy: 0.6888 - auc: 0.7588 - val_loss: 0.6203 - val_accuracy: 0.6605 -
val_auc: 0.7354
Epoch 993/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6012 -
accuracy: 0.6954 - auc: 0.7620 - val_loss: 0.6117 - val_accuracy: 0.6659 -
val_auc: 0.7464
Epoch 994/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6022 -
accuracy: 0.6911 - auc: 0.7601 - val_loss: 0.6160 - val_accuracy: 0.6628 -
val_auc: 0.7410
Epoch 995/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6009 -
accuracy: 0.6948 - auc: 0.7626 - val_loss: 0.6149 - val_accuracy: 0.6620 -
val_auc: 0.7429
Epoch 996/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6013 -
accuracy: 0.6928 - auc: 0.7614 - val_loss: 0.6134 - val_accuracy: 0.6605 -
val_auc: 0.7442
Epoch 997/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6035 -
accuracy: 0.6938 - auc: 0.7610 - val_loss: 0.6135 - val_accuracy: 0.6628 -
val_auc: 0.7446
Epoch 998/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6041 -
accuracy: 0.6906 - auc: 0.7585 - val_loss: 0.6135 - val_accuracy: 0.6636 -
val_auc: 0.7441
Epoch 999/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6040 -
accuracy: 0.6916 - auc: 0.7589 - val_loss: 0.6146 - val_accuracy: 0.6628 -
val_auc: 0.7432
Epoch 1000/1000
189/189 [=====] - 4s 20ms/step - loss: 0.6016 -
accuracy: 0.6966 - auc: 0.7620 - val_loss: 0.6146 - val_accuracy: 0.6590 -
val_auc: 0.7429
training done
```

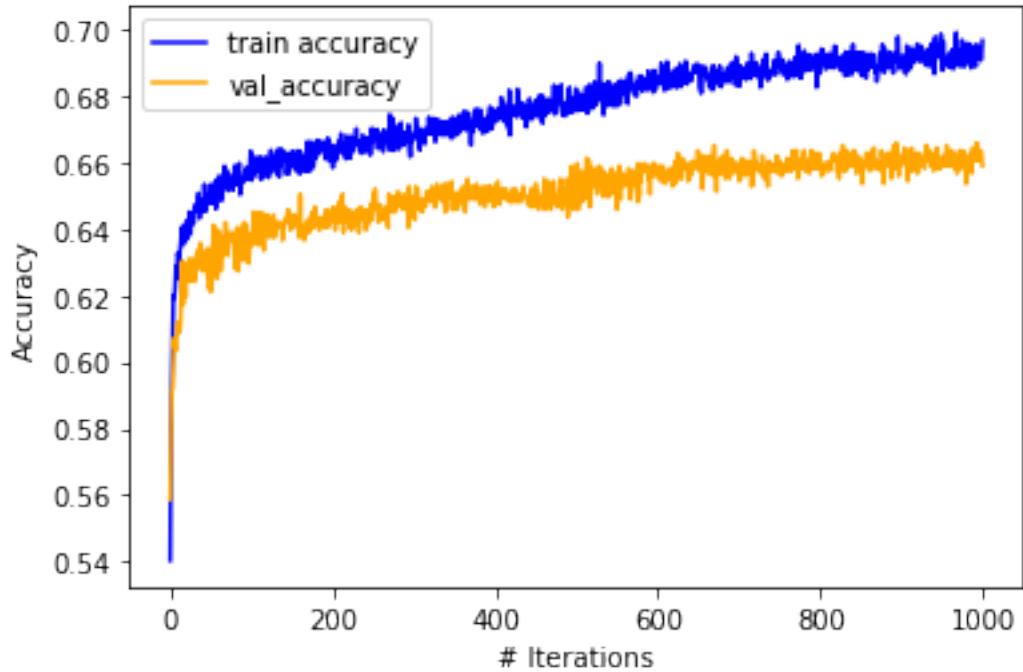
We will evaluate with same batch size of 64

```
[ ]: folder = "/content/drive/MyDrive/sCNNEPOCHS1000(Class_weight)/"
```

```
[ ]: import pickle
[ ]: # with open(folder + 'trainHistoryDict', 'wb') as file_pi:
#       # pickle.dump(history, file_pi)
[ ]: history = pickle.load(open(folder + 'trainHistoryDict', "rb"))
model = tf.keras.models.load_model(folder + 'model.h5')
[ ]: if same:
    evaluation = model.evaluate(X_test, Y_test, batch_size=32)
    print("loss =", evaluation[0])
    print("binary accuracy = {}%".format(evaluation[1] * 100))
    print("# iterations =", len(history.history['loss']))
41/41 [=====] - 13s 327ms/step - loss: 0.5928 -
accuracy: 0.6898 - auc: 0.7703
loss = 0.5928244590759277
binary accuracy = 68.9814805984497%
# iterations = 1000
[ ]: evaluation
[ ]: [0.5928244590759277, 0.6898148059844971, 0.7703403234481812]
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Binary Cross entropy Loss')
plt.plot(history.history['loss'], color='blue', label = "train loss")
plt.plot(history.history['val_loss'], color='orange', label = "val_loss")
plt.savefig(folder + "loss.jpeg")
plt.legend()
# plt.grid()
plt.show()
```



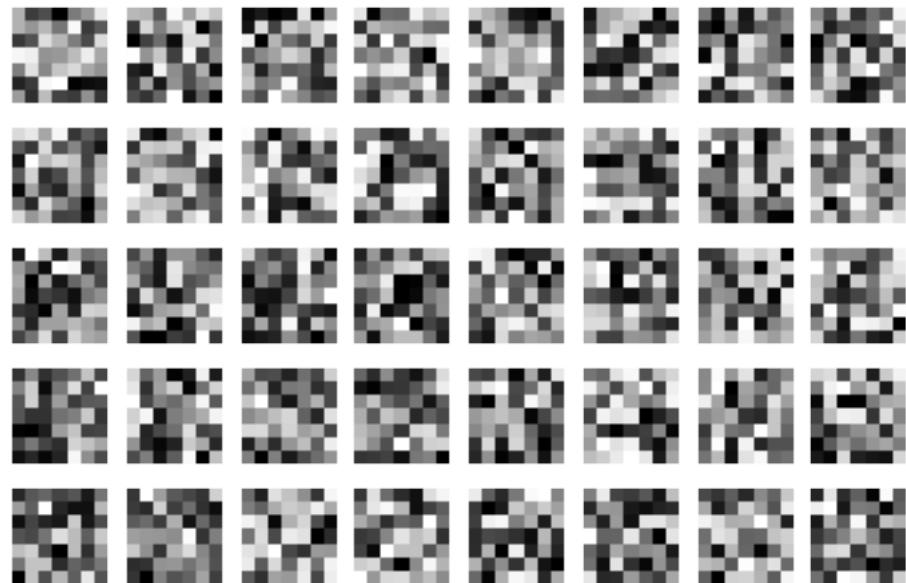
```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Accuracy')
plt.plot(history.history['accuracy'], color='blue',label = "train accuracy")
plt.plot(history.history['val_accuracy'], color='orange',label = "val_accuracy")
# plt.grid()
plt.legend()
plt.savefig(folder + "accuracy.jpeg")
plt.show()
```



```
[ ]: fig, ax = plt.subplots(5,8)

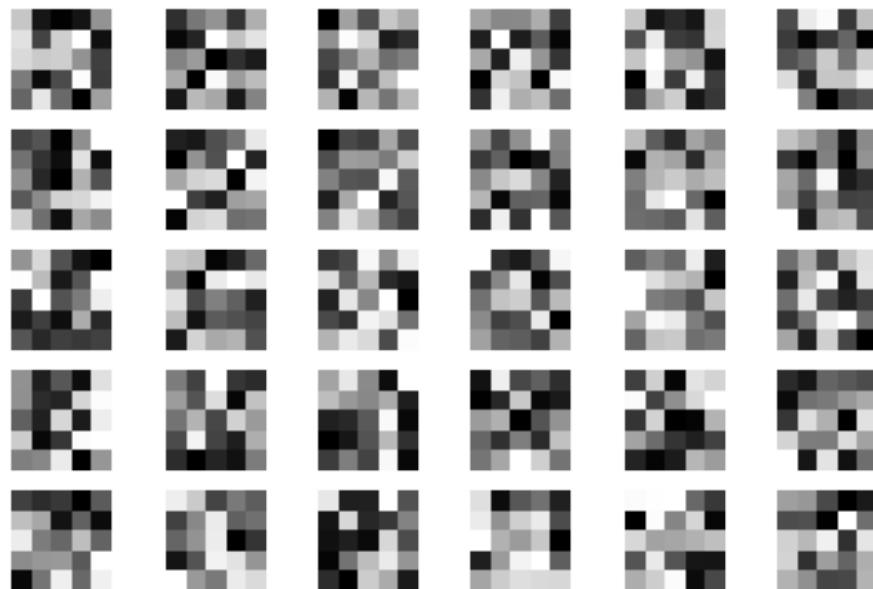
for i in range(40):
    ax[i%5, i//5].imshow(model.layers[0].weights[0] [:,:,0,i].numpy(),cmap='gray')
    ax[i%5, i//5].set_axis_off()
fig.suptitle('Layer 1')
plt.show()
```

Layer 1



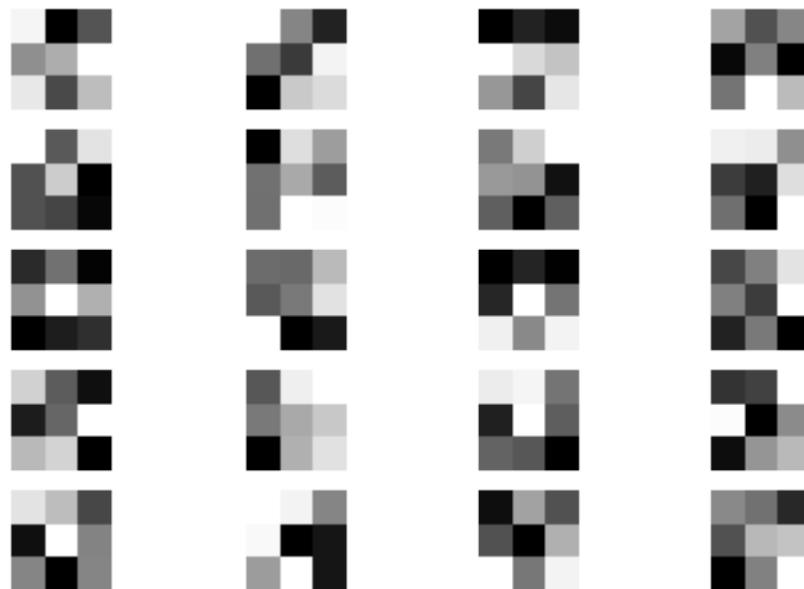
```
[ ]: fig, ax = plt.subplots(5,6, squeeze=False)
for i in range(30):
    ax[i%5, i//5].imshow(model.layers[3].weights[0][:,:,0,i].numpy(), cmap = 'gray')
    ax[i%5, i//5].set_axis_off()
fig.suptitle('Layer 2')
plt.show()
```

Layer 2



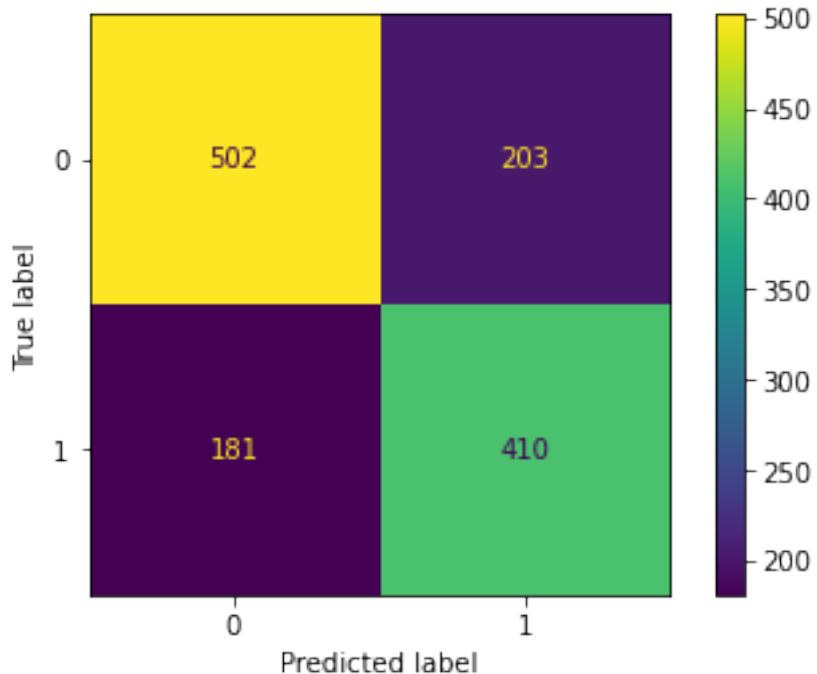
```
[ ]: fig, ax = plt.subplots(5, 4)
for i in range(20):
    ax[i%5, i//5].imshow(model.layers[6].weights[0] [:,:,0,i].numpy(), 'gray')
    ax[i%5, i//5].set_axis_off()
fig.suptitle('Layer 3')
plt.show()
```

Layer 3



```
[ ]: ConfusionMatrixDisplay(confusion_matrix(tf.argmax(Y_test, axis = 1), tf.  
    ↪argmax(model.predict(X_test), axis = 1))).plot()  
plt.savefig( folder + "confusionmatrix.jpeg")  
plt.show()
```

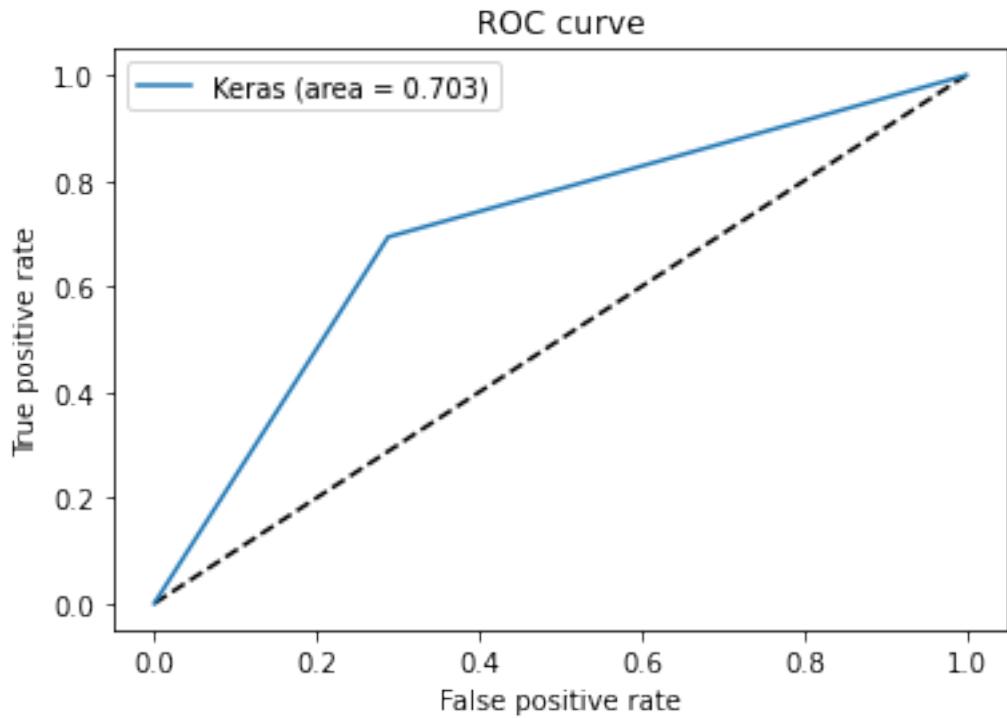
41/41 [=====] - 0s 7ms/step



```
[ ]: model.save( folder + 'model.h5')

[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = tf.argmax(model.predict(X_test),axis = 1)
fpr_keras, tpr_keras, thresholds_keras = roc_curve(tf.argmax(Y_test,axis = 1), y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.jpeg")
plt.show()
```

41/41 [=====] - 0s 6ms/step



[]:

Model1_SCNN_CedarTraining

November 14, 2024

1 Shallow Convolution Model

This model has only 3 convolution layers with filters of size 7, 5, 3 and 3 max pooling layers.

```
[ ]: from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import LeakyReLU, Softmax
from keras.layers import Conv2D, Activation, Input, Dropout, Lambda, Flatten, Dense
from keras.layers import Dense, Flatten, Reshape, Activation, MaxPool2D
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model

[ ]: import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

```
[ ]: from google.colab import drive  
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call  
drive.mount("/content/drive", force_remount=True).
```

```
There are 3 databases available with me. * BhSig100Bengali * BhSig160Hindi * Cedar * Sig-  
comp2011
```

We can select any of the three available.

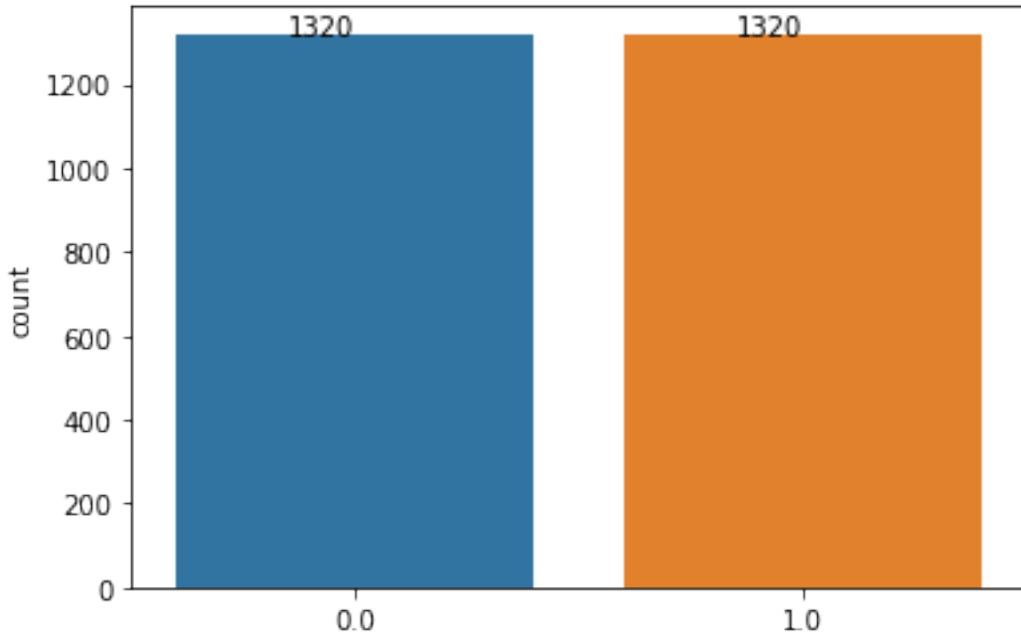
```
[ ]: # database = input('Database Name: ')  
# Models\model1_scnn\database\bhsig100bengali_128x64.h5  
# database = database.lower() + '_128x64.h5'  
# BASE_DIR = os.path.join(os.pardir, os.pardir)  
metadata = {}  
file = "/content/drive/MyDrive/cedar_128x643.h5"  
print(file)  
try:  
    with File(file, 'r') as hdf:  
        X = np.array(hdf.get('X'))  
        Y = np.array(hdf.get('Y'))  
        S = np.array(hdf.get('S'))  
except Exception as ex:  
    template = "An exception of type {0} occurred. Arguments:\n{1!r}"  
    message = template.format(type(ex).__name__, ex.args)  
    print(message)  
X = X / 255.0  
Y = Y * 1.0  
# Y = one_hot(Y * 1.0, depth=2)  
# Y = reshape(Y, (-1, 2))  
print("Feature shape =", X.shape)  
print("Label shape =", Y.shape)
```

```
/content/drive/MyDrive/cedar_128x643.h5
```

```
Feature shape = (2640, 64, 128, 3)
```

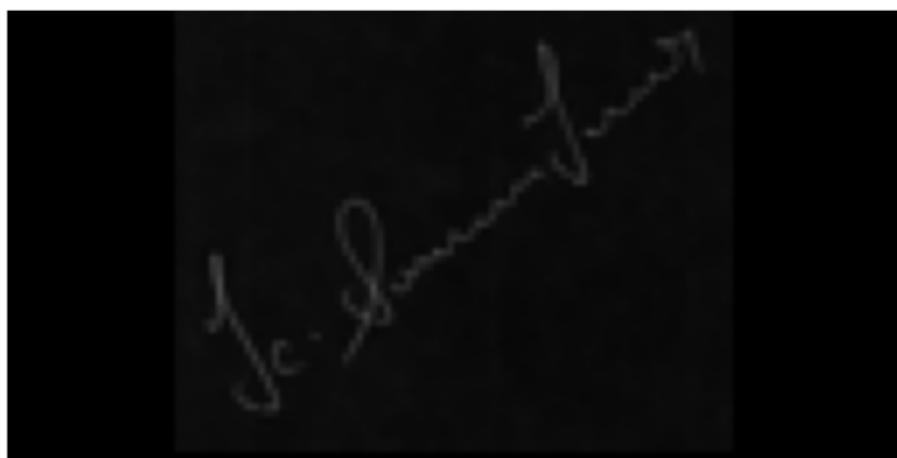
```
Label shape = (2640, 1)
```

```
[ ]: ax = sns.countplot(x = Y.reshape(Y.shape[0]))  
for p in ax.patches:  
    ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.  
    ↵01))
```

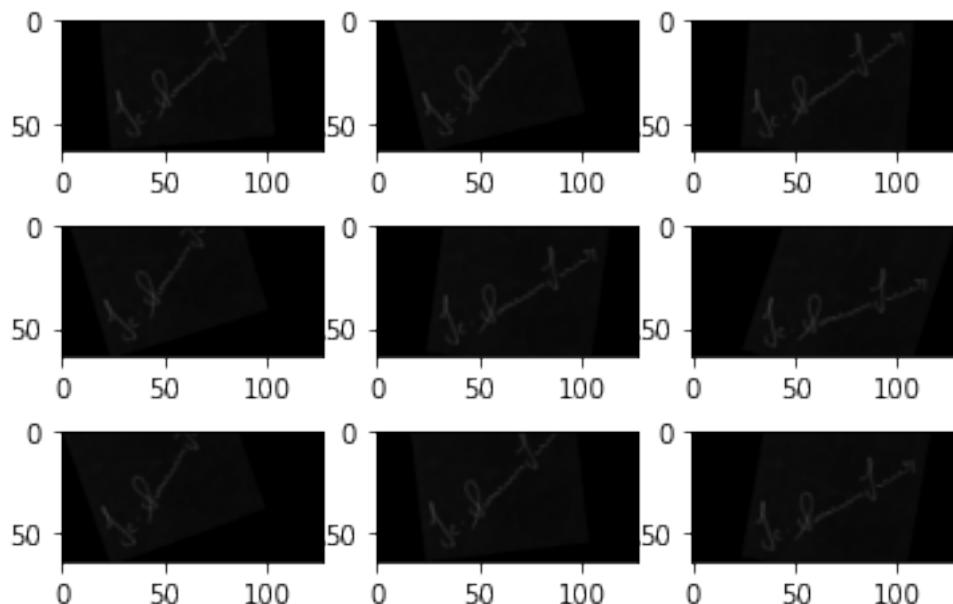


```
[ ]: x = np.random.randint(0,X.shape[0])
img = X[x]
print(x)
print(img.shape)
plt.imshow((img * 255).astype(np.uint8))
plt.axis("off")
plt.show()
```

410
(64, 128, 3)



```
[ ]: # load the image
# img = load_img('bird.jpg')
# convert to numpy array
data = img_to_array(img *255)
# expand dimension to one sample
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(rotation_range=20)
# prepare iterator
it = datagen.flow(samples, batch_size=1)
# generate samples and plot
for i in range(9):
    # define subplot
    plt.subplot(330 + 1 + i)
    # generate batch of images
    batch = it.next()
    # convert to unsigned integers for viewing
    image = batch[0].astype('uint8')
    # plot raw pixel data
    plt.imshow(image)
# pyplot.axis("off")
# show the figure
plt.show()
```



```
[ ]: import sklearn
class_weights = sklearn.utils.class_weight.compute_class_weight(class_weight = "balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict
```

```
[ ]: {0: 1.0, 1: 1.0}
```

```
[ ]: seed=randint(10)
metadata['split_seed']=seed
print('seed =', seed)
indices = permutation(X.shape[0])
# print(X)
same = True
if same:
    m = int(0.70 * X.shape[0])
    n = int(0.15 * X.shape[0])
    training_id, validation_id, test_id = indices[:m], indices[m: m + n], indices[m+n:]
    X_train, X_test, X_validate = X[training_id], X[test_id], X[validation_id]
    Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]
else:
    m = int(0.70 * X.shape[0])
    training_id, validation_id = indices[:m], indices[m:]
    X_train, X_validate = X[training_id], X[validation_id]
    Y_train, Y_validate = Y[training_id], Y[validation_id]
print("Shape of Features in training set =", X_train.shape)
print("Shape of Labels in training set =", Y_train.shape)
print("Shape of Features in testing set =", X_test.shape)
print("Shape of Labels in testing set =", Y_test.shape)

del Y,X
```

```
seed = 3
Shape of Features in training set = (1847, 64, 128, 3)
Shape of Labels in training set = (1847, 1)
Shape of Features in testing set = (397, 64, 128, 3)
Shape of Labels in testing set = (397, 1)
```

```
[ ]: #One hot Encoding
Y_train = one_hot(Y_train, depth=2)
Y_train = reshape(Y_train, (-1, 2))

Y_test = one_hot(Y_test, depth=2)
Y_test = reshape(Y_test, (-1, 2))

Y_validate = one_hot(Y_validate, depth=2)
Y_validate = reshape(Y_validate, (-1, 2))
```

```
[ ]: print("Shape of Labels in training set =", Y_train.shape)

Shape of Labels in training set = (1847, 2)

[ ]: # X_train, X_test, Y_train, Y_test = train_test_split(X, Y,train_size = 0.7,random_state = 42)

[ ]: # X_test, X_validate, Y_test, Y_validate = train_test_split(X_test, Y_test,train_size = 0.5, random_state = 42)

[ ]: # del Y,X

[ ]: seed =randint(10)
metadata['init_seed']=seed
print('seed='+str(seed))
weights1 = np.random.rand(7,7,3,40)
bias1 = np.random.rand(40)
weights2 = np.random.rand(5,5,40,30)
bias2 = np.random.rand(30)
weights3 = np.random.rand(3,3,30,20)
bias3 = np.random.rand(20)

model = Sequential()
model.add(Input(shape = (64, 128, 3),name='input'))
conv1 = Conv2D(40, 7, strides = 1,activation='relu',name='conv1',use_bias =True)
model.add(conv1)
model.add(BatchNormalization(epsilon = 1e-08, axis = 1,name = 'batchnorm1',momentum = 0.9))
model.add(MaxPool2D(strides=2,name = "Pool1"))

conv2 = Conv2D(30, 5,strides = 1, activation='relu', name='conv2',use_bias =True)
model.add(conv2)
model.add(BatchNormalization(epsilon = 1e-08, axis = 1,name = 'batchnorm2',momentum = 0.9))
model.add(MaxPool2D(strides=3,name = "Pool2"))

conv3 = Conv2D(20, 3, strides = 1,activation='relu',name='conv3',use_bias =True)
model.add(conv3)
model.add(BatchNormalization(epsilon = 1e-08, axis = 1,name = 'batchnorm3',momentum = 0.9))
model.add(MaxPool2D(strides=3,name = "Pool3"))

model.add(Flatten())
```

```
model.add(Dense(32, activation='relu',  
    ↪kernel_initializer=glorot_uniform(seed=seed), kernel_regularizer='l2'))  
model.add(Dense(2, activation='softmax',  
    ↪kernel_initializer=glorot_uniform(seed=seed), kernel_regularizer='l2'))
```

```
seed=3
```

```
[ ]: model.layers
```

```
[ ]: [<keras.layers.convolutional.conv2d.Conv2D at 0x7fabbf4ce310>,  
       <keras.layers.normalization.batch_normalization.BatchNormalization at  
       0x7fabbf2a0310>,  
       <keras.layers.pooling.max_pooling2d.MaxPooling2D at 0x7fabbf2a0b20>,  
       <keras.layers.convolutional.conv2d.Conv2D at 0x7fabd649bac0>,  
       <keras.layers.normalization.batch_normalization.BatchNormalization at  
       0x7fabbf2a0040>,  
       <keras.layers.pooling.max_pooling2d.MaxPooling2D at 0x7fabbf2f06a0>,  
       <keras.layers.convolutional.conv2d.Conv2D at 0x7fabbf2a0430>,  
       <keras.layers.normalization.batch_normalization.BatchNormalization at  
       0x7fabbf2f04c0>,  
       <keras.layers.pooling.max_pooling2d.MaxPooling2D at 0x7fabd6495c70>,  
       <keras.layers.reshape.flatten.Flatten at 0x7fabbf284f10>,  
       <keras.layers.core.dense.Dense at 0x7fabbf25e3a0>,  
       <keras.layers.core.dense.Dense at 0x7fabbf25e1f0>]
```

```
[ ]: model.layers[0].set_weights([weights1,bias1])  
model.layers[3].set_weights([weights2,bias2])  
model.layers[6].set_weights([weights3,bias3])
```

```
[ ]: model.layers[6].get_weights()[0].shape
```

```
[ ]: (3, 3, 30, 20)
```

```
[ ]: # len(conv1.get_weights())  
# [len(a) for a in conv1.get_weights()]
```

The summary of the model is given below:

```
[ ]: print(model.summary())
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv1 (Conv2D)	(None, 58, 122, 40)	5920
batchnorm1 (BatchNormalizat ion)	(None, 58, 122, 40)	232

Pool1 (MaxPooling2D)	(None, 29, 61, 40)	0
conv2 (Conv2D)	(None, 25, 57, 30)	30030
batchnorm2 (BatchNormalizat ion)	(None, 25, 57, 30)	100
Pool2 (MaxPooling2D)	(None, 8, 19, 30)	0
conv3 (Conv2D)	(None, 6, 17, 20)	5420
batchnorm3 (BatchNormalizat ion)	(None, 6, 17, 20)	24
Pool3 (MaxPooling2D)	(None, 2, 6, 20)	0
flatten_1 (Flatten)	(None, 240)	0
dense_2 (Dense)	(None, 32)	7712
dense_3 (Dense)	(None, 2)	66
=====		
Total params:	49,504	
Trainable params:	49,326	
Non-trainable params:	178	

None

In comile process * Optimizer is *Adam* * Loss is *Binary Cross Entropy Loss* * Metrics involve *Binary Accuracy*

```
[ ]: with tf.device('/device:GPU:0'):
    SGD = tf.keras.optimizers.SGD(learning_rate = 1e-04,momentum = 0.9,clipvalue= 7)
    model.compile(optimizer = SGD, loss='binary_crossentropy',metrics=['accuracy','AUC'])
    #, 'TruePositives', 'TrueNegatives', 'FalsePositives', 'FalseNegatives'])
```

1.1 Batch Size Testing

```
[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),  
            epochs = 3,  
            batch_size=8)
```

Epoch 1/3
231/231 [=====] - 14s 16ms/step - loss: 1.1496 -
accuracy: 0.7899 - auc: 0.8437 - val_loss: 1.0247 - val_accuracy: 0.8561 -

```

val_auc: 0.9169
Epoch 2/3
231/231 [=====] - 2s 9ms/step - loss: 1.0231 -
accuracy: 0.8343 - auc: 0.9011 - val_loss: 0.9569 - val_accuracy: 0.8813 -
val_auc: 0.9373
Epoch 3/3
231/231 [=====] - 2s 8ms/step - loss: 0.9842 -
accuracy: 0.8462 - auc: 0.9156 - val_loss: 0.9421 - val_accuracy: 0.8788 -
val_auc: 0.9364

[ ]: <keras.callbacks.History at 0x7fabd644aee0>

[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
batch_size=16)

116/116 [=====] - 2s 13ms/step - loss: 0.9540 -
accuracy: 0.8598 - auc: 0.9247 - val_loss: 0.9397 - val_accuracy: 0.8712 -
val_auc: 0.9375

[ ]: <keras.callbacks.History at 0x7fabbe7b8040>

[ ]: model.fit(X_train, Y_train, epochs = 10, batch_size = 32,
validation_data=(X_validate, Y_validate),
shuffle=True,class_weight = class_weight_dict)

Epoch 1/10
58/58 [=====] - 2s 25ms/step - loss: 0.9316 - accuracy:
0.8782 - auc: 0.9343 - val_loss: 0.9356 - val_accuracy: 0.8788 - val_auc: 0.9346
Epoch 2/10
58/58 [=====] - 1s 19ms/step - loss: 0.9300 - accuracy:
0.8744 - auc: 0.9320 - val_loss: 0.9417 - val_accuracy: 0.8687 - val_auc: 0.9316
Epoch 3/10
58/58 [=====] - 1s 19ms/step - loss: 0.9268 - accuracy:
0.8782 - auc: 0.9345 - val_loss: 0.9327 - val_accuracy: 0.8788 - val_auc: 0.9341
Epoch 4/10
58/58 [=====] - 1s 19ms/step - loss: 0.9199 - accuracy:
0.8831 - auc: 0.9360 - val_loss: 0.9204 - val_accuracy: 0.8788 - val_auc: 0.9390
Epoch 5/10
58/58 [=====] - 1s 19ms/step - loss: 0.9112 - accuracy:
0.8820 - auc: 0.9383 - val_loss: 0.9271 - val_accuracy: 0.8813 - val_auc: 0.9343
Epoch 6/10
58/58 [=====] - 1s 19ms/step - loss: 0.9007 - accuracy:
0.8841 - auc: 0.9426 - val_loss: 0.9080 - val_accuracy: 0.8838 - val_auc: 0.9424
Epoch 7/10
58/58 [=====] - 1s 20ms/step - loss: 0.9080 - accuracy:
0.8809 - auc: 0.9373 - val_loss: 0.9046 - val_accuracy: 0.8838 - val_auc: 0.9443
Epoch 8/10
58/58 [=====] - 1s 19ms/step - loss: 0.9078 - accuracy:
0.8798 - auc: 0.9387 - val_loss: 0.9027 - val_accuracy: 0.8788 - val_auc: 0.9429

```

```

Epoch 9/10
58/58 [=====] - 1s 19ms/step - loss: 0.8920 - accuracy: 0.8912 - auc: 0.9438 - val_loss: 0.9073 - val_accuracy: 0.8788 - val_auc: 0.9407
Epoch 10/10
58/58 [=====] - 1s 19ms/step - loss: 0.8891 - accuracy: 0.8885 - auc: 0.9444 - val_loss: 0.9086 - val_accuracy: 0.8788 - val_auc: 0.9390

[ ]: <keras.callbacks.History at 0x7fabbf7f3340>

[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
batch_size=64)

29/29 [=====] - 2s 58ms/step - loss: 0.8827 - accuracy: 0.8939 - auc: 0.9465 - val_loss: 0.8980 - val_accuracy: 0.8813 - val_auc: 0.9429

[ ]: <keras.callbacks.History at 0x7fabbf684970>

[ ]: model.fit(x = X_train,y = Y_train,validation_data = (X_validate,Y_validate),
batch_size=128)

15/15 [=====] - 2s 88ms/step - loss: 0.8849 - accuracy: 0.8890 - auc: 0.9452 - val_loss: 0.8971 - val_accuracy: 0.8813 - val_auc: 0.9430

[ ]: <keras.callbacks.History at 0x7fabbd63736d0>

```

1.2 In fit process

- epochs = 20
- batch size = 32

```

[ ]: with tf.device('/device:GPU:0'):
    SGD = tf.keras.optimizers.SGD(learning_rate = 1e-04,momentum = 0.9,clipvalue=7)
    model.compile(optimizer = SGD, loss='binary_crossentropy',metrics=['accuracy','AUC'])
    #, 'TruePositives', 'TrueNegatives', 'FalsePositives', 'FalseNegatives'])

[ ]: history = model.fit(X_train, Y_train, epochs = 1000, batch_size = 32,validation_data=(X_validate, Y_validate),shuffle=True
                        ,class_weight = class_weight_dict)
print('training done')

```

```

Epoch 1/1000
58/58 [=====] - 2s 26ms/step - loss: 2.0789 - accuracy: 0.1684 - auc: 0.1457 - val_loss: 1.6440 - val_accuracy: 0.2273 - val_auc: 0.1999
Epoch 2/1000
58/58 [=====] - 1s 19ms/step - loss: 1.3932 - accuracy: 0.5143 - auc: 0.4877 - val_loss: 1.2151 - val_accuracy: 0.7980 - val_auc: 0.8647
Epoch 3/1000
58/58 [=====] - 1s 19ms/step - loss: 1.1680 - accuracy:

```

0.7991 - auc: 0.8757 - val_loss: 1.0983 - val_accuracy: 0.8434 - val_auc: 0.9156
Epoch 4/1000
58/58 [=====] - 1s 19ms/step - loss: 1.0895 - accuracy: 0.8360 - auc: 0.9041 - val_loss: 1.0495 - val_accuracy: 0.8561 - val_auc: 0.9190
Epoch 5/1000
58/58 [=====] - 1s 19ms/step - loss: 1.0487 - accuracy: 0.8441 - auc: 0.9097 - val_loss: 1.0195 - val_accuracy: 0.8586 - val_auc: 0.9226
Epoch 6/1000
58/58 [=====] - 1s 23ms/step - loss: 1.0262 - accuracy: 0.8446 - auc: 0.9112 - val_loss: 0.9974 - val_accuracy: 0.8737 - val_auc: 0.9252
Epoch 7/1000
58/58 [=====] - 2s 27ms/step - loss: 1.0043 - accuracy: 0.8560 - auc: 0.9149 - val_loss: 0.9811 - val_accuracy: 0.8586 - val_auc: 0.9285
Epoch 8/1000
58/58 [=====] - 2s 27ms/step - loss: 0.9971 - accuracy: 0.8419 - auc: 0.9137 - val_loss: 0.9766 - val_accuracy: 0.8662 - val_auc: 0.9277
Epoch 9/1000
58/58 [=====] - 1s 21ms/step - loss: 0.9766 - accuracy: 0.8571 - auc: 0.9216 - val_loss: 0.9593 - val_accuracy: 0.8662 - val_auc: 0.9319
Epoch 10/1000
58/58 [=====] - 1s 19ms/step - loss: 0.9668 - accuracy: 0.8576 - auc: 0.9232 - val_loss: 0.9547 - val_accuracy: 0.8561 - val_auc: 0.9313
Epoch 11/1000
58/58 [=====] - 1s 19ms/step - loss: 0.9614 - accuracy: 0.8511 - auc: 0.9232 - val_loss: 0.9406 - val_accuracy: 0.8712 - val_auc: 0.9360
Epoch 12/1000
58/58 [=====] - 1s 20ms/step - loss: 0.9534 - accuracy: 0.8554 - auc: 0.9247 - val_loss: 0.9400 - val_accuracy: 0.8611 - val_auc: 0.9358
Epoch 13/1000
58/58 [=====] - 1s 20ms/step - loss: 0.9413 - accuracy: 0.8636 - auc: 0.9289 - val_loss: 0.9247 - val_accuracy: 0.8636 - val_auc: 0.9377
Epoch 14/1000
58/58 [=====] - 1s 20ms/step - loss: 0.9458 - accuracy: 0.8587 - auc: 0.9242 - val_loss: 0.9182 - val_accuracy: 0.8737 - val_auc: 0.9407
Epoch 15/1000
58/58 [=====] - 1s 19ms/step - loss: 0.9349 - accuracy: 0.8706 - auc: 0.9282 - val_loss: 0.9145 - val_accuracy: 0.8712 - val_auc: 0.9411
Epoch 16/1000
58/58 [=====] - 1s 19ms/step - loss: 0.9348 - accuracy: 0.8576 - auc: 0.9274 - val_loss: 0.9084 - val_accuracy: 0.8763 - val_auc: 0.9442
Epoch 17/1000
58/58 [=====] - 1s 20ms/step - loss: 0.9297 - accuracy: 0.8619 - auc: 0.9283 - val_loss: 0.8963 - val_accuracy: 0.8838 - val_auc: 0.9476
Epoch 18/1000
58/58 [=====] - 1s 19ms/step - loss: 0.9210 - accuracy: 0.8663 - auc: 0.9320 - val_loss: 0.8925 - val_accuracy: 0.8838 - val_auc: 0.9487
Epoch 19/1000
58/58 [=====] - 1s 19ms/step - loss: 0.9191 - accuracy:

0.8657 - auc: 0.9314 - val_loss: 0.8878 - val_accuracy: 0.8788 - val_auc: 0.9489
Epoch 20/1000
58/58 [=====] - 1s 20ms/step - loss: 0.9207 - accuracy: 0.8679 - auc: 0.9299 - val_loss: 0.8828 - val_accuracy: 0.8864 - val_auc: 0.9502
Epoch 21/1000
58/58 [=====] - 1s 20ms/step - loss: 0.9086 - accuracy: 0.8701 - auc: 0.9349 - val_loss: 0.8832 - val_accuracy: 0.8864 - val_auc: 0.9501
Epoch 22/1000
58/58 [=====] - 1s 19ms/step - loss: 0.9034 - accuracy: 0.8738 - auc: 0.9365 - val_loss: 0.8754 - val_accuracy: 0.8914 - val_auc: 0.9523
Epoch 23/1000
58/58 [=====] - 1s 19ms/step - loss: 0.9109 - accuracy: 0.8652 - auc: 0.9315 - val_loss: 0.8696 - val_accuracy: 0.8889 - val_auc: 0.9534
Epoch 24/1000
58/58 [=====] - 1s 19ms/step - loss: 0.9056 - accuracy: 0.8690 - auc: 0.9319 - val_loss: 0.8712 - val_accuracy: 0.8889 - val_auc: 0.9518
Epoch 25/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8964 - accuracy: 0.8722 - auc: 0.9369 - val_loss: 0.8612 - val_accuracy: 0.8939 - val_auc: 0.9543
Epoch 26/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8952 - accuracy: 0.8738 - auc: 0.9358 - val_loss: 0.8611 - val_accuracy: 0.8889 - val_auc: 0.9550
Epoch 27/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8970 - accuracy: 0.8706 - auc: 0.9347 - val_loss: 0.8676 - val_accuracy: 0.8939 - val_auc: 0.9512
Epoch 28/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8851 - accuracy: 0.8701 - auc: 0.9394 - val_loss: 0.8539 - val_accuracy: 0.8965 - val_auc: 0.9557
Epoch 29/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8871 - accuracy: 0.8738 - auc: 0.9368 - val_loss: 0.8558 - val_accuracy: 0.8939 - val_auc: 0.9540
Epoch 30/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8739 - accuracy: 0.8771 - auc: 0.9428 - val_loss: 0.8557 - val_accuracy: 0.8939 - val_auc: 0.9536
Epoch 31/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8685 - accuracy: 0.8782 - auc: 0.9440 - val_loss: 0.8396 - val_accuracy: 0.8939 - val_auc: 0.9574
Epoch 32/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8728 - accuracy: 0.8766 - auc: 0.9401 - val_loss: 0.8514 - val_accuracy: 0.8939 - val_auc: 0.9537
Epoch 33/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8720 - accuracy: 0.8841 - auc: 0.9414 - val_loss: 0.8478 - val_accuracy: 0.8965 - val_auc: 0.9550
Epoch 34/1000
58/58 [=====] - 1s 20ms/step - loss: 0.8686 - accuracy: 0.8847 - auc: 0.9410 - val_loss: 0.8397 - val_accuracy: 0.8914 - val_auc: 0.9561
Epoch 35/1000
58/58 [=====] - 1s 23ms/step - loss: 0.8625 - accuracy:

0.8814 - auc: 0.9438 - val_loss: 0.8307 - val_accuracy: 0.9040 - val_auc: 0.9597
Epoch 36/1000
58/58 [=====] - 1s 23ms/step - loss: 0.8651 - accuracy: 0.8803 - auc: 0.9422 - val_loss: 0.8312 - val_accuracy: 0.9015 - val_auc: 0.9591
Epoch 37/1000
58/58 [=====] - 1s 21ms/step - loss: 0.8653 - accuracy: 0.8744 - auc: 0.9408 - val_loss: 0.8304 - val_accuracy: 0.8990 - val_auc: 0.9587
Epoch 38/1000
58/58 [=====] - 1s 20ms/step - loss: 0.8598 - accuracy: 0.8820 - auc: 0.9429 - val_loss: 0.8307 - val_accuracy: 0.8965 - val_auc: 0.9577
Epoch 39/1000
58/58 [=====] - 1s 20ms/step - loss: 0.8593 - accuracy: 0.8717 - auc: 0.9433 - val_loss: 0.8265 - val_accuracy: 0.9015 - val_auc: 0.9592
Epoch 40/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8527 - accuracy: 0.8841 - auc: 0.9444 - val_loss: 0.8210 - val_accuracy: 0.9015 - val_auc: 0.9607
Epoch 41/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8652 - accuracy: 0.8760 - auc: 0.9384 - val_loss: 0.8214 - val_accuracy: 0.8990 - val_auc: 0.9600
Epoch 42/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8502 - accuracy: 0.8814 - auc: 0.9435 - val_loss: 0.8176 - val_accuracy: 0.8990 - val_auc: 0.9601
Epoch 43/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8484 - accuracy: 0.8841 - auc: 0.9452 - val_loss: 0.8150 - val_accuracy: 0.9040 - val_auc: 0.9610
Epoch 44/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8487 - accuracy: 0.8814 - auc: 0.9445 - val_loss: 0.8091 - val_accuracy: 0.9066 - val_auc: 0.9627
Epoch 45/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8312 - accuracy: 0.8879 - auc: 0.9498 - val_loss: 0.8161 - val_accuracy: 0.9015 - val_auc: 0.9596
Epoch 46/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8416 - accuracy: 0.8782 - auc: 0.9457 - val_loss: 0.8138 - val_accuracy: 0.8965 - val_auc: 0.9590
Epoch 47/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8314 - accuracy: 0.8879 - auc: 0.9492 - val_loss: 0.8002 - val_accuracy: 0.9066 - val_auc: 0.9643
Epoch 48/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8314 - accuracy: 0.8879 - auc: 0.9488 - val_loss: 0.7959 - val_accuracy: 0.9116 - val_auc: 0.9651
Epoch 49/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8303 - accuracy: 0.8906 - auc: 0.9474 - val_loss: 0.8031 - val_accuracy: 0.9015 - val_auc: 0.9609
Epoch 50/1000
58/58 [=====] - 1s 20ms/step - loss: 0.8326 - accuracy: 0.8809 - auc: 0.9468 - val_loss: 0.7901 - val_accuracy: 0.9091 - val_auc: 0.9660
Epoch 51/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8308 - accuracy:

0.8868 - auc: 0.9470 - val_loss: 0.8036 - val_accuracy: 0.8990 - val_auc: 0.9605
Epoch 52/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8278 - accuracy: 0.8863 - auc: 0.9473 - val_loss: 0.7899 - val_accuracy: 0.9116 - val_auc: 0.9655
Epoch 53/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8235 - accuracy: 0.8879 - auc: 0.9485 - val_loss: 0.7833 - val_accuracy: 0.9141 - val_auc: 0.9677
Epoch 54/1000
58/58 [=====] - 1s 20ms/step - loss: 0.8307 - accuracy: 0.8852 - auc: 0.9452 - val_loss: 0.7781 - val_accuracy: 0.9141 - val_auc: 0.9685
Epoch 55/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8170 - accuracy: 0.8923 - auc: 0.9498 - val_loss: 0.7988 - val_accuracy: 0.8990 - val_auc: 0.9585
Epoch 56/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8177 - accuracy: 0.8814 - auc: 0.9500 - val_loss: 0.7779 - val_accuracy: 0.9066 - val_auc: 0.9676
Epoch 57/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8059 - accuracy: 0.8933 - auc: 0.9530 - val_loss: 0.7740 - val_accuracy: 0.9141 - val_auc: 0.9687
Epoch 58/1000
58/58 [=====] - 1s 20ms/step - loss: 0.8084 - accuracy: 0.8923 - auc: 0.9523 - val_loss: 0.7749 - val_accuracy: 0.9141 - val_auc: 0.9678
Epoch 59/1000
58/58 [=====] - 1s 19ms/step - loss: 0.8073 - accuracy: 0.8858 - auc: 0.9525 - val_loss: 0.7748 - val_accuracy: 0.9116 - val_auc: 0.9668
Epoch 60/1000
58/58 [=====] - 1s 20ms/step - loss: 0.8125 - accuracy: 0.8825 - auc: 0.9500 - val_loss: 0.7658 - val_accuracy: 0.9091 - val_auc: 0.9694
Epoch 61/1000
58/58 [=====] - 1s 20ms/step - loss: 0.8052 - accuracy: 0.8912 - auc: 0.9514 - val_loss: 0.7735 - val_accuracy: 0.9066 - val_auc: 0.9661
Epoch 62/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7980 - accuracy: 0.8950 - auc: 0.9539 - val_loss: 0.7714 - val_accuracy: 0.9091 - val_auc: 0.9677
Epoch 63/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7959 - accuracy: 0.9020 - auc: 0.9539 - val_loss: 0.7704 - val_accuracy: 0.9116 - val_auc: 0.9666
Epoch 64/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7988 - accuracy: 0.8933 - auc: 0.9534 - val_loss: 0.7677 - val_accuracy: 0.9091 - val_auc: 0.9663
Epoch 65/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7989 - accuracy: 0.8868 - auc: 0.9527 - val_loss: 0.7706 - val_accuracy: 0.9015 - val_auc: 0.9648
Epoch 66/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7858 - accuracy: 0.8960 - auc: 0.9571 - val_loss: 0.7636 - val_accuracy: 0.9091 - val_auc: 0.9675
Epoch 67/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7869 - accuracy:

0.8998 - auc: 0.9546 - val_loss: 0.7486 - val_accuracy: 0.9167 - val_auc: 0.9716
Epoch 68/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7902 - accuracy: 0.8960 - auc: 0.9554 - val_loss: 0.7649 - val_accuracy: 0.9040 - val_auc: 0.9669
Epoch 69/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7918 - accuracy: 0.8944 - auc: 0.9530 - val_loss: 0.7696 - val_accuracy: 0.9015 - val_auc: 0.9629
Epoch 70/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7889 - accuracy: 0.8944 - auc: 0.9539 - val_loss: 0.7549 - val_accuracy: 0.9167 - val_auc: 0.9693
Epoch 71/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7865 - accuracy: 0.9031 - auc: 0.9525 - val_loss: 0.7466 - val_accuracy: 0.9192 - val_auc: 0.9714
Epoch 72/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7816 - accuracy: 0.8982 - auc: 0.9552 - val_loss: 0.7431 - val_accuracy: 0.9192 - val_auc: 0.9734
Epoch 73/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7760 - accuracy: 0.8977 - auc: 0.9572 - val_loss: 0.7506 - val_accuracy: 0.9116 - val_auc: 0.9689
Epoch 74/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7689 - accuracy: 0.9004 - auc: 0.9597 - val_loss: 0.7546 - val_accuracy: 0.9091 - val_auc: 0.9669
Epoch 75/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7812 - accuracy: 0.8917 - auc: 0.9534 - val_loss: 0.7517 - val_accuracy: 0.9040 - val_auc: 0.9686
Epoch 76/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7690 - accuracy: 0.9009 - auc: 0.9573 - val_loss: 0.7370 - val_accuracy: 0.9192 - val_auc: 0.9721
Epoch 77/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7735 - accuracy: 0.9009 - auc: 0.9553 - val_loss: 0.7386 - val_accuracy: 0.9167 - val_auc: 0.9712
Epoch 78/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7696 - accuracy: 0.8971 - auc: 0.9570 - val_loss: 0.7322 - val_accuracy: 0.9167 - val_auc: 0.9739
Epoch 79/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7676 - accuracy: 0.9020 - auc: 0.9568 - val_loss: 0.7293 - val_accuracy: 0.9167 - val_auc: 0.9726
Epoch 80/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7617 - accuracy: 0.9036 - auc: 0.9596 - val_loss: 0.7421 - val_accuracy: 0.9066 - val_auc: 0.9697
Epoch 81/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7646 - accuracy: 0.9015 - auc: 0.9561 - val_loss: 0.7325 - val_accuracy: 0.9192 - val_auc: 0.9706
Epoch 82/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7677 - accuracy: 0.8988 - auc: 0.9554 - val_loss: 0.7194 - val_accuracy: 0.9192 - val_auc: 0.9731
Epoch 83/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7651 - accuracy:

0.8955 - auc: 0.9568 - val_loss: 0.7299 - val_accuracy: 0.9091 - val_auc: 0.9721
Epoch 84/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7493 - accuracy: 0.9004 - auc: 0.9624 - val_loss: 0.7337 - val_accuracy: 0.9066 - val_auc: 0.9703
Epoch 85/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7538 - accuracy: 0.9053 - auc: 0.9587 - val_loss: 0.7232 - val_accuracy: 0.9217 - val_auc: 0.9743
Epoch 86/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7505 - accuracy: 0.9025 - auc: 0.9603 - val_loss: 0.7245 - val_accuracy: 0.9141 - val_auc: 0.9729
Epoch 87/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7485 - accuracy: 0.9031 - auc: 0.9604 - val_loss: 0.7224 - val_accuracy: 0.9167 - val_auc: 0.9729
Epoch 88/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7515 - accuracy: 0.9009 - auc: 0.9596 - val_loss: 0.7119 - val_accuracy: 0.9242 - val_auc: 0.9761
Epoch 89/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7496 - accuracy: 0.8977 - auc: 0.9607 - val_loss: 0.7203 - val_accuracy: 0.9192 - val_auc: 0.9730
Epoch 90/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7450 - accuracy: 0.9069 - auc: 0.9598 - val_loss: 0.7210 - val_accuracy: 0.9116 - val_auc: 0.9727
Epoch 91/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7514 - accuracy: 0.8982 - auc: 0.9579 - val_loss: 0.7283 - val_accuracy: 0.9091 - val_auc: 0.9687
Epoch 92/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7403 - accuracy: 0.9074 - auc: 0.9602 - val_loss: 0.7256 - val_accuracy: 0.9116 - val_auc: 0.9689
Epoch 93/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7386 - accuracy: 0.9134 - auc: 0.9606 - val_loss: 0.7126 - val_accuracy: 0.9192 - val_auc: 0.9747
Epoch 94/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7399 - accuracy: 0.9085 - auc: 0.9605 - val_loss: 0.7122 - val_accuracy: 0.9192 - val_auc: 0.9742
Epoch 95/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7361 - accuracy: 0.9090 - auc: 0.9605 - val_loss: 0.7123 - val_accuracy: 0.9192 - val_auc: 0.9737
Epoch 96/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7405 - accuracy: 0.9058 - auc: 0.9597 - val_loss: 0.7018 - val_accuracy: 0.9217 - val_auc: 0.9762
Epoch 97/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7281 - accuracy: 0.9096 - auc: 0.9623 - val_loss: 0.7064 - val_accuracy: 0.9242 - val_auc: 0.9752
Epoch 98/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7436 - accuracy: 0.8955 - auc: 0.9580 - val_loss: 0.7101 - val_accuracy: 0.9167 - val_auc: 0.9731
Epoch 99/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7253 - accuracy:

```
0.9101 - auc: 0.9632 - val_loss: 0.6929 - val_accuracy: 0.9242 - val_auc: 0.9784
Epoch 100/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7188 - accuracy: 0.9128 - auc: 0.9652 - val_loss: 0.6992 - val_accuracy: 0.9293 - val_auc: 0.9757
Epoch 101/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7202 - accuracy: 0.9042 - auc: 0.9653 - val_loss: 0.7047 - val_accuracy: 0.9217 - val_auc: 0.9737
Epoch 102/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7371 - accuracy: 0.9080 - auc: 0.9569 - val_loss: 0.6814 - val_accuracy: 0.9293 - val_auc: 0.9790
Epoch 103/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7208 - accuracy: 0.9107 - auc: 0.9632 - val_loss: 0.6896 - val_accuracy: 0.9268 - val_auc: 0.9771
Epoch 104/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7132 - accuracy: 0.9166 - auc: 0.9655 - val_loss: 0.6856 - val_accuracy: 0.9293 - val_auc: 0.9780
Epoch 105/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7284 - accuracy: 0.9047 - auc: 0.9599 - val_loss: 0.6909 - val_accuracy: 0.9293 - val_auc: 0.9762
Epoch 106/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7126 - accuracy: 0.9188 - auc: 0.9647 - val_loss: 0.6712 - val_accuracy: 0.9343 - val_auc: 0.9823
Epoch 107/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7120 - accuracy: 0.9090 - auc: 0.9656 - val_loss: 0.6893 - val_accuracy: 0.9242 - val_auc: 0.9761
Epoch 108/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7086 - accuracy: 0.9199 - auc: 0.9641 - val_loss: 0.6810 - val_accuracy: 0.9318 - val_auc: 0.9789
Epoch 109/1000
58/58 [=====] - 1s 19ms/step - loss: 0.7055 - accuracy: 0.9139 - auc: 0.9672 - val_loss: 0.6750 - val_accuracy: 0.9293 - val_auc: 0.9798
Epoch 110/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7158 - accuracy: 0.9107 - auc: 0.9615 - val_loss: 0.6787 - val_accuracy: 0.9318 - val_auc: 0.9786
Epoch 111/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7117 - accuracy: 0.9085 - auc: 0.9635 - val_loss: 0.6674 - val_accuracy: 0.9293 - val_auc: 0.9799
Epoch 112/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7066 - accuracy: 0.9155 - auc: 0.9634 - val_loss: 0.6719 - val_accuracy: 0.9268 - val_auc: 0.9805
Epoch 113/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7017 - accuracy: 0.9107 - auc: 0.9656 - val_loss: 0.6784 - val_accuracy: 0.9318 - val_auc: 0.9777
Epoch 114/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7092 - accuracy: 0.9112 - auc: 0.9617 - val_loss: 0.6757 - val_accuracy: 0.9318 - val_auc: 0.9780
Epoch 115/1000
58/58 [=====] - 1s 20ms/step - loss: 0.7110 - accuracy:
```

0.9155 - auc: 0.9613 - val_loss: 0.6725 - val_accuracy: 0.9293 - val_auc: 0.9784
Epoch 116/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6996 - accuracy: 0.9161 - auc: 0.9650 - val_loss: 0.6826 - val_accuracy: 0.9217 - val_auc: 0.9739
Epoch 117/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6917 - accuracy: 0.9150 - auc: 0.9688 - val_loss: 0.6708 - val_accuracy: 0.9318 - val_auc: 0.9787
Epoch 118/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6973 - accuracy: 0.9107 - auc: 0.9652 - val_loss: 0.6677 - val_accuracy: 0.9268 - val_auc: 0.9798
Epoch 119/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6867 - accuracy: 0.9166 - auc: 0.9701 - val_loss: 0.6615 - val_accuracy: 0.9318 - val_auc: 0.9807
Epoch 120/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6839 - accuracy: 0.9166 - auc: 0.9698 - val_loss: 0.6651 - val_accuracy: 0.9268 - val_auc: 0.9791
Epoch 121/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6946 - accuracy: 0.9166 - auc: 0.9650 - val_loss: 0.6664 - val_accuracy: 0.9318 - val_auc: 0.9786
Epoch 122/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6827 - accuracy: 0.9145 - auc: 0.9696 - val_loss: 0.6579 - val_accuracy: 0.9343 - val_auc: 0.9808
Epoch 123/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6849 - accuracy: 0.9177 - auc: 0.9677 - val_loss: 0.6642 - val_accuracy: 0.9318 - val_auc: 0.9789
Epoch 124/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6791 - accuracy: 0.9188 - auc: 0.9701 - val_loss: 0.6590 - val_accuracy: 0.9343 - val_auc: 0.9797
Epoch 125/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6742 - accuracy: 0.9215 - auc: 0.9718 - val_loss: 0.6592 - val_accuracy: 0.9318 - val_auc: 0.9795
Epoch 126/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6802 - accuracy: 0.9220 - auc: 0.9669 - val_loss: 0.6684 - val_accuracy: 0.9268 - val_auc: 0.9751
Epoch 127/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6801 - accuracy: 0.9123 - auc: 0.9685 - val_loss: 0.6541 - val_accuracy: 0.9394 - val_auc: 0.9799
Epoch 128/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6779 - accuracy: 0.9188 - auc: 0.9682 - val_loss: 0.6439 - val_accuracy: 0.9394 - val_auc: 0.9831
Epoch 129/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6813 - accuracy: 0.9199 - auc: 0.9670 - val_loss: 0.6406 - val_accuracy: 0.9419 - val_auc: 0.9838
Epoch 130/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6852 - accuracy: 0.9177 - auc: 0.9625 - val_loss: 0.6600 - val_accuracy: 0.9394 - val_auc: 0.9776
Epoch 131/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6761 - accuracy:

0.9155 - auc: 0.9678 - val_loss: 0.6370 - val_accuracy: 0.9394 - val_auc: 0.9843
Epoch 132/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6727 - accuracy: 0.9172 - auc: 0.9700 - val_loss: 0.6477 - val_accuracy: 0.9369 - val_auc: 0.9809
Epoch 133/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6654 - accuracy: 0.9210 - auc: 0.9703 - val_loss: 0.6359 - val_accuracy: 0.9394 - val_auc: 0.9834
Epoch 134/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6620 - accuracy: 0.9210 - auc: 0.9713 - val_loss: 0.6489 - val_accuracy: 0.9369 - val_auc: 0.9798
Epoch 135/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6596 - accuracy: 0.9215 - auc: 0.9726 - val_loss: 0.6343 - val_accuracy: 0.9394 - val_auc: 0.9840
Epoch 136/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6706 - accuracy: 0.9242 - auc: 0.9675 - val_loss: 0.6390 - val_accuracy: 0.9369 - val_auc: 0.9820
Epoch 137/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6645 - accuracy: 0.9188 - auc: 0.9686 - val_loss: 0.6252 - val_accuracy: 0.9470 - val_auc: 0.9853
Epoch 138/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6581 - accuracy: 0.9226 - auc: 0.9714 - val_loss: 0.6378 - val_accuracy: 0.9369 - val_auc: 0.9821
Epoch 139/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6595 - accuracy: 0.9226 - auc: 0.9700 - val_loss: 0.6275 - val_accuracy: 0.9394 - val_auc: 0.9841
Epoch 140/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6568 - accuracy: 0.9237 - auc: 0.9705 - val_loss: 0.6163 - val_accuracy: 0.9470 - val_auc: 0.9846
Epoch 141/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6517 - accuracy: 0.9215 - auc: 0.9722 - val_loss: 0.6384 - val_accuracy: 0.9394 - val_auc: 0.9806
Epoch 142/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6652 - accuracy: 0.9182 - auc: 0.9670 - val_loss: 0.6161 - val_accuracy: 0.9470 - val_auc: 0.9867
Epoch 143/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6552 - accuracy: 0.9220 - auc: 0.9709 - val_loss: 0.6390 - val_accuracy: 0.9394 - val_auc: 0.9800
Epoch 144/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6522 - accuracy: 0.9220 - auc: 0.9707 - val_loss: 0.6246 - val_accuracy: 0.9394 - val_auc: 0.9839
Epoch 145/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6414 - accuracy: 0.9253 - auc: 0.9742 - val_loss: 0.6398 - val_accuracy: 0.9343 - val_auc: 0.9774
Epoch 146/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6430 - accuracy: 0.9274 - auc: 0.9736 - val_loss: 0.6186 - val_accuracy: 0.9419 - val_auc: 0.9844
Epoch 147/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6420 - accuracy:

0.9269 - auc: 0.9738 - val_loss: 0.6180 - val_accuracy: 0.9444 - val_auc: 0.9847
Epoch 148/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6465 - accuracy: 0.9269 - auc: 0.9717 - val_loss: 0.6032 - val_accuracy: 0.9444 - val_auc: 0.9861
Epoch 149/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6427 - accuracy: 0.9220 - auc: 0.9730 - val_loss: 0.6149 - val_accuracy: 0.9394 - val_auc: 0.9844
Epoch 150/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6418 - accuracy: 0.9242 - auc: 0.9712 - val_loss: 0.6056 - val_accuracy: 0.9495 - val_auc: 0.9872
Epoch 151/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6416 - accuracy: 0.9291 - auc: 0.9714 - val_loss: 0.6138 - val_accuracy: 0.9394 - val_auc: 0.9845
Epoch 152/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6383 - accuracy: 0.9323 - auc: 0.9721 - val_loss: 0.6144 - val_accuracy: 0.9419 - val_auc: 0.9843
Epoch 153/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6414 - accuracy: 0.9258 - auc: 0.9711 - val_loss: 0.6048 - val_accuracy: 0.9444 - val_auc: 0.9867
Epoch 154/1000
58/58 [=====] - 1s 21ms/step - loss: 0.6353 - accuracy: 0.9329 - auc: 0.9721 - val_loss: 0.6102 - val_accuracy: 0.9419 - val_auc: 0.9845
Epoch 155/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6364 - accuracy: 0.9247 - auc: 0.9720 - val_loss: 0.6069 - val_accuracy: 0.9444 - val_auc: 0.9854
Epoch 156/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6284 - accuracy: 0.9269 - auc: 0.9750 - val_loss: 0.6022 - val_accuracy: 0.9444 - val_auc: 0.9865
Epoch 157/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6368 - accuracy: 0.9280 - auc: 0.9712 - val_loss: 0.6079 - val_accuracy: 0.9419 - val_auc: 0.9847
Epoch 158/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6251 - accuracy: 0.9334 - auc: 0.9751 - val_loss: 0.5944 - val_accuracy: 0.9495 - val_auc: 0.9878
Epoch 159/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6287 - accuracy: 0.9312 - auc: 0.9724 - val_loss: 0.5970 - val_accuracy: 0.9444 - val_auc: 0.9863
Epoch 160/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6255 - accuracy: 0.9302 - auc: 0.9731 - val_loss: 0.5923 - val_accuracy: 0.9470 - val_auc: 0.9854
Epoch 161/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6362 - accuracy: 0.9215 - auc: 0.9702 - val_loss: 0.5908 - val_accuracy: 0.9495 - val_auc: 0.9879
Epoch 162/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6191 - accuracy: 0.9307 - auc: 0.9758 - val_loss: 0.5995 - val_accuracy: 0.9419 - val_auc: 0.9855
Epoch 163/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6255 - accuracy:

0.9226 - auc: 0.9733 - val_loss: 0.5893 - val_accuracy: 0.9470 - val_auc: 0.9876
Epoch 164/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6257 - accuracy: 0.9274 - auc: 0.9727 - val_loss: 0.5840 - val_accuracy: 0.9495 - val_auc: 0.9887
Epoch 165/1000
58/58 [=====] - 1s 19ms/step - loss: 0.6203 - accuracy: 0.9312 - auc: 0.9747 - val_loss: 0.6021 - val_accuracy: 0.9394 - val_auc: 0.9839
Epoch 166/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6221 - accuracy: 0.9291 - auc: 0.9731 - val_loss: 0.5828 - val_accuracy: 0.9495 - val_auc: 0.9866
Epoch 167/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6125 - accuracy: 0.9269 - auc: 0.9776 - val_loss: 0.5939 - val_accuracy: 0.9419 - val_auc: 0.9857
Epoch 168/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6213 - accuracy: 0.9291 - auc: 0.9728 - val_loss: 0.5894 - val_accuracy: 0.9444 - val_auc: 0.9863
Epoch 169/1000
58/58 [=====] - 1s 21ms/step - loss: 0.6232 - accuracy: 0.9269 - auc: 0.9720 - val_loss: 0.5794 - val_accuracy: 0.9495 - val_auc: 0.9888
Epoch 170/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6135 - accuracy: 0.9296 - auc: 0.9752 - val_loss: 0.5820 - val_accuracy: 0.9470 - val_auc: 0.9874
Epoch 171/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6082 - accuracy: 0.9296 - auc: 0.9773 - val_loss: 0.5799 - val_accuracy: 0.9470 - val_auc: 0.9877
Epoch 172/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6064 - accuracy: 0.9323 - auc: 0.9758 - val_loss: 0.5867 - val_accuracy: 0.9419 - val_auc: 0.9860
Epoch 173/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6121 - accuracy: 0.9323 - auc: 0.9743 - val_loss: 0.5683 - val_accuracy: 0.9545 - val_auc: 0.9883
Epoch 174/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5981 - accuracy: 0.9296 - auc: 0.9792 - val_loss: 0.5695 - val_accuracy: 0.9495 - val_auc: 0.9895
Epoch 175/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6084 - accuracy: 0.9339 - auc: 0.9749 - val_loss: 0.5718 - val_accuracy: 0.9470 - val_auc: 0.9883
Epoch 176/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6084 - accuracy: 0.9312 - auc: 0.9746 - val_loss: 0.5685 - val_accuracy: 0.9495 - val_auc: 0.9891
Epoch 177/1000
58/58 [=====] - 1s 19ms/step - loss: 0.5950 - accuracy: 0.9334 - auc: 0.9779 - val_loss: 0.5708 - val_accuracy: 0.9470 - val_auc: 0.9888
Epoch 178/1000
58/58 [=====] - 1s 20ms/step - loss: 0.6010 - accuracy: 0.9361 - auc: 0.9770 - val_loss: 0.5617 - val_accuracy: 0.9520 - val_auc: 0.9887
Epoch 179/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5969 - accuracy:

0.9356 - auc: 0.9754 - val_loss: 0.5574 - val_accuracy: 0.9545 - val_auc: 0.9912
Epoch 180/1000
58/58 [=====] - 1s 19ms/step - loss: 0.5981 - accuracy: 0.9345 - auc: 0.9769 - val_loss: 0.5629 - val_accuracy: 0.9520 - val_auc: 0.9900
Epoch 181/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5939 - accuracy: 0.9345 - auc: 0.9765 - val_loss: 0.5648 - val_accuracy: 0.9495 - val_auc: 0.9888
Epoch 182/1000
58/58 [=====] - 1s 19ms/step - loss: 0.5839 - accuracy: 0.9394 - auc: 0.9804 - val_loss: 0.5647 - val_accuracy: 0.9470 - val_auc: 0.9866
Epoch 183/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5869 - accuracy: 0.9350 - auc: 0.9778 - val_loss: 0.5567 - val_accuracy: 0.9520 - val_auc: 0.9887
Epoch 184/1000
58/58 [=====] - 1s 21ms/step - loss: 0.5917 - accuracy: 0.9318 - auc: 0.9771 - val_loss: 0.5615 - val_accuracy: 0.9470 - val_auc: 0.9888
Epoch 185/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5929 - accuracy: 0.9367 - auc: 0.9763 - val_loss: 0.5571 - val_accuracy: 0.9495 - val_auc: 0.9900
Epoch 186/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5853 - accuracy: 0.9334 - auc: 0.9778 - val_loss: 0.5579 - val_accuracy: 0.9495 - val_auc: 0.9895
Epoch 187/1000
58/58 [=====] - 1s 19ms/step - loss: 0.5932 - accuracy: 0.9339 - auc: 0.9763 - val_loss: 0.5629 - val_accuracy: 0.9470 - val_auc: 0.9883
Epoch 188/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5769 - accuracy: 0.9372 - auc: 0.9792 - val_loss: 0.5532 - val_accuracy: 0.9520 - val_auc: 0.9899
Epoch 189/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5862 - accuracy: 0.9334 - auc: 0.9772 - val_loss: 0.5481 - val_accuracy: 0.9545 - val_auc: 0.9913
Epoch 190/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5805 - accuracy: 0.9377 - auc: 0.9782 - val_loss: 0.5553 - val_accuracy: 0.9495 - val_auc: 0.9892
Epoch 191/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5745 - accuracy: 0.9394 - auc: 0.9791 - val_loss: 0.5555 - val_accuracy: 0.9470 - val_auc: 0.9889
Epoch 192/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5781 - accuracy: 0.9367 - auc: 0.9791 - val_loss: 0.5572 - val_accuracy: 0.9470 - val_auc: 0.9885
Epoch 193/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5813 - accuracy: 0.9399 - auc: 0.9773 - val_loss: 0.5535 - val_accuracy: 0.9495 - val_auc: 0.9890
Epoch 194/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5721 - accuracy: 0.9394 - auc: 0.9797 - val_loss: 0.5458 - val_accuracy: 0.9495 - val_auc: 0.9901
Epoch 195/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5768 - accuracy:

0.9334 - auc: 0.9798 - val_loss: 0.5416 - val_accuracy: 0.9545 - val_auc: 0.9911
Epoch 196/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5716 - accuracy: 0.9372 - auc: 0.9803 - val_loss: 0.5529 - val_accuracy: 0.9495 - val_auc: 0.9888
Epoch 197/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5773 - accuracy: 0.9372 - auc: 0.9780 - val_loss: 0.5482 - val_accuracy: 0.9495 - val_auc: 0.9893
Epoch 198/1000
58/58 [=====] - 1s 23ms/step - loss: 0.5709 - accuracy: 0.9404 - auc: 0.9789 - val_loss: 0.5464 - val_accuracy: 0.9495 - val_auc: 0.9898
Epoch 199/1000
58/58 [=====] - 1s 23ms/step - loss: 0.5732 - accuracy: 0.9356 - auc: 0.9777 - val_loss: 0.5386 - val_accuracy: 0.9495 - val_auc: 0.9912
Epoch 200/1000
58/58 [=====] - 1s 22ms/step - loss: 0.5645 - accuracy: 0.9410 - auc: 0.9798 - val_loss: 0.5345 - val_accuracy: 0.9545 - val_auc: 0.9920
Epoch 201/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5648 - accuracy: 0.9356 - auc: 0.9788 - val_loss: 0.5338 - val_accuracy: 0.9520 - val_auc: 0.9914
Epoch 202/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5662 - accuracy: 0.9404 - auc: 0.9791 - val_loss: 0.5373 - val_accuracy: 0.9470 - val_auc: 0.9888
Epoch 203/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5658 - accuracy: 0.9426 - auc: 0.9784 - val_loss: 0.5310 - val_accuracy: 0.9545 - val_auc: 0.9900
Epoch 204/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5763 - accuracy: 0.9339 - auc: 0.9764 - val_loss: 0.5321 - val_accuracy: 0.9520 - val_auc: 0.9917
Epoch 205/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5569 - accuracy: 0.9415 - auc: 0.9815 - val_loss: 0.5349 - val_accuracy: 0.9545 - val_auc: 0.9909
Epoch 206/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5631 - accuracy: 0.9399 - auc: 0.9805 - val_loss: 0.5261 - val_accuracy: 0.9545 - val_auc: 0.9904
Epoch 207/1000
58/58 [=====] - 1s 19ms/step - loss: 0.5554 - accuracy: 0.9421 - auc: 0.9816 - val_loss: 0.5236 - val_accuracy: 0.9545 - val_auc: 0.9910
Epoch 208/1000
58/58 [=====] - 1s 19ms/step - loss: 0.5479 - accuracy: 0.9469 - auc: 0.9838 - val_loss: 0.5316 - val_accuracy: 0.9545 - val_auc: 0.9904
Epoch 209/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5570 - accuracy: 0.9426 - auc: 0.9808 - val_loss: 0.5222 - val_accuracy: 0.9520 - val_auc: 0.9906
Epoch 210/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5528 - accuracy: 0.9383 - auc: 0.9810 - val_loss: 0.5271 - val_accuracy: 0.9545 - val_auc: 0.9916
Epoch 211/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5481 - accuracy:

0.9432 - auc: 0.9826 - val_loss: 0.5202 - val_accuracy: 0.9545 - val_auc: 0.9928
Epoch 212/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5555 - accuracy: 0.9399 - auc: 0.9807 - val_loss: 0.5295 - val_accuracy: 0.9545 - val_auc: 0.9885
Epoch 213/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5532 - accuracy: 0.9367 - auc: 0.9816 - val_loss: 0.5148 - val_accuracy: 0.9545 - val_auc: 0.9912
Epoch 214/1000
58/58 [=====] - 1s 19ms/step - loss: 0.5502 - accuracy: 0.9448 - auc: 0.9797 - val_loss: 0.5161 - val_accuracy: 0.9571 - val_auc: 0.9930
Epoch 215/1000
58/58 [=====] - 1s 21ms/step - loss: 0.5596 - accuracy: 0.9383 - auc: 0.9774 - val_loss: 0.5179 - val_accuracy: 0.9571 - val_auc: 0.9919
Epoch 216/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5486 - accuracy: 0.9437 - auc: 0.9803 - val_loss: 0.5169 - val_accuracy: 0.9596 - val_auc: 0.9918
Epoch 217/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5349 - accuracy: 0.9486 - auc: 0.9831 - val_loss: 0.5169 - val_accuracy: 0.9571 - val_auc: 0.9922
Epoch 218/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5523 - accuracy: 0.9453 - auc: 0.9785 - val_loss: 0.5264 - val_accuracy: 0.9545 - val_auc: 0.9877
Epoch 219/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5395 - accuracy: 0.9453 - auc: 0.9816 - val_loss: 0.5078 - val_accuracy: 0.9545 - val_auc: 0.9934
Epoch 220/1000
58/58 [=====] - 1s 21ms/step - loss: 0.5441 - accuracy: 0.9426 - auc: 0.9804 - val_loss: 0.5082 - val_accuracy: 0.9646 - val_auc: 0.9939
Epoch 221/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5399 - accuracy: 0.9453 - auc: 0.9826 - val_loss: 0.5142 - val_accuracy: 0.9621 - val_auc: 0.9918
Epoch 222/1000
58/58 [=====] - 1s 21ms/step - loss: 0.5447 - accuracy: 0.9377 - auc: 0.9811 - val_loss: 0.5053 - val_accuracy: 0.9571 - val_auc: 0.9936
Epoch 223/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5359 - accuracy: 0.9464 - auc: 0.9829 - val_loss: 0.5096 - val_accuracy: 0.9545 - val_auc: 0.9930
Epoch 224/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5385 - accuracy: 0.9432 - auc: 0.9799 - val_loss: 0.5080 - val_accuracy: 0.9621 - val_auc: 0.9926
Epoch 225/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5440 - accuracy: 0.9421 - auc: 0.9786 - val_loss: 0.5002 - val_accuracy: 0.9646 - val_auc: 0.9940
Epoch 226/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5314 - accuracy: 0.9453 - auc: 0.9830 - val_loss: 0.5070 - val_accuracy: 0.9596 - val_auc: 0.9932
Epoch 227/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5259 - accuracy:

0.9469 - auc: 0.9842 - val_loss: 0.4998 - val_accuracy: 0.9571 - val_auc: 0.9936
Epoch 228/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5366 - accuracy: 0.9388 - auc: 0.9818 - val_loss: 0.4995 - val_accuracy: 0.9646 - val_auc: 0.9933
Epoch 229/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5368 - accuracy: 0.9377 - auc: 0.9809 - val_loss: 0.5006 - val_accuracy: 0.9571 - val_auc: 0.9916
Epoch 230/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5274 - accuracy: 0.9469 - auc: 0.9820 - val_loss: 0.4968 - val_accuracy: 0.9596 - val_auc: 0.9940
Epoch 231/1000
58/58 [=====] - 1s 21ms/step - loss: 0.5274 - accuracy: 0.9453 - auc: 0.9834 - val_loss: 0.4906 - val_accuracy: 0.9697 - val_auc: 0.9948
Epoch 232/1000
58/58 [=====] - 1s 21ms/step - loss: 0.5258 - accuracy: 0.9453 - auc: 0.9830 - val_loss: 0.4929 - val_accuracy: 0.9646 - val_auc: 0.9944
Epoch 233/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5356 - accuracy: 0.9421 - auc: 0.9817 - val_loss: 0.4911 - val_accuracy: 0.9571 - val_auc: 0.9924
Epoch 234/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5284 - accuracy: 0.9410 - auc: 0.9823 - val_loss: 0.4927 - val_accuracy: 0.9672 - val_auc: 0.9945
Epoch 235/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5228 - accuracy: 0.9399 - auc: 0.9832 - val_loss: 0.4842 - val_accuracy: 0.9596 - val_auc: 0.9930
Epoch 236/1000
58/58 [=====] - 1s 21ms/step - loss: 0.5220 - accuracy: 0.9486 - auc: 0.9823 - val_loss: 0.5059 - val_accuracy: 0.9596 - val_auc: 0.9911
Epoch 237/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5147 - accuracy: 0.9469 - auc: 0.9851 - val_loss: 0.4921 - val_accuracy: 0.9621 - val_auc: 0.9938
Epoch 238/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5137 - accuracy: 0.9502 - auc: 0.9850 - val_loss: 0.4905 - val_accuracy: 0.9646 - val_auc: 0.9933
Epoch 239/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5217 - accuracy: 0.9464 - auc: 0.9823 - val_loss: 0.4819 - val_accuracy: 0.9621 - val_auc: 0.9929
Epoch 240/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5115 - accuracy: 0.9480 - auc: 0.9866 - val_loss: 0.4826 - val_accuracy: 0.9596 - val_auc: 0.9945
Epoch 241/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5215 - accuracy: 0.9448 - auc: 0.9822 - val_loss: 0.4774 - val_accuracy: 0.9697 - val_auc: 0.9935
Epoch 242/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5158 - accuracy: 0.9480 - auc: 0.9843 - val_loss: 0.4873 - val_accuracy: 0.9646 - val_auc: 0.9935
Epoch 243/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5130 - accuracy:

0.9507 - auc: 0.9843 - val_loss: 0.4876 - val_accuracy: 0.9646 - val_auc: 0.9935
Epoch 244/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5204 - accuracy: 0.9491 - auc: 0.9810 - val_loss: 0.4993 - val_accuracy: 0.9596 - val_auc: 0.9911
Epoch 245/1000
58/58 [=====] - 1s 21ms/step - loss: 0.5108 - accuracy: 0.9475 - auc: 0.9841 - val_loss: 0.4846 - val_accuracy: 0.9646 - val_auc: 0.9934
Epoch 246/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5065 - accuracy: 0.9486 - auc: 0.9859 - val_loss: 0.4758 - val_accuracy: 0.9672 - val_auc: 0.9949
Epoch 247/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5070 - accuracy: 0.9491 - auc: 0.9843 - val_loss: 0.4822 - val_accuracy: 0.9646 - val_auc: 0.9938
Epoch 248/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5052 - accuracy: 0.9529 - auc: 0.9845 - val_loss: 0.4871 - val_accuracy: 0.9621 - val_auc: 0.9927
Epoch 249/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5038 - accuracy: 0.9502 - auc: 0.9851 - val_loss: 0.4718 - val_accuracy: 0.9697 - val_auc: 0.9951
Epoch 250/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5100 - accuracy: 0.9491 - auc: 0.9839 - val_loss: 0.4757 - val_accuracy: 0.9646 - val_auc: 0.9944
Epoch 251/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5060 - accuracy: 0.9448 - auc: 0.9839 - val_loss: 0.4735 - val_accuracy: 0.9646 - val_auc: 0.9946
Epoch 252/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5110 - accuracy: 0.9426 - auc: 0.9833 - val_loss: 0.4738 - val_accuracy: 0.9646 - val_auc: 0.9944
Epoch 253/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5057 - accuracy: 0.9459 - auc: 0.9852 - val_loss: 0.4757 - val_accuracy: 0.9646 - val_auc: 0.9936
Epoch 254/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5054 - accuracy: 0.9480 - auc: 0.9840 - val_loss: 0.4658 - val_accuracy: 0.9672 - val_auc: 0.9952
Epoch 255/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4995 - accuracy: 0.9491 - auc: 0.9856 - val_loss: 0.4644 - val_accuracy: 0.9697 - val_auc: 0.9954
Epoch 256/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5031 - accuracy: 0.9464 - auc: 0.9846 - val_loss: 0.4619 - val_accuracy: 0.9722 - val_auc: 0.9960
Epoch 257/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4974 - accuracy: 0.9453 - auc: 0.9857 - val_loss: 0.4646 - val_accuracy: 0.9672 - val_auc: 0.9949
Epoch 258/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4924 - accuracy: 0.9464 - auc: 0.9868 - val_loss: 0.4626 - val_accuracy: 0.9697 - val_auc: 0.9954
Epoch 259/1000
58/58 [=====] - 1s 20ms/step - loss: 0.5030 - accuracy:

0.9486 - auc: 0.9820 - val_loss: 0.4589 - val_accuracy: 0.9722 - val_auc: 0.9960
Epoch 260/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4956 - accuracy: 0.9448 - auc: 0.9854 - val_loss: 0.4613 - val_accuracy: 0.9697 - val_auc: 0.9955
Epoch 261/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4997 - accuracy: 0.9496 - auc: 0.9831 - val_loss: 0.4620 - val_accuracy: 0.9672 - val_auc: 0.9951
Epoch 262/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4849 - accuracy: 0.9513 - auc: 0.9865 - val_loss: 0.4549 - val_accuracy: 0.9697 - val_auc: 0.9942
Epoch 263/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4966 - accuracy: 0.9507 - auc: 0.9838 - val_loss: 0.4532 - val_accuracy: 0.9773 - val_auc: 0.9962
Epoch 264/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4863 - accuracy: 0.9540 - auc: 0.9867 - val_loss: 0.4516 - val_accuracy: 0.9773 - val_auc: 0.9962
Epoch 265/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4846 - accuracy: 0.9529 - auc: 0.9867 - val_loss: 0.4633 - val_accuracy: 0.9646 - val_auc: 0.9942
Epoch 266/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4838 - accuracy: 0.9518 - auc: 0.9866 - val_loss: 0.4658 - val_accuracy: 0.9646 - val_auc: 0.9936
Epoch 267/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4807 - accuracy: 0.9529 - auc: 0.9873 - val_loss: 0.4532 - val_accuracy: 0.9697 - val_auc: 0.9956
Epoch 268/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4924 - accuracy: 0.9502 - auc: 0.9837 - val_loss: 0.4484 - val_accuracy: 0.9747 - val_auc: 0.9963
Epoch 269/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4787 - accuracy: 0.9534 - auc: 0.9869 - val_loss: 0.4592 - val_accuracy: 0.9646 - val_auc: 0.9946
Epoch 270/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4787 - accuracy: 0.9545 - auc: 0.9876 - val_loss: 0.4433 - val_accuracy: 0.9798 - val_auc: 0.9945
Epoch 271/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4776 - accuracy: 0.9556 - auc: 0.9874 - val_loss: 0.4476 - val_accuracy: 0.9697 - val_auc: 0.9960
Epoch 272/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4815 - accuracy: 0.9524 - auc: 0.9868 - val_loss: 0.4473 - val_accuracy: 0.9722 - val_auc: 0.9961
Epoch 273/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4742 - accuracy: 0.9534 - auc: 0.9882 - val_loss: 0.4490 - val_accuracy: 0.9697 - val_auc: 0.9958
Epoch 274/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4810 - accuracy: 0.9486 - auc: 0.9858 - val_loss: 0.4536 - val_accuracy: 0.9672 - val_auc: 0.9949
Epoch 275/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4797 - accuracy:

0.9518 - auc: 0.9852 - val_loss: 0.4444 - val_accuracy: 0.9697 - val_auc: 0.9960
Epoch 276/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4718 - accuracy: 0.9551 - auc: 0.9880 - val_loss: 0.4436 - val_accuracy: 0.9697 - val_auc: 0.9961
Epoch 277/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4739 - accuracy: 0.9561 - auc: 0.9872 - val_loss: 0.4454 - val_accuracy: 0.9697 - val_auc: 0.9958
Epoch 278/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4717 - accuracy: 0.9524 - auc: 0.9880 - val_loss: 0.4392 - val_accuracy: 0.9773 - val_auc: 0.9965
Epoch 279/1000
58/58 [=====] - 1s 22ms/step - loss: 0.4725 - accuracy: 0.9578 - auc: 0.9853 - val_loss: 0.4359 - val_accuracy: 0.9773 - val_auc: 0.9966
Epoch 280/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4661 - accuracy: 0.9551 - auc: 0.9879 - val_loss: 0.4373 - val_accuracy: 0.9697 - val_auc: 0.9963
Epoch 281/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4688 - accuracy: 0.9529 - auc: 0.9876 - val_loss: 0.4482 - val_accuracy: 0.9646 - val_auc: 0.9946
Epoch 282/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4774 - accuracy: 0.9496 - auc: 0.9845 - val_loss: 0.4508 - val_accuracy: 0.9621 - val_auc: 0.9942
Epoch 283/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4739 - accuracy: 0.9556 - auc: 0.9838 - val_loss: 0.4341 - val_accuracy: 0.9773 - val_auc: 0.9967
Epoch 284/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4696 - accuracy: 0.9513 - auc: 0.9864 - val_loss: 0.4304 - val_accuracy: 0.9773 - val_auc: 0.9947
Epoch 285/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4629 - accuracy: 0.9567 - auc: 0.9883 - val_loss: 0.4292 - val_accuracy: 0.9773 - val_auc: 0.9969
Epoch 286/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4717 - accuracy: 0.9529 - auc: 0.9840 - val_loss: 0.4386 - val_accuracy: 0.9697 - val_auc: 0.9959
Epoch 287/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4584 - accuracy: 0.9561 - auc: 0.9885 - val_loss: 0.4313 - val_accuracy: 0.9747 - val_auc: 0.9964
Epoch 288/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4680 - accuracy: 0.9551 - auc: 0.9873 - val_loss: 0.4299 - val_accuracy: 0.9697 - val_auc: 0.9965
Epoch 289/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4694 - accuracy: 0.9513 - auc: 0.9853 - val_loss: 0.4278 - val_accuracy: 0.9773 - val_auc: 0.9968
Epoch 290/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4620 - accuracy: 0.9561 - auc: 0.9877 - val_loss: 0.4308 - val_accuracy: 0.9672 - val_auc: 0.9961
Epoch 291/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4627 - accuracy:

0.9534 - auc: 0.9866 - val_loss: 0.4392 - val_accuracy: 0.9646 - val_auc: 0.9952
Epoch 292/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4566 - accuracy: 0.9556 - auc: 0.9873 - val_loss: 0.4280 - val_accuracy: 0.9747 - val_auc: 0.9965
Epoch 293/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4670 - accuracy: 0.9534 - auc: 0.9877 - val_loss: 0.4217 - val_accuracy: 0.9773 - val_auc: 0.9969
Epoch 294/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4545 - accuracy: 0.9529 - auc: 0.9886 - val_loss: 0.4250 - val_accuracy: 0.9722 - val_auc: 0.9967
Epoch 295/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4512 - accuracy: 0.9599 - auc: 0.9884 - val_loss: 0.4227 - val_accuracy: 0.9798 - val_auc: 0.9968
Epoch 296/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4457 - accuracy: 0.9578 - auc: 0.9906 - val_loss: 0.4259 - val_accuracy: 0.9697 - val_auc: 0.9965
Epoch 297/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4567 - accuracy: 0.9556 - auc: 0.9880 - val_loss: 0.4200 - val_accuracy: 0.9747 - val_auc: 0.9968
Epoch 298/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4491 - accuracy: 0.9545 - auc: 0.9895 - val_loss: 0.4211 - val_accuracy: 0.9747 - val_auc: 0.9966
Epoch 299/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4529 - accuracy: 0.9572 - auc: 0.9891 - val_loss: 0.4186 - val_accuracy: 0.9747 - val_auc: 0.9969
Epoch 300/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4539 - accuracy: 0.9551 - auc: 0.9860 - val_loss: 0.4207 - val_accuracy: 0.9798 - val_auc: 0.9968
Epoch 301/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4487 - accuracy: 0.9567 - auc: 0.9886 - val_loss: 0.4217 - val_accuracy: 0.9697 - val_auc: 0.9965
Epoch 302/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4501 - accuracy: 0.9551 - auc: 0.9885 - val_loss: 0.4154 - val_accuracy: 0.9798 - val_auc: 0.9970
Epoch 303/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4438 - accuracy: 0.9572 - auc: 0.9896 - val_loss: 0.4186 - val_accuracy: 0.9697 - val_auc: 0.9964
Epoch 304/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4420 - accuracy: 0.9594 - auc: 0.9897 - val_loss: 0.4170 - val_accuracy: 0.9722 - val_auc: 0.9967
Epoch 305/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4375 - accuracy: 0.9610 - auc: 0.9903 - val_loss: 0.4131 - val_accuracy: 0.9798 - val_auc: 0.9971
Epoch 306/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4478 - accuracy: 0.9589 - auc: 0.9878 - val_loss: 0.4083 - val_accuracy: 0.9773 - val_auc: 0.9974
Epoch 307/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4442 - accuracy:

0.9534 - auc: 0.9887 - val_loss: 0.4110 - val_accuracy: 0.9773 - val_auc: 0.9970
Epoch 308/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4531 - accuracy: 0.9534 - auc: 0.9859 - val_loss: 0.4128 - val_accuracy: 0.9798 - val_auc: 0.9970
Epoch 309/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4442 - accuracy: 0.9589 - auc: 0.9883 - val_loss: 0.4083 - val_accuracy: 0.9798 - val_auc: 0.9973
Epoch 310/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4402 - accuracy: 0.9561 - auc: 0.9890 - val_loss: 0.4106 - val_accuracy: 0.9773 - val_auc: 0.9970
Epoch 311/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4370 - accuracy: 0.9621 - auc: 0.9890 - val_loss: 0.4151 - val_accuracy: 0.9697 - val_auc: 0.9967
Epoch 312/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4404 - accuracy: 0.9561 - auc: 0.9893 - val_loss: 0.4141 - val_accuracy: 0.9697 - val_auc: 0.9964
Epoch 313/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4490 - accuracy: 0.9572 - auc: 0.9859 - val_loss: 0.4016 - val_accuracy: 0.9773 - val_auc: 0.9975
Epoch 314/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4373 - accuracy: 0.9540 - auc: 0.9901 - val_loss: 0.4053 - val_accuracy: 0.9798 - val_auc: 0.9973
Epoch 315/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4341 - accuracy: 0.9621 - auc: 0.9888 - val_loss: 0.4063 - val_accuracy: 0.9798 - val_auc: 0.9970
Epoch 316/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4591 - accuracy: 0.9448 - auc: 0.9839 - val_loss: 0.4118 - val_accuracy: 0.9697 - val_auc: 0.9967
Epoch 317/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4429 - accuracy: 0.9599 - auc: 0.9870 - val_loss: 0.4018 - val_accuracy: 0.9798 - val_auc: 0.9973
Epoch 318/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4312 - accuracy: 0.9583 - auc: 0.9896 - val_loss: 0.4101 - val_accuracy: 0.9672 - val_auc: 0.9965
Epoch 319/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4367 - accuracy: 0.9572 - auc: 0.9892 - val_loss: 0.4133 - val_accuracy: 0.9621 - val_auc: 0.9964
Epoch 320/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4362 - accuracy: 0.9534 - auc: 0.9884 - val_loss: 0.4060 - val_accuracy: 0.9697 - val_auc: 0.9970
Epoch 321/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4306 - accuracy: 0.9589 - auc: 0.9903 - val_loss: 0.4021 - val_accuracy: 0.9747 - val_auc: 0.9971
Epoch 322/1000
58/58 [=====] - 1s 22ms/step - loss: 0.4337 - accuracy: 0.9626 - auc: 0.9878 - val_loss: 0.4053 - val_accuracy: 0.9672 - val_auc: 0.9967
Epoch 323/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4298 - accuracy:

0.9551 - auc: 0.9892 - val_loss: 0.3942 - val_accuracy: 0.9798 - val_auc: 0.9977
Epoch 324/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4285 - accuracy: 0.9572 - auc: 0.9893 - val_loss: 0.3988 - val_accuracy: 0.9747 - val_auc: 0.9971
Epoch 325/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4308 - accuracy: 0.9567 - auc: 0.9891 - val_loss: 0.3948 - val_accuracy: 0.9798 - val_auc: 0.9975
Epoch 326/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4330 - accuracy: 0.9556 - auc: 0.9877 - val_loss: 0.3944 - val_accuracy: 0.9798 - val_auc: 0.9977
Epoch 327/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4206 - accuracy: 0.9648 - auc: 0.9914 - val_loss: 0.3947 - val_accuracy: 0.9773 - val_auc: 0.9974
Epoch 328/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4267 - accuracy: 0.9572 - auc: 0.9894 - val_loss: 0.3939 - val_accuracy: 0.9773 - val_auc: 0.9976
Epoch 329/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4317 - accuracy: 0.9621 - auc: 0.9877 - val_loss: 0.4001 - val_accuracy: 0.9646 - val_auc: 0.9965
Epoch 330/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4224 - accuracy: 0.9583 - auc: 0.9907 - val_loss: 0.3905 - val_accuracy: 0.9798 - val_auc: 0.9975
Epoch 331/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4267 - accuracy: 0.9589 - auc: 0.9883 - val_loss: 0.3934 - val_accuracy: 0.9773 - val_auc: 0.9976
Epoch 332/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4223 - accuracy: 0.9599 - auc: 0.9902 - val_loss: 0.3866 - val_accuracy: 0.9798 - val_auc: 0.9976
Epoch 333/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4167 - accuracy: 0.9610 - auc: 0.9898 - val_loss: 0.3881 - val_accuracy: 0.9798 - val_auc: 0.9976
Epoch 334/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4218 - accuracy: 0.9605 - auc: 0.9899 - val_loss: 0.3932 - val_accuracy: 0.9722 - val_auc: 0.9973
Epoch 335/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4198 - accuracy: 0.9578 - auc: 0.9881 - val_loss: 0.3889 - val_accuracy: 0.9798 - val_auc: 0.9978
Epoch 336/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4166 - accuracy: 0.9594 - auc: 0.9900 - val_loss: 0.3892 - val_accuracy: 0.9773 - val_auc: 0.9976
Epoch 337/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4101 - accuracy: 0.9632 - auc: 0.9911 - val_loss: 0.3896 - val_accuracy: 0.9747 - val_auc: 0.9975
Epoch 338/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4177 - accuracy: 0.9616 - auc: 0.9891 - val_loss: 0.3847 - val_accuracy: 0.9773 - val_auc: 0.9976
Epoch 339/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4147 - accuracy:

0.9637 - auc: 0.9901 - val_loss: 0.3884 - val_accuracy: 0.9773 - val_auc: 0.9976
Epoch 340/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4189 - accuracy: 0.9578 - auc: 0.9889 - val_loss: 0.3833 - val_accuracy: 0.9798 - val_auc: 0.9977
Epoch 341/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4184 - accuracy: 0.9578 - auc: 0.9885 - val_loss: 0.3808 - val_accuracy: 0.9798 - val_auc: 0.9978
Epoch 342/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4203 - accuracy: 0.9551 - auc: 0.9876 - val_loss: 0.3835 - val_accuracy: 0.9798 - val_auc: 0.9978
Epoch 343/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4239 - accuracy: 0.9556 - auc: 0.9864 - val_loss: 0.3796 - val_accuracy: 0.9798 - val_auc: 0.9979
Epoch 344/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4245 - accuracy: 0.9545 - auc: 0.9878 - val_loss: 0.3806 - val_accuracy: 0.9798 - val_auc: 0.9980
Epoch 345/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4179 - accuracy: 0.9567 - auc: 0.9868 - val_loss: 0.3844 - val_accuracy: 0.9773 - val_auc: 0.9978
Epoch 346/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4170 - accuracy: 0.9599 - auc: 0.9883 - val_loss: 0.3869 - val_accuracy: 0.9722 - val_auc: 0.9971
Epoch 347/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4145 - accuracy: 0.9583 - auc: 0.9891 - val_loss: 0.3782 - val_accuracy: 0.9773 - val_auc: 0.9980
Epoch 348/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4141 - accuracy: 0.9599 - auc: 0.9892 - val_loss: 0.3772 - val_accuracy: 0.9798 - val_auc: 0.9977
Epoch 349/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4081 - accuracy: 0.9599 - auc: 0.9909 - val_loss: 0.3743 - val_accuracy: 0.9798 - val_auc: 0.9982
Epoch 350/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4127 - accuracy: 0.9583 - auc: 0.9894 - val_loss: 0.3782 - val_accuracy: 0.9798 - val_auc: 0.9978
Epoch 351/1000
58/58 [=====] - 1s 21ms/step - loss: 0.4104 - accuracy: 0.9589 - auc: 0.9894 - val_loss: 0.3784 - val_accuracy: 0.9773 - val_auc: 0.9978
Epoch 352/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4111 - accuracy: 0.9643 - auc: 0.9878 - val_loss: 0.3775 - val_accuracy: 0.9798 - val_auc: 0.9978
Epoch 353/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4134 - accuracy: 0.9578 - auc: 0.9880 - val_loss: 0.3784 - val_accuracy: 0.9773 - val_auc: 0.9978
Epoch 354/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4033 - accuracy: 0.9626 - auc: 0.9896 - val_loss: 0.3717 - val_accuracy: 0.9798 - val_auc: 0.9979
Epoch 355/1000
58/58 [=====] - 1s 22ms/step - loss: 0.4059 - accuracy:

0.9626 - auc: 0.9904 - val_loss: 0.3723 - val_accuracy: 0.9798 - val_auc: 0.9982
Epoch 356/1000
58/58 [=====] - 1s 22ms/step - loss: 0.4030 - accuracy: 0.9653 - auc: 0.9898 - val_loss: 0.3820 - val_accuracy: 0.9621 - val_auc: 0.9970
Epoch 357/1000
58/58 [=====] - 1s 22ms/step - loss: 0.3978 - accuracy: 0.9659 - auc: 0.9907 - val_loss: 0.3728 - val_accuracy: 0.9798 - val_auc: 0.9980
Epoch 358/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3906 - accuracy: 0.9643 - auc: 0.9926 - val_loss: 0.3698 - val_accuracy: 0.9798 - val_auc: 0.9981
Epoch 359/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3961 - accuracy: 0.9632 - auc: 0.9918 - val_loss: 0.3796 - val_accuracy: 0.9697 - val_auc: 0.9971
Epoch 360/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3975 - accuracy: 0.9648 - auc: 0.9910 - val_loss: 0.3688 - val_accuracy: 0.9823 - val_auc: 0.9982
Epoch 361/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3882 - accuracy: 0.9632 - auc: 0.9939 - val_loss: 0.3656 - val_accuracy: 0.9848 - val_auc: 0.9983
Epoch 362/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3966 - accuracy: 0.9610 - auc: 0.9913 - val_loss: 0.3715 - val_accuracy: 0.9773 - val_auc: 0.9977
Epoch 363/1000
58/58 [=====] - 1s 20ms/step - loss: 0.4029 - accuracy: 0.9578 - auc: 0.9887 - val_loss: 0.3643 - val_accuracy: 0.9848 - val_auc: 0.9983
Epoch 364/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3921 - accuracy: 0.9626 - auc: 0.9916 - val_loss: 0.3644 - val_accuracy: 0.9823 - val_auc: 0.9983
Epoch 365/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3907 - accuracy: 0.9637 - auc: 0.9932 - val_loss: 0.3703 - val_accuracy: 0.9747 - val_auc: 0.9977
Epoch 366/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3980 - accuracy: 0.9643 - auc: 0.9906 - val_loss: 0.3608 - val_accuracy: 0.9798 - val_auc: 0.9982
Epoch 367/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3874 - accuracy: 0.9664 - auc: 0.9915 - val_loss: 0.3673 - val_accuracy: 0.9773 - val_auc: 0.9978
Epoch 368/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3909 - accuracy: 0.9648 - auc: 0.9910 - val_loss: 0.3640 - val_accuracy: 0.9798 - val_auc: 0.9981
Epoch 369/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3941 - accuracy: 0.9583 - auc: 0.9903 - val_loss: 0.3617 - val_accuracy: 0.9798 - val_auc: 0.9982
Epoch 370/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3846 - accuracy: 0.9659 - auc: 0.9914 - val_loss: 0.3621 - val_accuracy: 0.9823 - val_auc: 0.9981
Epoch 371/1000
58/58 [=====] - 1s 22ms/step - loss: 0.3812 - accuracy:

0.9659 - auc: 0.9931 - val_loss: 0.3613 - val_accuracy: 0.9823 - val_auc: 0.9981
Epoch 372/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3901 - accuracy: 0.9664 - auc: 0.9906 - val_loss: 0.3588 - val_accuracy: 0.9848 - val_auc: 0.9983
Epoch 373/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3936 - accuracy: 0.9664 - auc: 0.9893 - val_loss: 0.3713 - val_accuracy: 0.9646 - val_auc: 0.9970
Epoch 374/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3882 - accuracy: 0.9605 - auc: 0.9915 - val_loss: 0.3556 - val_accuracy: 0.9798 - val_auc: 0.9984
Epoch 375/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3848 - accuracy: 0.9686 - auc: 0.9910 - val_loss: 0.3554 - val_accuracy: 0.9823 - val_auc: 0.9986
Epoch 376/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3848 - accuracy: 0.9621 - auc: 0.9917 - val_loss: 0.3537 - val_accuracy: 0.9798 - val_auc: 0.9984
Epoch 377/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3874 - accuracy: 0.9659 - auc: 0.9913 - val_loss: 0.3516 - val_accuracy: 0.9798 - val_auc: 0.9984
Epoch 378/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3865 - accuracy: 0.9648 - auc: 0.9901 - val_loss: 0.3571 - val_accuracy: 0.9823 - val_auc: 0.9982
Epoch 379/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3855 - accuracy: 0.9594 - auc: 0.9911 - val_loss: 0.3564 - val_accuracy: 0.9848 - val_auc: 0.9984
Epoch 380/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3849 - accuracy: 0.9632 - auc: 0.9911 - val_loss: 0.3597 - val_accuracy: 0.9747 - val_auc: 0.9978
Epoch 381/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3792 - accuracy: 0.9686 - auc: 0.9921 - val_loss: 0.3499 - val_accuracy: 0.9823 - val_auc: 0.9985
Epoch 382/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3851 - accuracy: 0.9637 - auc: 0.9905 - val_loss: 0.3586 - val_accuracy: 0.9773 - val_auc: 0.9980
Epoch 383/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3804 - accuracy: 0.9610 - auc: 0.9921 - val_loss: 0.3550 - val_accuracy: 0.9798 - val_auc: 0.9983
Epoch 384/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3869 - accuracy: 0.9643 - auc: 0.9903 - val_loss: 0.3502 - val_accuracy: 0.9798 - val_auc: 0.9984
Epoch 385/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3837 - accuracy: 0.9632 - auc: 0.9912 - val_loss: 0.3520 - val_accuracy: 0.9798 - val_auc: 0.9982
Epoch 386/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3822 - accuracy: 0.9632 - auc: 0.9910 - val_loss: 0.3493 - val_accuracy: 0.9848 - val_auc: 0.9986
Epoch 387/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3832 - accuracy:

0.9632 - auc: 0.9910 - val_loss: 0.3454 - val_accuracy: 0.9823 - val_auc: 0.9985
Epoch 388/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3748 - accuracy: 0.9675 - auc: 0.9922 - val_loss: 0.3509 - val_accuracy: 0.9823 - val_auc: 0.9983
Epoch 389/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3841 - accuracy: 0.9626 - auc: 0.9913 - val_loss: 0.3530 - val_accuracy: 0.9823 - val_auc: 0.9981
Epoch 390/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3699 - accuracy: 0.9691 - auc: 0.9930 - val_loss: 0.3433 - val_accuracy: 0.9823 - val_auc: 0.9986
Epoch 391/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3721 - accuracy: 0.9675 - auc: 0.9917 - val_loss: 0.3455 - val_accuracy: 0.9848 - val_auc: 0.9984
Epoch 392/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3748 - accuracy: 0.9659 - auc: 0.9918 - val_loss: 0.3519 - val_accuracy: 0.9773 - val_auc: 0.9979
Epoch 393/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3809 - accuracy: 0.9621 - auc: 0.9896 - val_loss: 0.3435 - val_accuracy: 0.9848 - val_auc: 0.9986
Epoch 394/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3778 - accuracy: 0.9616 - auc: 0.9917 - val_loss: 0.3427 - val_accuracy: 0.9823 - val_auc: 0.9985
Epoch 395/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3730 - accuracy: 0.9653 - auc: 0.9912 - val_loss: 0.3468 - val_accuracy: 0.9848 - val_auc: 0.9984
Epoch 396/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3800 - accuracy: 0.9637 - auc: 0.9892 - val_loss: 0.3410 - val_accuracy: 0.9823 - val_auc: 0.9984
Epoch 397/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3741 - accuracy: 0.9675 - auc: 0.9910 - val_loss: 0.3428 - val_accuracy: 0.9874 - val_auc: 0.9985
Epoch 398/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3708 - accuracy: 0.9670 - auc: 0.9914 - val_loss: 0.3410 - val_accuracy: 0.9848 - val_auc: 0.9987
Epoch 399/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3723 - accuracy: 0.9610 - auc: 0.9913 - val_loss: 0.3414 - val_accuracy: 0.9848 - val_auc: 0.9986
Epoch 400/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3751 - accuracy: 0.9691 - auc: 0.9897 - val_loss: 0.3430 - val_accuracy: 0.9874 - val_auc: 0.9986
Epoch 401/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3677 - accuracy: 0.9643 - auc: 0.9916 - val_loss: 0.3403 - val_accuracy: 0.9848 - val_auc: 0.9985
Epoch 402/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3723 - accuracy: 0.9659 - auc: 0.9897 - val_loss: 0.3419 - val_accuracy: 0.9848 - val_auc: 0.9985
Epoch 403/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3704 - accuracy:

0.9626 - auc: 0.9918 - val_loss: 0.3377 - val_accuracy: 0.9848 - val_auc: 0.9987
Epoch 404/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3750 - accuracy: 0.9648 - auc: 0.9898 - val_loss: 0.3363 - val_accuracy: 0.9823 - val_auc: 0.9986
Epoch 405/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3683 - accuracy: 0.9653 - auc: 0.9910 - val_loss: 0.3340 - val_accuracy: 0.9874 - val_auc: 0.9988
Epoch 406/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3755 - accuracy: 0.9605 - auc: 0.9885 - val_loss: 0.3363 - val_accuracy: 0.9823 - val_auc: 0.9987
Epoch 407/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3679 - accuracy: 0.9653 - auc: 0.9916 - val_loss: 0.3353 - val_accuracy: 0.9874 - val_auc: 0.9988
Epoch 408/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3773 - accuracy: 0.9589 - auc: 0.9898 - val_loss: 0.3339 - val_accuracy: 0.9798 - val_auc: 0.9987
Epoch 409/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3652 - accuracy: 0.9632 - auc: 0.9928 - val_loss: 0.3404 - val_accuracy: 0.9798 - val_auc: 0.9982
Epoch 410/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3754 - accuracy: 0.9626 - auc: 0.9888 - val_loss: 0.3331 - val_accuracy: 0.9848 - val_auc: 0.9988
Epoch 411/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3632 - accuracy: 0.9659 - auc: 0.9917 - val_loss: 0.3333 - val_accuracy: 0.9874 - val_auc: 0.9987
Epoch 412/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3586 - accuracy: 0.9675 - auc: 0.9932 - val_loss: 0.3292 - val_accuracy: 0.9874 - val_auc: 0.9988
Epoch 413/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3567 - accuracy: 0.9691 - auc: 0.9935 - val_loss: 0.3299 - val_accuracy: 0.9823 - val_auc: 0.9986
Epoch 414/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3562 - accuracy: 0.9691 - auc: 0.9927 - val_loss: 0.3295 - val_accuracy: 0.9798 - val_auc: 0.9986
Epoch 415/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3565 - accuracy: 0.9643 - auc: 0.9933 - val_loss: 0.3324 - val_accuracy: 0.9823 - val_auc: 0.9985
Epoch 416/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3580 - accuracy: 0.9670 - auc: 0.9918 - val_loss: 0.3333 - val_accuracy: 0.9848 - val_auc: 0.9984
Epoch 417/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3663 - accuracy: 0.9648 - auc: 0.9904 - val_loss: 0.3274 - val_accuracy: 0.9874 - val_auc: 0.9989
Epoch 418/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3507 - accuracy: 0.9729 - auc: 0.9935 - val_loss: 0.3272 - val_accuracy: 0.9848 - val_auc: 0.9988
Epoch 419/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3627 - accuracy:

0.9648 - auc: 0.9912 - val_loss: 0.3313 - val_accuracy: 0.9848 - val_auc: 0.9987
Epoch 420/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3538 - accuracy: 0.9675 - auc: 0.9927 - val_loss: 0.3259 - val_accuracy: 0.9874 - val_auc: 0.9989
Epoch 421/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3498 - accuracy: 0.9708 - auc: 0.9934 - val_loss: 0.3267 - val_accuracy: 0.9848 - val_auc: 0.9989
Epoch 422/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3593 - accuracy: 0.9653 - auc: 0.9906 - val_loss: 0.3247 - val_accuracy: 0.9874 - val_auc: 0.9988
Epoch 423/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3558 - accuracy: 0.9659 - auc: 0.9918 - val_loss: 0.3285 - val_accuracy: 0.9848 - val_auc: 0.9985
Epoch 424/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3646 - accuracy: 0.9616 - auc: 0.9901 - val_loss: 0.3295 - val_accuracy: 0.9848 - val_auc: 0.9985
Epoch 425/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3635 - accuracy: 0.9653 - auc: 0.9901 - val_loss: 0.3234 - val_accuracy: 0.9874 - val_auc: 0.9988
Epoch 426/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3694 - accuracy: 0.9583 - auc: 0.9890 - val_loss: 0.3196 - val_accuracy: 0.9823 - val_auc: 0.9988
Epoch 427/1000
58/58 [=====] - 1s 22ms/step - loss: 0.3438 - accuracy: 0.9724 - auc: 0.9931 - val_loss: 0.3256 - val_accuracy: 0.9848 - val_auc: 0.9987
Epoch 428/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3471 - accuracy: 0.9702 - auc: 0.9927 - val_loss: 0.3223 - val_accuracy: 0.9874 - val_auc: 0.9988
Epoch 429/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3536 - accuracy: 0.9681 - auc: 0.9913 - val_loss: 0.3236 - val_accuracy: 0.9874 - val_auc: 0.9987
Epoch 430/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3540 - accuracy: 0.9637 - auc: 0.9927 - val_loss: 0.3226 - val_accuracy: 0.9848 - val_auc: 0.9989
Epoch 431/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3512 - accuracy: 0.9659 - auc: 0.9917 - val_loss: 0.3207 - val_accuracy: 0.9848 - val_auc: 0.9989
Epoch 432/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3406 - accuracy: 0.9708 - auc: 0.9951 - val_loss: 0.3204 - val_accuracy: 0.9848 - val_auc: 0.9989
Epoch 433/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3476 - accuracy: 0.9724 - auc: 0.9907 - val_loss: 0.3202 - val_accuracy: 0.9848 - val_auc: 0.9989
Epoch 434/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3519 - accuracy: 0.9621 - auc: 0.9922 - val_loss: 0.3217 - val_accuracy: 0.9874 - val_auc: 0.9988
Epoch 435/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3390 - accuracy:

0.9735 - auc: 0.9942 - val_loss: 0.3232 - val_accuracy: 0.9848 - val_auc: 0.9986
Epoch 436/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3517 - accuracy: 0.9637 - auc: 0.9912 - val_loss: 0.3181 - val_accuracy: 0.9874 - val_auc: 0.9988
Epoch 437/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3602 - accuracy: 0.9659 - auc: 0.9892 - val_loss: 0.3159 - val_accuracy: 0.9874 - val_auc: 0.9989
Epoch 438/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3456 - accuracy: 0.9670 - auc: 0.9940 - val_loss: 0.3202 - val_accuracy: 0.9848 - val_auc: 0.9989
Epoch 439/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3468 - accuracy: 0.9691 - auc: 0.9914 - val_loss: 0.3152 - val_accuracy: 0.9874 - val_auc: 0.9989
Epoch 440/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3423 - accuracy: 0.9664 - auc: 0.9930 - val_loss: 0.3147 - val_accuracy: 0.9874 - val_auc: 0.9989
Epoch 441/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3520 - accuracy: 0.9664 - auc: 0.9915 - val_loss: 0.3110 - val_accuracy: 0.9848 - val_auc: 0.9991
Epoch 442/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3332 - accuracy: 0.9735 - auc: 0.9943 - val_loss: 0.3142 - val_accuracy: 0.9874 - val_auc: 0.9989
Epoch 443/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3369 - accuracy: 0.9735 - auc: 0.9934 - val_loss: 0.3115 - val_accuracy: 0.9823 - val_auc: 0.9990
Epoch 444/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3478 - accuracy: 0.9681 - auc: 0.9911 - val_loss: 0.3098 - val_accuracy: 0.9848 - val_auc: 0.9990
Epoch 445/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3375 - accuracy: 0.9697 - auc: 0.9940 - val_loss: 0.3167 - val_accuracy: 0.9823 - val_auc: 0.9988
Epoch 446/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3456 - accuracy: 0.9643 - auc: 0.9926 - val_loss: 0.3208 - val_accuracy: 0.9823 - val_auc: 0.9985
Epoch 447/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3479 - accuracy: 0.9659 - auc: 0.9918 - val_loss: 0.3103 - val_accuracy: 0.9874 - val_auc: 0.9990
Epoch 448/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3395 - accuracy: 0.9691 - auc: 0.9924 - val_loss: 0.3082 - val_accuracy: 0.9874 - val_auc: 0.9990
Epoch 449/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3333 - accuracy: 0.9708 - auc: 0.9936 - val_loss: 0.3113 - val_accuracy: 0.9874 - val_auc: 0.9990
Epoch 450/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3326 - accuracy: 0.9718 - auc: 0.9937 - val_loss: 0.3060 - val_accuracy: 0.9848 - val_auc: 0.9990
Epoch 451/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3356 - accuracy:

0.9708 - auc: 0.9946 - val_loss: 0.3077 - val_accuracy: 0.9874 - val_auc: 0.9990
Epoch 452/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3320 - accuracy: 0.9724 - auc: 0.9935 - val_loss: 0.3071 - val_accuracy: 0.9874 - val_auc: 0.9990
Epoch 453/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3312 - accuracy: 0.9681 - auc: 0.9940 - val_loss: 0.3119 - val_accuracy: 0.9899 - val_auc: 0.9989
Epoch 454/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3393 - accuracy: 0.9702 - auc: 0.9928 - val_loss: 0.3060 - val_accuracy: 0.9848 - val_auc: 0.9991
Epoch 455/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3342 - accuracy: 0.9702 - auc: 0.9923 - val_loss: 0.3062 - val_accuracy: 0.9874 - val_auc: 0.9990
Epoch 456/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3349 - accuracy: 0.9681 - auc: 0.9927 - val_loss: 0.3087 - val_accuracy: 0.9899 - val_auc: 0.9989
Epoch 457/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3348 - accuracy: 0.9729 - auc: 0.9925 - val_loss: 0.3058 - val_accuracy: 0.9874 - val_auc: 0.9991
Epoch 458/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3324 - accuracy: 0.9653 - auc: 0.9939 - val_loss: 0.3036 - val_accuracy: 0.9874 - val_auc: 0.9990
Epoch 459/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3352 - accuracy: 0.9675 - auc: 0.9911 - val_loss: 0.3071 - val_accuracy: 0.9899 - val_auc: 0.9989
Epoch 460/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3269 - accuracy: 0.9686 - auc: 0.9946 - val_loss: 0.3054 - val_accuracy: 0.9924 - val_auc: 0.9990
Epoch 461/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3440 - accuracy: 0.9605 - auc: 0.9906 - val_loss: 0.3062 - val_accuracy: 0.9899 - val_auc: 0.9990
Epoch 462/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3286 - accuracy: 0.9708 - auc: 0.9930 - val_loss: 0.3015 - val_accuracy: 0.9899 - val_auc: 0.9990
Epoch 463/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3304 - accuracy: 0.9691 - auc: 0.9935 - val_loss: 0.2997 - val_accuracy: 0.9874 - val_auc: 0.9992
Epoch 464/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3359 - accuracy: 0.9653 - auc: 0.9920 - val_loss: 0.3074 - val_accuracy: 0.9899 - val_auc: 0.9989
Epoch 465/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3379 - accuracy: 0.9653 - auc: 0.9915 - val_loss: 0.2966 - val_accuracy: 0.9848 - val_auc: 0.9992
Epoch 466/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3248 - accuracy: 0.9746 - auc: 0.9922 - val_loss: 0.2976 - val_accuracy: 0.9848 - val_auc: 0.9992
Epoch 467/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3249 - accuracy:

0.9691 - auc: 0.9935 - val_loss: 0.2979 - val_accuracy: 0.9899 - val_auc: 0.9991
Epoch 468/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3297 - accuracy: 0.9691 - auc: 0.9913 - val_loss: 0.2971 - val_accuracy: 0.9899 - val_auc: 0.9991
Epoch 469/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3293 - accuracy: 0.9664 - auc: 0.9927 - val_loss: 0.2972 - val_accuracy: 0.9924 - val_auc: 0.9992
Epoch 470/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3240 - accuracy: 0.9713 - auc: 0.9945 - val_loss: 0.2954 - val_accuracy: 0.9899 - val_auc: 0.9992
Epoch 471/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3253 - accuracy: 0.9681 - auc: 0.9932 - val_loss: 0.2960 - val_accuracy: 0.9874 - val_auc: 0.9991
Epoch 472/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3270 - accuracy: 0.9686 - auc: 0.9928 - val_loss: 0.2963 - val_accuracy: 0.9874 - val_auc: 0.9992
Epoch 473/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3267 - accuracy: 0.9713 - auc: 0.9920 - val_loss: 0.2959 - val_accuracy: 0.9848 - val_auc: 0.9992
Epoch 474/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3308 - accuracy: 0.9702 - auc: 0.9917 - val_loss: 0.2934 - val_accuracy: 0.9874 - val_auc: 0.9992
Epoch 475/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3214 - accuracy: 0.9718 - auc: 0.9933 - val_loss: 0.2933 - val_accuracy: 0.9874 - val_auc: 0.9991
Epoch 476/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3186 - accuracy: 0.9729 - auc: 0.9934 - val_loss: 0.2907 - val_accuracy: 0.9848 - val_auc: 0.9992
Epoch 477/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3175 - accuracy: 0.9718 - auc: 0.9943 - val_loss: 0.2931 - val_accuracy: 0.9899 - val_auc: 0.9993
Epoch 478/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3248 - accuracy: 0.9702 - auc: 0.9927 - val_loss: 0.2910 - val_accuracy: 0.9848 - val_auc: 0.9992
Epoch 479/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3224 - accuracy: 0.9718 - auc: 0.9914 - val_loss: 0.2931 - val_accuracy: 0.9899 - val_auc: 0.9991
Epoch 480/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3333 - accuracy: 0.9643 - auc: 0.9917 - val_loss: 0.2901 - val_accuracy: 0.9848 - val_auc: 0.9992
Epoch 481/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3283 - accuracy: 0.9708 - auc: 0.9912 - val_loss: 0.2893 - val_accuracy: 0.9848 - val_auc: 0.9993
Epoch 482/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3170 - accuracy: 0.9713 - auc: 0.9942 - val_loss: 0.2893 - val_accuracy: 0.9899 - val_auc: 0.9992
Epoch 483/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3095 - accuracy:

0.9756 - auc: 0.9948 - val_loss: 0.2891 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 484/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3170 - accuracy: 0.9697 - auc: 0.9945 - val_loss: 0.2918 - val_accuracy: 0.9924 - val_auc: 0.9992
Epoch 485/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3190 - accuracy: 0.9713 - auc: 0.9937 - val_loss: 0.2903 - val_accuracy: 0.9924 - val_auc: 0.9991
Epoch 486/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3136 - accuracy: 0.9740 - auc: 0.9932 - val_loss: 0.2888 - val_accuracy: 0.9874 - val_auc: 0.9991
Epoch 487/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3089 - accuracy: 0.9735 - auc: 0.9948 - val_loss: 0.2888 - val_accuracy: 0.9899 - val_auc: 0.9992
Epoch 488/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3093 - accuracy: 0.9762 - auc: 0.9947 - val_loss: 0.2854 - val_accuracy: 0.9899 - val_auc: 0.9993
Epoch 489/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3167 - accuracy: 0.9713 - auc: 0.9938 - val_loss: 0.2870 - val_accuracy: 0.9899 - val_auc: 0.9992
Epoch 490/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3159 - accuracy: 0.9708 - auc: 0.9942 - val_loss: 0.2847 - val_accuracy: 0.9848 - val_auc: 0.9992
Epoch 491/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3236 - accuracy: 0.9670 - auc: 0.9915 - val_loss: 0.2868 - val_accuracy: 0.9924 - val_auc: 0.9992
Epoch 492/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3103 - accuracy: 0.9735 - auc: 0.9938 - val_loss: 0.2863 - val_accuracy: 0.9924 - val_auc: 0.9992
Epoch 493/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3152 - accuracy: 0.9708 - auc: 0.9928 - val_loss: 0.2835 - val_accuracy: 0.9848 - val_auc: 0.9993
Epoch 494/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3064 - accuracy: 0.9729 - auc: 0.9953 - val_loss: 0.2865 - val_accuracy: 0.9924 - val_auc: 0.9992
Epoch 495/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3122 - accuracy: 0.9697 - auc: 0.9934 - val_loss: 0.2863 - val_accuracy: 0.9924 - val_auc: 0.9992
Epoch 496/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3076 - accuracy: 0.9681 - auc: 0.9954 - val_loss: 0.2895 - val_accuracy: 0.9899 - val_auc: 0.9991
Epoch 497/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3174 - accuracy: 0.9724 - auc: 0.9925 - val_loss: 0.2857 - val_accuracy: 0.9924 - val_auc: 0.9992
Epoch 498/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3060 - accuracy: 0.9735 - auc: 0.9944 - val_loss: 0.2815 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 499/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3066 - accuracy:

0.9713 - auc: 0.9944 - val_loss: 0.2801 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 500/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3032 - accuracy: 0.9751 - auc: 0.9942 - val_loss: 0.2792 - val_accuracy: 0.9899 - val_auc: 0.9994
Epoch 501/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3126 - accuracy: 0.9681 - auc: 0.9927 - val_loss: 0.2805 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 502/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3031 - accuracy: 0.9746 - auc: 0.9938 - val_loss: 0.2801 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 503/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3181 - accuracy: 0.9670 - auc: 0.9918 - val_loss: 0.2756 - val_accuracy: 0.9899 - val_auc: 0.9994
Epoch 504/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3069 - accuracy: 0.9729 - auc: 0.9937 - val_loss: 0.2791 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 505/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3049 - accuracy: 0.9697 - auc: 0.9952 - val_loss: 0.2773 - val_accuracy: 0.9874 - val_auc: 0.9993
Epoch 506/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3012 - accuracy: 0.9718 - auc: 0.9942 - val_loss: 0.2779 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 507/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3174 - accuracy: 0.9681 - auc: 0.9907 - val_loss: 0.2762 - val_accuracy: 0.9899 - val_auc: 0.9995
Epoch 508/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3006 - accuracy: 0.9756 - auc: 0.9934 - val_loss: 0.2779 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 509/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3007 - accuracy: 0.9735 - auc: 0.9945 - val_loss: 0.2757 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 510/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3026 - accuracy: 0.9746 - auc: 0.9940 - val_loss: 0.2734 - val_accuracy: 0.9848 - val_auc: 0.9994
Epoch 511/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3115 - accuracy: 0.9735 - auc: 0.9930 - val_loss: 0.2749 - val_accuracy: 0.9899 - val_auc: 0.9994
Epoch 512/1000
58/58 [=====] - 1s 22ms/step - loss: 0.3033 - accuracy: 0.9713 - auc: 0.9947 - val_loss: 0.2718 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 513/1000
58/58 [=====] - 1s 23ms/step - loss: 0.3017 - accuracy: 0.9708 - auc: 0.9934 - val_loss: 0.2732 - val_accuracy: 0.9874 - val_auc: 0.9993
Epoch 514/1000
58/58 [=====] - 1s 22ms/step - loss: 0.3082 - accuracy: 0.9691 - auc: 0.9932 - val_loss: 0.2717 - val_accuracy: 0.9899 - val_auc: 0.9994
Epoch 515/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3017 - accuracy:

```
0.9708 - auc: 0.9933 - val_loss: 0.2733 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 516/1000
58/58 [=====] - 1s 22ms/step - loss: 0.3077 - accuracy: 0.9675 - auc: 0.9928 - val_loss: 0.2732 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 517/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2937 - accuracy: 0.9751 - auc: 0.9949 - val_loss: 0.2717 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 518/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3114 - accuracy: 0.9681 - auc: 0.9921 - val_loss: 0.2769 - val_accuracy: 0.9899 - val_auc: 0.9992
Epoch 519/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3029 - accuracy: 0.9675 - auc: 0.9941 - val_loss: 0.2690 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 520/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3013 - accuracy: 0.9729 - auc: 0.9941 - val_loss: 0.2720 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 521/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3002 - accuracy: 0.9686 - auc: 0.9938 - val_loss: 0.2683 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 522/1000
58/58 [=====] - 1s 22ms/step - loss: 0.3034 - accuracy: 0.9691 - auc: 0.9923 - val_loss: 0.2713 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 523/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2922 - accuracy: 0.9767 - auc: 0.9943 - val_loss: 0.2691 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 524/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2946 - accuracy: 0.9751 - auc: 0.9951 - val_loss: 0.2675 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 525/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2940 - accuracy: 0.9794 - auc: 0.9941 - val_loss: 0.2690 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 526/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2921 - accuracy: 0.9735 - auc: 0.9948 - val_loss: 0.2664 - val_accuracy: 0.9899 - val_auc: 0.9994
Epoch 527/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2979 - accuracy: 0.9718 - auc: 0.9932 - val_loss: 0.2699 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 528/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2953 - accuracy: 0.9724 - auc: 0.9943 - val_loss: 0.2650 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 529/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2884 - accuracy: 0.9762 - auc: 0.9960 - val_loss: 0.2650 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 530/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2938 - accuracy: 0.9746 - auc: 0.9940 - val_loss: 0.2687 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 531/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2983 - accuracy:
```

0.9713 - auc: 0.9926 - val_loss: 0.2694 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 532/1000
58/58 [=====] - 1s 21ms/step - loss: 0.3000 - accuracy: 0.9653 - auc: 0.9931 - val_loss: 0.2619 - val_accuracy: 0.9874 - val_auc: 0.9995
Epoch 533/1000
58/58 [=====] - 1s 20ms/step - loss: 0.3089 - accuracy: 0.9686 - auc: 0.9897 - val_loss: 0.2669 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 534/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2938 - accuracy: 0.9713 - auc: 0.9938 - val_loss: 0.2656 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 535/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2952 - accuracy: 0.9735 - auc: 0.9936 - val_loss: 0.2648 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 536/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2976 - accuracy: 0.9708 - auc: 0.9920 - val_loss: 0.2616 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 537/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2957 - accuracy: 0.9718 - auc: 0.9929 - val_loss: 0.2652 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 538/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2870 - accuracy: 0.9751 - auc: 0.9938 - val_loss: 0.2644 - val_accuracy: 0.9924 - val_auc: 0.9993
Epoch 539/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2849 - accuracy: 0.9751 - auc: 0.9944 - val_loss: 0.2616 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 540/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2912 - accuracy: 0.9729 - auc: 0.9935 - val_loss: 0.2643 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 541/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2860 - accuracy: 0.9724 - auc: 0.9944 - val_loss: 0.2635 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 542/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2965 - accuracy: 0.9718 - auc: 0.9923 - val_loss: 0.2597 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 543/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2867 - accuracy: 0.9702 - auc: 0.9956 - val_loss: 0.2574 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 544/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2831 - accuracy: 0.9756 - auc: 0.9945 - val_loss: 0.2598 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 545/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2810 - accuracy: 0.9783 - auc: 0.9951 - val_loss: 0.2572 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 546/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2869 - accuracy: 0.9724 - auc: 0.9956 - val_loss: 0.2594 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 547/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2868 - accuracy:

0.9740 - auc: 0.9944 - val_loss: 0.2578 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 548/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2857 - accuracy: 0.9740 - auc: 0.9935 - val_loss: 0.2565 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 549/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2863 - accuracy: 0.9718 - auc: 0.9940 - val_loss: 0.2565 - val_accuracy: 0.9899 - val_auc: 0.9995
Epoch 550/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2843 - accuracy: 0.9724 - auc: 0.9941 - val_loss: 0.2560 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 551/1000
58/58 [=====] - 1s 19ms/step - loss: 0.2763 - accuracy: 0.9762 - auc: 0.9956 - val_loss: 0.2570 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 552/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2824 - accuracy: 0.9740 - auc: 0.9946 - val_loss: 0.2537 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 553/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2910 - accuracy: 0.9718 - auc: 0.9928 - val_loss: 0.2625 - val_accuracy: 0.9899 - val_auc: 0.9992
Epoch 554/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2873 - accuracy: 0.9702 - auc: 0.9935 - val_loss: 0.2521 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 555/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2766 - accuracy: 0.9746 - auc: 0.9957 - val_loss: 0.2590 - val_accuracy: 0.9949 - val_auc: 0.9994
Epoch 556/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2779 - accuracy: 0.9751 - auc: 0.9944 - val_loss: 0.2533 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 557/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2844 - accuracy: 0.9713 - auc: 0.9939 - val_loss: 0.2559 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 558/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2770 - accuracy: 0.9762 - auc: 0.9954 - val_loss: 0.2527 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 559/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2846 - accuracy: 0.9735 - auc: 0.9939 - val_loss: 0.2517 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 560/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2775 - accuracy: 0.9729 - auc: 0.9944 - val_loss: 0.2498 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 561/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2848 - accuracy: 0.9713 - auc: 0.9945 - val_loss: 0.2516 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 562/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2807 - accuracy: 0.9729 - auc: 0.9937 - val_loss: 0.2493 - val_accuracy: 0.9874 - val_auc: 0.9995
Epoch 563/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2807 - accuracy:

0.9756 - auc: 0.9934 - val_loss: 0.2491 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 564/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2832 - accuracy: 0.9724 - auc: 0.9932 - val_loss: 0.2496 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 565/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2778 - accuracy: 0.9740 - auc: 0.9943 - val_loss: 0.2514 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 566/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2763 - accuracy: 0.9729 - auc: 0.9952 - val_loss: 0.2483 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 567/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2709 - accuracy: 0.9767 - auc: 0.9959 - val_loss: 0.2548 - val_accuracy: 0.9899 - val_auc: 0.9992
Epoch 568/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2826 - accuracy: 0.9735 - auc: 0.9932 - val_loss: 0.2477 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 569/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2760 - accuracy: 0.9724 - auc: 0.9948 - val_loss: 0.2483 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 570/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2770 - accuracy: 0.9708 - auc: 0.9937 - val_loss: 0.2474 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 571/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2749 - accuracy: 0.9756 - auc: 0.9948 - val_loss: 0.2430 - val_accuracy: 0.9899 - val_auc: 0.9996
Epoch 572/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2750 - accuracy: 0.9724 - auc: 0.9947 - val_loss: 0.2498 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 573/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2682 - accuracy: 0.9778 - auc: 0.9962 - val_loss: 0.2453 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 574/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2806 - accuracy: 0.9702 - auc: 0.9934 - val_loss: 0.2458 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 575/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2711 - accuracy: 0.9729 - auc: 0.9950 - val_loss: 0.2492 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 576/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2795 - accuracy: 0.9691 - auc: 0.9927 - val_loss: 0.2491 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 577/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2768 - accuracy: 0.9708 - auc: 0.9946 - val_loss: 0.2445 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 578/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2702 - accuracy: 0.9767 - auc: 0.9954 - val_loss: 0.2435 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 579/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2651 - accuracy:

0.9773 - auc: 0.9968 - val_loss: 0.2436 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 580/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2735 - accuracy: 0.9751 - auc: 0.9942 - val_loss: 0.2421 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 581/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2708 - accuracy: 0.9762 - auc: 0.9946 - val_loss: 0.2435 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 582/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2622 - accuracy: 0.9800 - auc: 0.9960 - val_loss: 0.2440 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 583/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2738 - accuracy: 0.9681 - auc: 0.9945 - val_loss: 0.2400 - val_accuracy: 0.9899 - val_auc: 0.9995
Epoch 584/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2658 - accuracy: 0.9762 - auc: 0.9957 - val_loss: 0.2419 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 585/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2646 - accuracy: 0.9751 - auc: 0.9948 - val_loss: 0.2431 - val_accuracy: 0.9924 - val_auc: 0.9994
Epoch 586/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2682 - accuracy: 0.9762 - auc: 0.9930 - val_loss: 0.2415 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 587/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2721 - accuracy: 0.9718 - auc: 0.9936 - val_loss: 0.2421 - val_accuracy: 0.9949 - val_auc: 0.9995
Epoch 588/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2639 - accuracy: 0.9767 - auc: 0.9946 - val_loss: 0.2395 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 589/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2692 - accuracy: 0.9718 - auc: 0.9942 - val_loss: 0.2383 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 590/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2660 - accuracy: 0.9740 - auc: 0.9949 - val_loss: 0.2391 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 591/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2668 - accuracy: 0.9735 - auc: 0.9939 - val_loss: 0.2391 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 592/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2731 - accuracy: 0.9702 - auc: 0.9937 - val_loss: 0.2375 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 593/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2645 - accuracy: 0.9773 - auc: 0.9959 - val_loss: 0.2395 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 594/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2756 - accuracy: 0.9697 - auc: 0.9933 - val_loss: 0.2364 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 595/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2586 - accuracy:

0.9746 - auc: 0.9968 - val_loss: 0.2374 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 596/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2658 - accuracy: 0.9767 - auc: 0.9943 - val_loss: 0.2397 - val_accuracy: 0.9949 - val_auc: 0.9995
Epoch 597/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2535 - accuracy: 0.9816 - auc: 0.9964 - val_loss: 0.2403 - val_accuracy: 0.9949 - val_auc: 0.9995
Epoch 598/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2625 - accuracy: 0.9735 - auc: 0.9949 - val_loss: 0.2405 - val_accuracy: 0.9949 - val_auc: 0.9994
Epoch 599/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2578 - accuracy: 0.9789 - auc: 0.9958 - val_loss: 0.2342 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 600/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2555 - accuracy: 0.9794 - auc: 0.9956 - val_loss: 0.2331 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 601/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2585 - accuracy: 0.9783 - auc: 0.9951 - val_loss: 0.2332 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 602/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2652 - accuracy: 0.9783 - auc: 0.9938 - val_loss: 0.2347 - val_accuracy: 0.9949 - val_auc: 0.9995
Epoch 603/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2650 - accuracy: 0.9740 - auc: 0.9940 - val_loss: 0.2318 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 604/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2607 - accuracy: 0.9735 - auc: 0.9956 - val_loss: 0.2371 - val_accuracy: 0.9949 - val_auc: 0.9995
Epoch 605/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2654 - accuracy: 0.9691 - auc: 0.9945 - val_loss: 0.2321 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 606/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2641 - accuracy: 0.9740 - auc: 0.9943 - val_loss: 0.2326 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 607/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2570 - accuracy: 0.9756 - auc: 0.9956 - val_loss: 0.2344 - val_accuracy: 0.9949 - val_auc: 0.9995
Epoch 608/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2555 - accuracy: 0.9767 - auc: 0.9947 - val_loss: 0.2319 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 609/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2560 - accuracy: 0.9783 - auc: 0.9953 - val_loss: 0.2294 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 610/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2700 - accuracy: 0.9681 - auc: 0.9928 - val_loss: 0.2369 - val_accuracy: 0.9949 - val_auc: 0.9994
Epoch 611/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2582 - accuracy:

0.9751 - auc: 0.9950 - val_loss: 0.2300 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 612/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2586 - accuracy: 0.9756 - auc: 0.9958 - val_loss: 0.2290 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 613/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2612 - accuracy: 0.9718 - auc: 0.9951 - val_loss: 0.2281 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 614/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2590 - accuracy: 0.9724 - auc: 0.9952 - val_loss: 0.2285 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 615/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2523 - accuracy: 0.9783 - auc: 0.9959 - val_loss: 0.2292 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 616/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2573 - accuracy: 0.9740 - auc: 0.9940 - val_loss: 0.2292 - val_accuracy: 0.9899 - val_auc: 0.9995
Epoch 617/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2683 - accuracy: 0.9670 - auc: 0.9944 - val_loss: 0.2281 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 618/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2633 - accuracy: 0.9713 - auc: 0.9930 - val_loss: 0.2250 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 619/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2498 - accuracy: 0.9773 - auc: 0.9949 - val_loss: 0.2254 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 620/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2512 - accuracy: 0.9773 - auc: 0.9954 - val_loss: 0.2241 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 621/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2457 - accuracy: 0.9811 - auc: 0.9969 - val_loss: 0.2238 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 622/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2559 - accuracy: 0.9735 - auc: 0.9949 - val_loss: 0.2289 - val_accuracy: 0.9949 - val_auc: 0.9995
Epoch 623/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2602 - accuracy: 0.9756 - auc: 0.9933 - val_loss: 0.2321 - val_accuracy: 0.9949 - val_auc: 0.9994
Epoch 624/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2512 - accuracy: 0.9746 - auc: 0.9954 - val_loss: 0.2247 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 625/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2583 - accuracy: 0.9729 - auc: 0.9938 - val_loss: 0.2234 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 626/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2523 - accuracy: 0.9767 - auc: 0.9950 - val_loss: 0.2228 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 627/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2481 - accuracy:

0.9740 - auc: 0.9963 - val_loss: 0.2240 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 628/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2531 - accuracy: 0.9740 - auc: 0.9950 - val_loss: 0.2285 - val_accuracy: 0.9949 - val_auc: 0.9995
Epoch 629/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2556 - accuracy: 0.9762 - auc: 0.9948 - val_loss: 0.2244 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 630/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2545 - accuracy: 0.9729 - auc: 0.9943 - val_loss: 0.2223 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 631/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2568 - accuracy: 0.9751 - auc: 0.9927 - val_loss: 0.2240 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 632/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2494 - accuracy: 0.9762 - auc: 0.9942 - val_loss: 0.2220 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 633/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2513 - accuracy: 0.9773 - auc: 0.9946 - val_loss: 0.2230 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 634/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2500 - accuracy: 0.9762 - auc: 0.9950 - val_loss: 0.2243 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 635/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2475 - accuracy: 0.9762 - auc: 0.9948 - val_loss: 0.2210 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 636/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2460 - accuracy: 0.9783 - auc: 0.9959 - val_loss: 0.2234 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 637/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2591 - accuracy: 0.9713 - auc: 0.9917 - val_loss: 0.2183 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 638/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2509 - accuracy: 0.9756 - auc: 0.9949 - val_loss: 0.2185 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 639/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2450 - accuracy: 0.9767 - auc: 0.9959 - val_loss: 0.2189 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 640/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2526 - accuracy: 0.9767 - auc: 0.9937 - val_loss: 0.2187 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 641/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2487 - accuracy: 0.9751 - auc: 0.9950 - val_loss: 0.2190 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 642/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2496 - accuracy: 0.9756 - auc: 0.9935 - val_loss: 0.2172 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 643/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2492 - accuracy:

0.9729 - auc: 0.9953 - val_loss: 0.2176 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 644/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2479 - accuracy: 0.9805 - auc: 0.9946 - val_loss: 0.2175 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 645/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2436 - accuracy: 0.9789 - auc: 0.9948 - val_loss: 0.2215 - val_accuracy: 0.9949 - val_auc: 0.9995
Epoch 646/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2466 - accuracy: 0.9773 - auc: 0.9951 - val_loss: 0.2181 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 647/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2478 - accuracy: 0.9751 - auc: 0.9949 - val_loss: 0.2188 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 648/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2419 - accuracy: 0.9794 - auc: 0.9953 - val_loss: 0.2178 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 649/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2457 - accuracy: 0.9762 - auc: 0.9955 - val_loss: 0.2179 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 650/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2360 - accuracy: 0.9805 - auc: 0.9972 - val_loss: 0.2145 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 651/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2426 - accuracy: 0.9767 - auc: 0.9948 - val_loss: 0.2158 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 652/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2370 - accuracy: 0.9805 - auc: 0.9962 - val_loss: 0.2170 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 653/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2421 - accuracy: 0.9767 - auc: 0.9961 - val_loss: 0.2175 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 654/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2438 - accuracy: 0.9746 - auc: 0.9950 - val_loss: 0.2124 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 655/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2379 - accuracy: 0.9794 - auc: 0.9964 - val_loss: 0.2140 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 656/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2490 - accuracy: 0.9718 - auc: 0.9940 - val_loss: 0.2212 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 657/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2334 - accuracy: 0.9789 - auc: 0.9964 - val_loss: 0.2171 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 658/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2424 - accuracy: 0.9783 - auc: 0.9955 - val_loss: 0.2163 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 659/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2460 - accuracy:

0.9740 - auc: 0.9937 - val_loss: 0.2143 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 660/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2414 - accuracy: 0.9773 - auc: 0.9957 - val_loss: 0.2128 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 661/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2366 - accuracy: 0.9789 - auc: 0.9960 - val_loss: 0.2097 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 662/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2305 - accuracy: 0.9832 - auc: 0.9969 - val_loss: 0.2098 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 663/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2452 - accuracy: 0.9746 - auc: 0.9938 - val_loss: 0.2096 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 664/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2413 - accuracy: 0.9756 - auc: 0.9956 - val_loss: 0.2096 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 665/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2305 - accuracy: 0.9800 - auc: 0.9974 - val_loss: 0.2107 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 666/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2367 - accuracy: 0.9751 - auc: 0.9953 - val_loss: 0.2112 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 667/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2372 - accuracy: 0.9735 - auc: 0.9957 - val_loss: 0.2086 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 668/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2374 - accuracy: 0.9773 - auc: 0.9952 - val_loss: 0.2102 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 669/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2469 - accuracy: 0.9735 - auc: 0.9933 - val_loss: 0.2120 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 670/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2430 - accuracy: 0.9735 - auc: 0.9943 - val_loss: 0.2070 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 671/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2407 - accuracy: 0.9756 - auc: 0.9954 - val_loss: 0.2066 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 672/1000
58/58 [=====] - 1s 23ms/step - loss: 0.2273 - accuracy: 0.9821 - auc: 0.9960 - val_loss: 0.2081 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 673/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2321 - accuracy: 0.9794 - auc: 0.9970 - val_loss: 0.2065 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 674/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2384 - accuracy: 0.9746 - auc: 0.9952 - val_loss: 0.2053 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 675/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2338 - accuracy:

0.9767 - auc: 0.9958 - val_loss: 0.2081 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 676/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2342 - accuracy: 0.9794 - auc: 0.9950 - val_loss: 0.2084 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 677/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2377 - accuracy: 0.9783 - auc: 0.9946 - val_loss: 0.2098 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 678/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2346 - accuracy: 0.9794 - auc: 0.9948 - val_loss: 0.2084 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 679/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2305 - accuracy: 0.9800 - auc: 0.9955 - val_loss: 0.2039 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 680/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2403 - accuracy: 0.9783 - auc: 0.9928 - val_loss: 0.2024 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 681/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2319 - accuracy: 0.9789 - auc: 0.9964 - val_loss: 0.2078 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 682/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2326 - accuracy: 0.9762 - auc: 0.9962 - val_loss: 0.2057 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 683/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2295 - accuracy: 0.9783 - auc: 0.9959 - val_loss: 0.2032 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 684/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2201 - accuracy: 0.9843 - auc: 0.9967 - val_loss: 0.2058 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 685/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2340 - accuracy: 0.9767 - auc: 0.9954 - val_loss: 0.2020 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 686/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2302 - accuracy: 0.9783 - auc: 0.9955 - val_loss: 0.2011 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 687/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2281 - accuracy: 0.9751 - auc: 0.9965 - val_loss: 0.2045 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 688/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2264 - accuracy: 0.9794 - auc: 0.9972 - val_loss: 0.2019 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 689/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2219 - accuracy: 0.9800 - auc: 0.9973 - val_loss: 0.2022 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 690/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2260 - accuracy: 0.9773 - auc: 0.9965 - val_loss: 0.2034 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 691/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2371 - accuracy:

0.9751 - auc: 0.9940 - val_loss: 0.2007 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 692/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2279 - accuracy: 0.9794 - auc: 0.9959 - val_loss: 0.2015 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 693/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2292 - accuracy: 0.9773 - auc: 0.9963 - val_loss: 0.1991 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 694/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2327 - accuracy: 0.9767 - auc: 0.9950 - val_loss: 0.2026 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 695/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2197 - accuracy: 0.9800 - auc: 0.9971 - val_loss: 0.2009 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 696/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2337 - accuracy: 0.9751 - auc: 0.9944 - val_loss: 0.1997 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 697/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2248 - accuracy: 0.9783 - auc: 0.9963 - val_loss: 0.1987 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 698/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2292 - accuracy: 0.9762 - auc: 0.9955 - val_loss: 0.1995 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 699/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2197 - accuracy: 0.9816 - auc: 0.9969 - val_loss: 0.1993 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 700/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2237 - accuracy: 0.9821 - auc: 0.9960 - val_loss: 0.1982 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 701/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2303 - accuracy: 0.9756 - auc: 0.9952 - val_loss: 0.2064 - val_accuracy: 0.9924 - val_auc: 0.9995
Epoch 702/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2211 - accuracy: 0.9811 - auc: 0.9963 - val_loss: 0.1959 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 703/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2264 - accuracy: 0.9794 - auc: 0.9950 - val_loss: 0.2040 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 704/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2282 - accuracy: 0.9751 - auc: 0.9948 - val_loss: 0.1966 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 705/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2255 - accuracy: 0.9751 - auc: 0.9959 - val_loss: 0.1960 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 706/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2209 - accuracy: 0.9789 - auc: 0.9956 - val_loss: 0.1986 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 707/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2220 - accuracy:

0.9789 - auc: 0.9962 - val_loss: 0.1947 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 708/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2331 - accuracy: 0.9751 - auc: 0.9947 - val_loss: 0.1974 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 709/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2210 - accuracy: 0.9794 - auc: 0.9967 - val_loss: 0.1977 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 710/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2218 - accuracy: 0.9794 - auc: 0.9962 - val_loss: 0.1971 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 711/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2292 - accuracy: 0.9746 - auc: 0.9946 - val_loss: 0.1963 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 712/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2267 - accuracy: 0.9778 - auc: 0.9957 - val_loss: 0.1946 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 713/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2283 - accuracy: 0.9740 - auc: 0.9955 - val_loss: 0.1955 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 714/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2185 - accuracy: 0.9778 - auc: 0.9961 - val_loss: 0.1933 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 715/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2315 - accuracy: 0.9762 - auc: 0.9933 - val_loss: 0.1987 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 716/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2212 - accuracy: 0.9778 - auc: 0.9963 - val_loss: 0.1966 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 717/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2189 - accuracy: 0.9805 - auc: 0.9962 - val_loss: 0.1944 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 718/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2276 - accuracy: 0.9751 - auc: 0.9941 - val_loss: 0.1953 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 719/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2140 - accuracy: 0.9854 - auc: 0.9970 - val_loss: 0.1930 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 720/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2167 - accuracy: 0.9794 - auc: 0.9968 - val_loss: 0.1907 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 721/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2280 - accuracy: 0.9762 - auc: 0.9939 - val_loss: 0.1905 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 722/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2133 - accuracy: 0.9821 - auc: 0.9973 - val_loss: 0.1951 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 723/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2180 - accuracy:

0.9783 - auc: 0.9955 - val_loss: 0.1917 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 724/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2210 - accuracy: 0.9789 - auc: 0.9949 - val_loss: 0.1901 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 725/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2154 - accuracy: 0.9805 - auc: 0.9962 - val_loss: 0.1931 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 726/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2203 - accuracy: 0.9767 - auc: 0.9949 - val_loss: 0.1893 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 727/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2165 - accuracy: 0.9821 - auc: 0.9961 - val_loss: 0.1894 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 728/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2083 - accuracy: 0.9832 - auc: 0.9977 - val_loss: 0.1895 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 729/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2222 - accuracy: 0.9778 - auc: 0.9957 - val_loss: 0.1919 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 730/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2184 - accuracy: 0.9783 - auc: 0.9959 - val_loss: 0.1899 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 731/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2099 - accuracy: 0.9848 - auc: 0.9969 - val_loss: 0.1889 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 732/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2173 - accuracy: 0.9778 - auc: 0.9960 - val_loss: 0.1946 - val_accuracy: 0.9924 - val_auc: 0.9996
Epoch 733/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2121 - accuracy: 0.9783 - auc: 0.9964 - val_loss: 0.1893 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 734/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2240 - accuracy: 0.9724 - auc: 0.9954 - val_loss: 0.1866 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 735/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2182 - accuracy: 0.9762 - auc: 0.9959 - val_loss: 0.1872 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 736/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2151 - accuracy: 0.9794 - auc: 0.9965 - val_loss: 0.1875 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 737/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2087 - accuracy: 0.9805 - auc: 0.9969 - val_loss: 0.1890 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 738/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2102 - accuracy: 0.9811 - auc: 0.9964 - val_loss: 0.1856 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 739/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2142 - accuracy:

0.9789 - auc: 0.9947 - val_loss: 0.1859 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 740/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2160 - accuracy: 0.9794 - auc: 0.9940 - val_loss: 0.1854 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 741/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2091 - accuracy: 0.9811 - auc: 0.9969 - val_loss: 0.1873 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 742/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2107 - accuracy: 0.9789 - auc: 0.9967 - val_loss: 0.1888 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 743/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2146 - accuracy: 0.9762 - auc: 0.9968 - val_loss: 0.1844 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 744/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2035 - accuracy: 0.9843 - auc: 0.9969 - val_loss: 0.1854 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 745/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2115 - accuracy: 0.9811 - auc: 0.9963 - val_loss: 0.1861 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 746/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2128 - accuracy: 0.9794 - auc: 0.9964 - val_loss: 0.1914 - val_accuracy: 0.9949 - val_auc: 0.9996
Epoch 747/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2061 - accuracy: 0.9811 - auc: 0.9975 - val_loss: 0.1821 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 748/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2120 - accuracy: 0.9800 - auc: 0.9960 - val_loss: 0.1829 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 749/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2138 - accuracy: 0.9783 - auc: 0.9964 - val_loss: 0.1845 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 750/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2054 - accuracy: 0.9805 - auc: 0.9964 - val_loss: 0.1841 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 751/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2121 - accuracy: 0.9789 - auc: 0.9965 - val_loss: 0.1832 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 752/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2052 - accuracy: 0.9805 - auc: 0.9970 - val_loss: 0.1819 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 753/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2075 - accuracy: 0.9811 - auc: 0.9963 - val_loss: 0.1837 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 754/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2085 - accuracy: 0.9800 - auc: 0.9962 - val_loss: 0.1835 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 755/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2026 - accuracy:

0.9854 - auc: 0.9967 - val_loss: 0.1810 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 756/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2066 - accuracy: 0.9805 - auc: 0.9968 - val_loss: 0.1847 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 757/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2102 - accuracy: 0.9789 - auc: 0.9961 - val_loss: 0.1809 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 758/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2061 - accuracy: 0.9800 - auc: 0.9968 - val_loss: 0.1805 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 759/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2158 - accuracy: 0.9751 - auc: 0.9951 - val_loss: 0.1836 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 760/1000
58/58 [=====] - 1s 22ms/step - loss: 0.2146 - accuracy: 0.9718 - auc: 0.9961 - val_loss: 0.1790 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 761/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2042 - accuracy: 0.9821 - auc: 0.9969 - val_loss: 0.1790 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 762/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2071 - accuracy: 0.9811 - auc: 0.9952 - val_loss: 0.1859 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 763/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2079 - accuracy: 0.9789 - auc: 0.9962 - val_loss: 0.1786 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 764/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2047 - accuracy: 0.9800 - auc: 0.9968 - val_loss: 0.1818 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 765/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2058 - accuracy: 0.9783 - auc: 0.9962 - val_loss: 0.1795 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 766/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2016 - accuracy: 0.9805 - auc: 0.9970 - val_loss: 0.1788 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 767/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2079 - accuracy: 0.9773 - auc: 0.9968 - val_loss: 0.1787 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 768/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2034 - accuracy: 0.9794 - auc: 0.9969 - val_loss: 0.1805 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 769/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1986 - accuracy: 0.9838 - auc: 0.9971 - val_loss: 0.1798 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 770/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2053 - accuracy: 0.9789 - auc: 0.9968 - val_loss: 0.1773 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 771/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2037 - accuracy:

0.9778 - auc: 0.9962 - val_loss: 0.1787 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 772/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1993 - accuracy: 0.9816 - auc: 0.9972 - val_loss: 0.1819 - val_accuracy: 0.9924 - val_auc: 0.9998
Epoch 773/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2042 - accuracy: 0.9800 - auc: 0.9951 - val_loss: 0.1769 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 774/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1987 - accuracy: 0.9832 - auc: 0.9966 - val_loss: 0.1791 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 775/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2002 - accuracy: 0.9805 - auc: 0.9974 - val_loss: 0.1780 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 776/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2061 - accuracy: 0.9773 - auc: 0.9961 - val_loss: 0.1759 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 777/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2036 - accuracy: 0.9794 - auc: 0.9967 - val_loss: 0.1771 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 778/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2074 - accuracy: 0.9762 - auc: 0.9954 - val_loss: 0.1744 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 779/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1964 - accuracy: 0.9838 - auc: 0.9966 - val_loss: 0.1808 - val_accuracy: 0.9924 - val_auc: 0.9997
Epoch 780/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2034 - accuracy: 0.9800 - auc: 0.9956 - val_loss: 0.1747 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 781/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2015 - accuracy: 0.9816 - auc: 0.9958 - val_loss: 0.1753 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 782/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2002 - accuracy: 0.9805 - auc: 0.9965 - val_loss: 0.1751 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 783/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2005 - accuracy: 0.9811 - auc: 0.9964 - val_loss: 0.1751 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 784/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1984 - accuracy: 0.9789 - auc: 0.9974 - val_loss: 0.1754 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 785/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2029 - accuracy: 0.9767 - auc: 0.9970 - val_loss: 0.1722 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 786/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1996 - accuracy: 0.9816 - auc: 0.9968 - val_loss: 0.1742 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 787/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1961 - accuracy:

0.9794 - auc: 0.9971 - val_loss: 0.1729 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 788/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1967 - accuracy: 0.9800 - auc: 0.9970 - val_loss: 0.1726 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 789/1000
58/58 [=====] - 1s 21ms/step - loss: 0.2001 - accuracy: 0.9816 - auc: 0.9959 - val_loss: 0.1775 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 790/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1995 - accuracy: 0.9821 - auc: 0.9963 - val_loss: 0.1760 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 791/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1998 - accuracy: 0.9794 - auc: 0.9956 - val_loss: 0.1711 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 792/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1945 - accuracy: 0.9805 - auc: 0.9972 - val_loss: 0.1714 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 793/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1968 - accuracy: 0.9811 - auc: 0.9969 - val_loss: 0.1736 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 794/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1984 - accuracy: 0.9756 - auc: 0.9969 - val_loss: 0.1715 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 795/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1942 - accuracy: 0.9783 - auc: 0.9970 - val_loss: 0.1756 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 796/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2053 - accuracy: 0.9794 - auc: 0.9934 - val_loss: 0.1766 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 797/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1926 - accuracy: 0.9816 - auc: 0.9977 - val_loss: 0.1728 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 798/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1985 - accuracy: 0.9800 - auc: 0.9969 - val_loss: 0.1708 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 799/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1981 - accuracy: 0.9800 - auc: 0.9954 - val_loss: 0.1703 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 800/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2022 - accuracy: 0.9751 - auc: 0.9953 - val_loss: 0.1705 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 801/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1996 - accuracy: 0.9789 - auc: 0.9956 - val_loss: 0.1711 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 802/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1937 - accuracy: 0.9805 - auc: 0.9970 - val_loss: 0.1724 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 803/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1915 - accuracy:

0.9800 - auc: 0.9966 - val_loss: 0.1710 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 804/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1963 - accuracy: 0.9805 - auc: 0.9967 - val_loss: 0.1692 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 805/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1928 - accuracy: 0.9816 - auc: 0.9974 - val_loss: 0.1680 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 806/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1959 - accuracy: 0.9794 - auc: 0.9967 - val_loss: 0.1689 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 807/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1998 - accuracy: 0.9773 - auc: 0.9950 - val_loss: 0.1671 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 808/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1958 - accuracy: 0.9805 - auc: 0.9963 - val_loss: 0.1683 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 809/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1881 - accuracy: 0.9848 - auc: 0.9968 - val_loss: 0.1677 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 810/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1863 - accuracy: 0.9865 - auc: 0.9974 - val_loss: 0.1666 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 811/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1978 - accuracy: 0.9751 - auc: 0.9966 - val_loss: 0.1666 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 812/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1923 - accuracy: 0.9848 - auc: 0.9968 - val_loss: 0.1662 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 813/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1900 - accuracy: 0.9811 - auc: 0.9975 - val_loss: 0.1664 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 814/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1982 - accuracy: 0.9789 - auc: 0.9954 - val_loss: 0.1681 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 815/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1936 - accuracy: 0.9794 - auc: 0.9963 - val_loss: 0.1676 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 816/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1981 - accuracy: 0.9756 - auc: 0.9964 - val_loss: 0.1680 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 817/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1914 - accuracy: 0.9821 - auc: 0.9963 - val_loss: 0.1679 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 818/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1928 - accuracy: 0.9805 - auc: 0.9963 - val_loss: 0.1644 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 819/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1835 - accuracy:

0.9859 - auc: 0.9980 - val_loss: 0.1645 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 820/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1869 - accuracy: 0.9816 - auc: 0.9972 - val_loss: 0.1639 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 821/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1880 - accuracy: 0.9800 - auc: 0.9981 - val_loss: 0.1658 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 822/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1850 - accuracy: 0.9843 - auc: 0.9977 - val_loss: 0.1643 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 823/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1919 - accuracy: 0.9794 - auc: 0.9969 - val_loss: 0.1669 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 824/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1850 - accuracy: 0.9843 - auc: 0.9972 - val_loss: 0.1653 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 825/1000
58/58 [=====] - 1s 20ms/step - loss: 0.2044 - accuracy: 0.9746 - auc: 0.9940 - val_loss: 0.1656 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 826/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1901 - accuracy: 0.9794 - auc: 0.9965 - val_loss: 0.1633 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 827/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1955 - accuracy: 0.9811 - auc: 0.9950 - val_loss: 0.1635 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 828/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1875 - accuracy: 0.9821 - auc: 0.9970 - val_loss: 0.1618 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 829/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1887 - accuracy: 0.9838 - auc: 0.9970 - val_loss: 0.1640 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 830/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1926 - accuracy: 0.9767 - auc: 0.9962 - val_loss: 0.1619 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 831/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1851 - accuracy: 0.9816 - auc: 0.9971 - val_loss: 0.1621 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 832/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1816 - accuracy: 0.9859 - auc: 0.9968 - val_loss: 0.1643 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 833/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1831 - accuracy: 0.9832 - auc: 0.9968 - val_loss: 0.1633 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 834/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1939 - accuracy: 0.9794 - auc: 0.9960 - val_loss: 0.1599 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 835/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1863 - accuracy:

```
0.9838 - auc: 0.9974 - val_loss: 0.1626 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 836/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1804 - accuracy: 0.9854 - auc: 0.9975 - val_loss: 0.1600 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 837/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1830 - accuracy: 0.9821 - auc: 0.9977 - val_loss: 0.1631 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 838/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1870 - accuracy: 0.9800 - auc: 0.9969 - val_loss: 0.1615 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 839/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1837 - accuracy: 0.9821 - auc: 0.9961 - val_loss: 0.1622 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 840/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1964 - accuracy: 0.9767 - auc: 0.9948 - val_loss: 0.1606 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 841/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1824 - accuracy: 0.9838 - auc: 0.9967 - val_loss: 0.1609 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 842/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1881 - accuracy: 0.9789 - auc: 0.9963 - val_loss: 0.1601 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 843/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1868 - accuracy: 0.9827 - auc: 0.9959 - val_loss: 0.1619 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 844/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1828 - accuracy: 0.9821 - auc: 0.9961 - val_loss: 0.1609 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 845/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1902 - accuracy: 0.9778 - auc: 0.9957 - val_loss: 0.1597 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 846/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1887 - accuracy: 0.9767 - auc: 0.9962 - val_loss: 0.1636 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 847/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1914 - accuracy: 0.9767 - auc: 0.9950 - val_loss: 0.1600 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 848/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1822 - accuracy: 0.9838 - auc: 0.9970 - val_loss: 0.1562 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 849/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1841 - accuracy: 0.9821 - auc: 0.9968 - val_loss: 0.1576 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 850/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1825 - accuracy: 0.9811 - auc: 0.9961 - val_loss: 0.1580 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 851/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1827 - accuracy:
```

```
0.9832 - auc: 0.9975 - val_loss: 0.1575 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 852/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1797 - accuracy: 0.9843 - auc: 0.9977 - val_loss: 0.1615 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 853/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1878 - accuracy: 0.9805 - auc: 0.9942 - val_loss: 0.1569 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 854/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1786 - accuracy: 0.9816 - auc: 0.9978 - val_loss: 0.1567 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 855/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1855 - accuracy: 0.9805 - auc: 0.9967 - val_loss: 0.1594 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 856/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1854 - accuracy: 0.9778 - auc: 0.9958 - val_loss: 0.1582 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 857/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1826 - accuracy: 0.9794 - auc: 0.9975 - val_loss: 0.1598 - val_accuracy: 0.9975 - val_auc: 0.9998
Epoch 858/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1813 - accuracy: 0.9811 - auc: 0.9966 - val_loss: 0.1568 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 859/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1886 - accuracy: 0.9783 - auc: 0.9960 - val_loss: 0.1602 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 860/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1874 - accuracy: 0.9794 - auc: 0.9960 - val_loss: 0.1573 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 861/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1855 - accuracy: 0.9789 - auc: 0.9962 - val_loss: 0.1549 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 862/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1849 - accuracy: 0.9783 - auc: 0.9963 - val_loss: 0.1551 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 863/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1914 - accuracy: 0.9773 - auc: 0.9944 - val_loss: 0.1544 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 864/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1760 - accuracy: 0.9827 - auc: 0.9974 - val_loss: 0.1541 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 865/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1758 - accuracy: 0.9816 - auc: 0.9973 - val_loss: 0.1553 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 866/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1782 - accuracy: 0.9794 - auc: 0.9977 - val_loss: 0.1558 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 867/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1743 - accuracy:
```

0.9854 - auc: 0.9979 - val_loss: 0.1540 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 868/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1802 - accuracy: 0.9821 - auc: 0.9970 - val_loss: 0.1536 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 869/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1774 - accuracy: 0.9838 - auc: 0.9957 - val_loss: 0.1531 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 870/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1843 - accuracy: 0.9800 - auc: 0.9958 - val_loss: 0.1539 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 871/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1774 - accuracy: 0.9800 - auc: 0.9971 - val_loss: 0.1534 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 872/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1778 - accuracy: 0.9816 - auc: 0.9973 - val_loss: 0.1530 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 873/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1795 - accuracy: 0.9816 - auc: 0.9955 - val_loss: 0.1527 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 874/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1747 - accuracy: 0.9821 - auc: 0.9978 - val_loss: 0.1533 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 875/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1816 - accuracy: 0.9789 - auc: 0.9963 - val_loss: 0.1526 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 876/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1785 - accuracy: 0.9805 - auc: 0.9974 - val_loss: 0.1516 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 877/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1812 - accuracy: 0.9811 - auc: 0.9963 - val_loss: 0.1522 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 878/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1766 - accuracy: 0.9821 - auc: 0.9971 - val_loss: 0.1544 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 879/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1787 - accuracy: 0.9816 - auc: 0.9969 - val_loss: 0.1518 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 880/1000
58/58 [=====] - 1s 23ms/step - loss: 0.1746 - accuracy: 0.9811 - auc: 0.9978 - val_loss: 0.1531 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 881/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1783 - accuracy: 0.9816 - auc: 0.9970 - val_loss: 0.1505 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 882/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1729 - accuracy: 0.9838 - auc: 0.9979 - val_loss: 0.1505 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 883/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1754 - accuracy:

0.9794 - auc: 0.9976 - val_loss: 0.1537 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 884/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1815 - accuracy: 0.9800 - auc: 0.9953 - val_loss: 0.1496 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 885/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1726 - accuracy: 0.9827 - auc: 0.9978 - val_loss: 0.1515 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 886/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1745 - accuracy: 0.9816 - auc: 0.9977 - val_loss: 0.1497 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 887/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1803 - accuracy: 0.9783 - auc: 0.9959 - val_loss: 0.1533 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 888/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1778 - accuracy: 0.9767 - auc: 0.9974 - val_loss: 0.1504 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 889/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1776 - accuracy: 0.9794 - auc: 0.9961 - val_loss: 0.1509 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 890/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1740 - accuracy: 0.9848 - auc: 0.9969 - val_loss: 0.1507 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 891/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1703 - accuracy: 0.9854 - auc: 0.9973 - val_loss: 0.1499 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 892/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1732 - accuracy: 0.9832 - auc: 0.9969 - val_loss: 0.1497 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 893/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1842 - accuracy: 0.9767 - auc: 0.9959 - val_loss: 0.1555 - val_accuracy: 0.9949 - val_auc: 0.9997
Epoch 894/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1715 - accuracy: 0.9832 - auc: 0.9972 - val_loss: 0.1485 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 895/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1855 - accuracy: 0.9762 - auc: 0.9964 - val_loss: 0.1474 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 896/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1729 - accuracy: 0.9800 - auc: 0.9977 - val_loss: 0.1474 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 897/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1665 - accuracy: 0.9838 - auc: 0.9981 - val_loss: 0.1489 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 898/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1693 - accuracy: 0.9843 - auc: 0.9969 - val_loss: 0.1485 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 899/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1755 - accuracy:

0.9789 - auc: 0.9970 - val_loss: 0.1480 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 900/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1758 - accuracy: 0.9789 - auc: 0.9965 - val_loss: 0.1488 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 901/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1659 - accuracy: 0.9838 - auc: 0.9976 - val_loss: 0.1490 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 902/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1677 - accuracy: 0.9832 - auc: 0.9984 - val_loss: 0.1458 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 903/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1717 - accuracy: 0.9832 - auc: 0.9967 - val_loss: 0.1473 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 904/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1706 - accuracy: 0.9827 - auc: 0.9967 - val_loss: 0.1485 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 905/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1654 - accuracy: 0.9854 - auc: 0.9980 - val_loss: 0.1469 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 906/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1753 - accuracy: 0.9811 - auc: 0.9960 - val_loss: 0.1483 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 907/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1752 - accuracy: 0.9816 - auc: 0.9965 - val_loss: 0.1477 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 908/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1744 - accuracy: 0.9827 - auc: 0.9964 - val_loss: 0.1472 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 909/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1791 - accuracy: 0.9783 - auc: 0.9952 - val_loss: 0.1481 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 910/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1622 - accuracy: 0.9843 - auc: 0.9981 - val_loss: 0.1468 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 911/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1664 - accuracy: 0.9843 - auc: 0.9980 - val_loss: 0.1455 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 912/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1659 - accuracy: 0.9838 - auc: 0.9974 - val_loss: 0.1464 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 913/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1719 - accuracy: 0.9816 - auc: 0.9971 - val_loss: 0.1489 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 914/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1622 - accuracy: 0.9881 - auc: 0.9977 - val_loss: 0.1452 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 915/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1657 - accuracy:

0.9832 - auc: 0.9980 - val_loss: 0.1435 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 916/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1635 - accuracy: 0.9832 - auc: 0.9975 - val_loss: 0.1454 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 917/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1747 - accuracy: 0.9794 - auc: 0.9962 - val_loss: 0.1444 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 918/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1731 - accuracy: 0.9832 - auc: 0.9965 - val_loss: 0.1493 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 919/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1665 - accuracy: 0.9848 - auc: 0.9975 - val_loss: 0.1435 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 920/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1681 - accuracy: 0.9821 - auc: 0.9977 - val_loss: 0.1445 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 921/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1721 - accuracy: 0.9800 - auc: 0.9964 - val_loss: 0.1438 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 922/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1662 - accuracy: 0.9838 - auc: 0.9978 - val_loss: 0.1421 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 923/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1682 - accuracy: 0.9811 - auc: 0.9971 - val_loss: 0.1426 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 924/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1654 - accuracy: 0.9859 - auc: 0.9968 - val_loss: 0.1449 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 925/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1673 - accuracy: 0.9805 - auc: 0.9957 - val_loss: 0.1430 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 926/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1676 - accuracy: 0.9816 - auc: 0.9976 - val_loss: 0.1458 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 927/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1619 - accuracy: 0.9865 - auc: 0.9980 - val_loss: 0.1419 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 928/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1710 - accuracy: 0.9816 - auc: 0.9969 - val_loss: 0.1395 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 929/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1657 - accuracy: 0.9827 - auc: 0.9973 - val_loss: 0.1430 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 930/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1617 - accuracy: 0.9848 - auc: 0.9980 - val_loss: 0.1414 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 931/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1692 - accuracy:

0.9816 - auc: 0.9965 - val_loss: 0.1413 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 932/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1626 - accuracy: 0.9854 - auc: 0.9976 - val_loss: 0.1416 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 933/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1618 - accuracy: 0.9821 - auc: 0.9984 - val_loss: 0.1428 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 934/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1726 - accuracy: 0.9783 - auc: 0.9963 - val_loss: 0.1406 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 935/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1649 - accuracy: 0.9821 - auc: 0.9968 - val_loss: 0.1421 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 936/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1655 - accuracy: 0.9811 - auc: 0.9978 - val_loss: 0.1411 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 937/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1627 - accuracy: 0.9816 - auc: 0.9983 - val_loss: 0.1422 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 938/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1585 - accuracy: 0.9859 - auc: 0.9982 - val_loss: 0.1412 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 939/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1619 - accuracy: 0.9859 - auc: 0.9973 - val_loss: 0.1396 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 940/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1631 - accuracy: 0.9843 - auc: 0.9968 - val_loss: 0.1398 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 941/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1661 - accuracy: 0.9811 - auc: 0.9966 - val_loss: 0.1401 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 942/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1617 - accuracy: 0.9859 - auc: 0.9968 - val_loss: 0.1402 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 943/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1626 - accuracy: 0.9843 - auc: 0.9968 - val_loss: 0.1398 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 944/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1583 - accuracy: 0.9854 - auc: 0.9981 - val_loss: 0.1385 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 945/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1618 - accuracy: 0.9805 - auc: 0.9969 - val_loss: 0.1400 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 946/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1672 - accuracy: 0.9827 - auc: 0.9965 - val_loss: 0.1380 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 947/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1661 - accuracy:

0.9811 - auc: 0.9965 - val_loss: 0.1391 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 948/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1655 - accuracy: 0.9816 - auc: 0.9970 - val_loss: 0.1373 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 949/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1628 - accuracy: 0.9816 - auc: 0.9972 - val_loss: 0.1387 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 950/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1590 - accuracy: 0.9800 - auc: 0.9980 - val_loss: 0.1394 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 951/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1607 - accuracy: 0.9848 - auc: 0.9969 - val_loss: 0.1386 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 952/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1560 - accuracy: 0.9865 - auc: 0.9981 - val_loss: 0.1378 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 953/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1593 - accuracy: 0.9843 - auc: 0.9979 - val_loss: 0.1395 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 954/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1594 - accuracy: 0.9859 - auc: 0.9974 - val_loss: 0.1436 - val_accuracy: 0.9949 - val_auc: 0.9998
Epoch 955/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1664 - accuracy: 0.9789 - auc: 0.9974 - val_loss: 0.1363 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 956/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1576 - accuracy: 0.9843 - auc: 0.9985 - val_loss: 0.1388 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 957/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1589 - accuracy: 0.9838 - auc: 0.9983 - val_loss: 0.1369 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 958/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1636 - accuracy: 0.9843 - auc: 0.9967 - val_loss: 0.1414 - val_accuracy: 0.9949 - val_auc: 0.9999
Epoch 959/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1599 - accuracy: 0.9838 - auc: 0.9974 - val_loss: 0.1387 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 960/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1741 - accuracy: 0.9740 - auc: 0.9958 - val_loss: 0.1373 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 961/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1658 - accuracy: 0.9805 - auc: 0.9959 - val_loss: 0.1359 - val_accuracy: 1.0000 - val_auc: 1.0000
Epoch 962/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1590 - accuracy: 0.9821 - auc: 0.9981 - val_loss: 0.1338 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 963/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1607 - accuracy:

0.9843 - auc: 0.9967 - val_loss: 0.1342 - val_accuracy: 1.0000 - val_auc: 1.0000
Epoch 964/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1574 - accuracy: 0.9827 - auc: 0.9979 - val_loss: 0.1391 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 965/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1546 - accuracy: 0.9848 - auc: 0.9981 - val_loss: 0.1339 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 966/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1623 - accuracy: 0.9789 - auc: 0.9975 - val_loss: 0.1385 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 967/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1648 - accuracy: 0.9800 - auc: 0.9965 - val_loss: 0.1346 - val_accuracy: 1.0000 - val_auc: 1.0000
Epoch 968/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1594 - accuracy: 0.9821 - auc: 0.9973 - val_loss: 0.1352 - val_accuracy: 1.0000 - val_auc: 1.0000
Epoch 969/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1536 - accuracy: 0.9881 - auc: 0.9972 - val_loss: 0.1347 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 970/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1631 - accuracy: 0.9783 - auc: 0.9975 - val_loss: 0.1372 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 971/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1584 - accuracy: 0.9811 - auc: 0.9973 - val_loss: 0.1370 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 972/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1577 - accuracy: 0.9838 - auc: 0.9973 - val_loss: 0.1337 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 973/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1590 - accuracy: 0.9816 - auc: 0.9972 - val_loss: 0.1336 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 974/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1548 - accuracy: 0.9832 - auc: 0.9979 - val_loss: 0.1362 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 975/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1549 - accuracy: 0.9854 - auc: 0.9975 - val_loss: 0.1340 - val_accuracy: 1.0000 - val_auc: 1.0000
Epoch 976/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1516 - accuracy: 0.9892 - auc: 0.9982 - val_loss: 0.1375 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 977/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1568 - accuracy: 0.9838 - auc: 0.9979 - val_loss: 0.1324 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 978/1000
58/58 [=====] - 1s 20ms/step - loss: 0.1583 - accuracy: 0.9827 - auc: 0.9972 - val_loss: 0.1336 - val_accuracy: 1.0000 - val_auc: 1.0000
Epoch 979/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1631 - accuracy:

```
0.9805 - auc: 0.9964 - val_loss: 0.1352 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 980/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1564 - accuracy: 0.9843 - auc: 0.9973 - val_loss: 0.1353 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 981/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1602 - accuracy: 0.9811 - auc: 0.9970 - val_loss: 0.1323 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 982/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1525 - accuracy: 0.9875 - auc: 0.9976 - val_loss: 0.1316 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 983/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1541 - accuracy: 0.9848 - auc: 0.9980 - val_loss: 0.1330 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 984/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1491 - accuracy: 0.9859 - auc: 0.9983 - val_loss: 0.1346 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 985/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1557 - accuracy: 0.9832 - auc: 0.9979 - val_loss: 0.1347 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 986/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1539 - accuracy: 0.9848 - auc: 0.9980 - val_loss: 0.1326 - val_accuracy: 1.0000 - val_auc: 1.0000
Epoch 987/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1515 - accuracy: 0.9865 - auc: 0.9980 - val_loss: 0.1346 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 988/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1558 - accuracy: 0.9821 - auc: 0.9973 - val_loss: 0.1313 - val_accuracy: 1.0000 - val_auc: 1.0000
Epoch 989/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1555 - accuracy: 0.9832 - auc: 0.9973 - val_loss: 0.1332 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 990/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1540 - accuracy: 0.9848 - auc: 0.9979 - val_loss: 0.1357 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 991/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1505 - accuracy: 0.9848 - auc: 0.9984 - val_loss: 0.1329 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 992/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1519 - accuracy: 0.9838 - auc: 0.9980 - val_loss: 0.1303 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 993/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1543 - accuracy: 0.9827 - auc: 0.9979 - val_loss: 0.1328 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 994/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1578 - accuracy: 0.9816 - auc: 0.9971 - val_loss: 0.1301 - val_accuracy: 0.9949 - val_auc: 1.0000
Epoch 995/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1583 - accuracy:
```

```

0.9811 - auc: 0.9979 - val_loss: 0.1313 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 996/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1508 - accuracy: 0.9838 - auc: 0.9980 - val_loss: 0.1313 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 997/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1518 - accuracy: 0.9875 - auc: 0.9970 - val_loss: 0.1323 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 998/1000
58/58 [=====] - 1s 22ms/step - loss: 0.1506 - accuracy: 0.9838 - auc: 0.9980 - val_loss: 0.1329 - val_accuracy: 0.9975 - val_auc: 0.9999
Epoch 999/1000
58/58 [=====] - 1s 21ms/step - loss: 0.1490 - accuracy: 0.9870 - auc: 0.9981 - val_loss: 0.1287 - val_accuracy: 0.9975 - val_auc: 1.0000
Epoch 1000/1000
58/58 [=====] - 1s 23ms/step - loss: 0.1531 - accuracy: 0.9827 - auc: 0.9978 - val_loss: 0.1357 - val_accuracy: 0.9975 - val_auc: 0.9998
training done

```

```
[ ]: folder = "/content/drive/MyDrive/sCNNEPOCHS1000Cedar(Class_weight)/"
```

```
[ ]: import pickle
```

```
[ ]: with open(folder + 'trainHistoryDict', 'wb') as file_pi:
    pickle.dump(history, file_pi)
```

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). These functions will not be directly callable after loading.

```
[ ]: model.save( folder + 'model.h5')
```

```
[ ]: history = pickle.load(open(folder + 'trainHistoryDict', "rb"))
model = tf.keras.models.load_model(folder + 'model.h5')
```

```
[ ]: if same:
    evaluation = model.evaluate(X_test, Y_test, batch_size=32)
    print("loss =", evaluation[0])
    print("binary accuracy = {}%".format(evaluation[1] * 100))
    print("# iterations =", len(history.history['loss']))
```

```

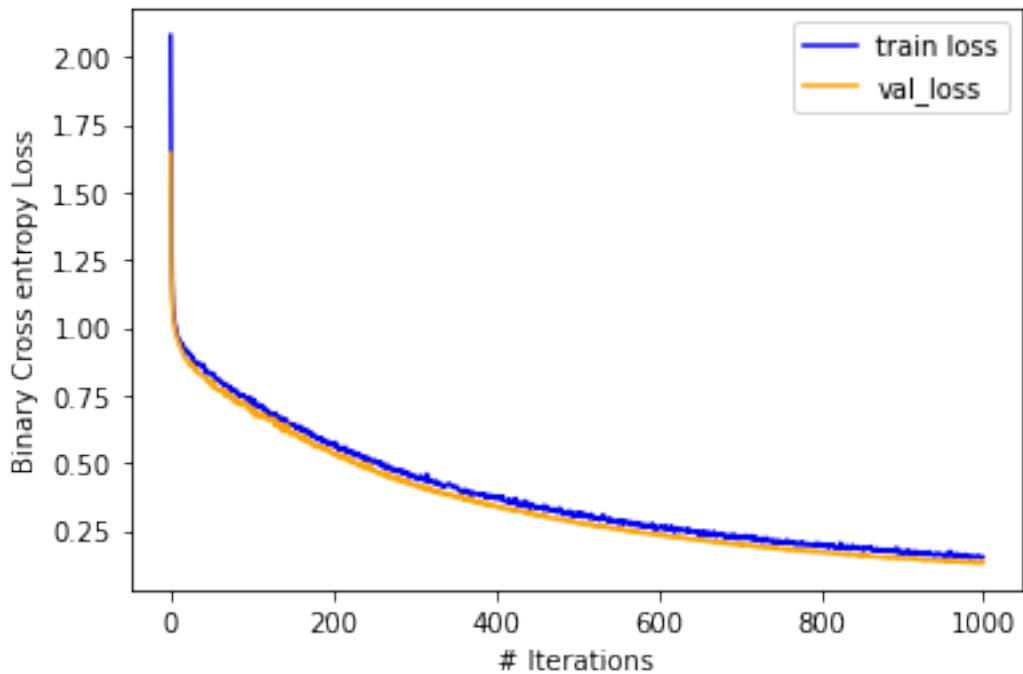
13/13 [=====] - 0s 18ms/step - loss: 0.1740 - accuracy: 0.9824 - auc: 0.9910
loss = 0.17401990294456482
binary accuracy = 98.2367753982544%
# iterations = 1000

```

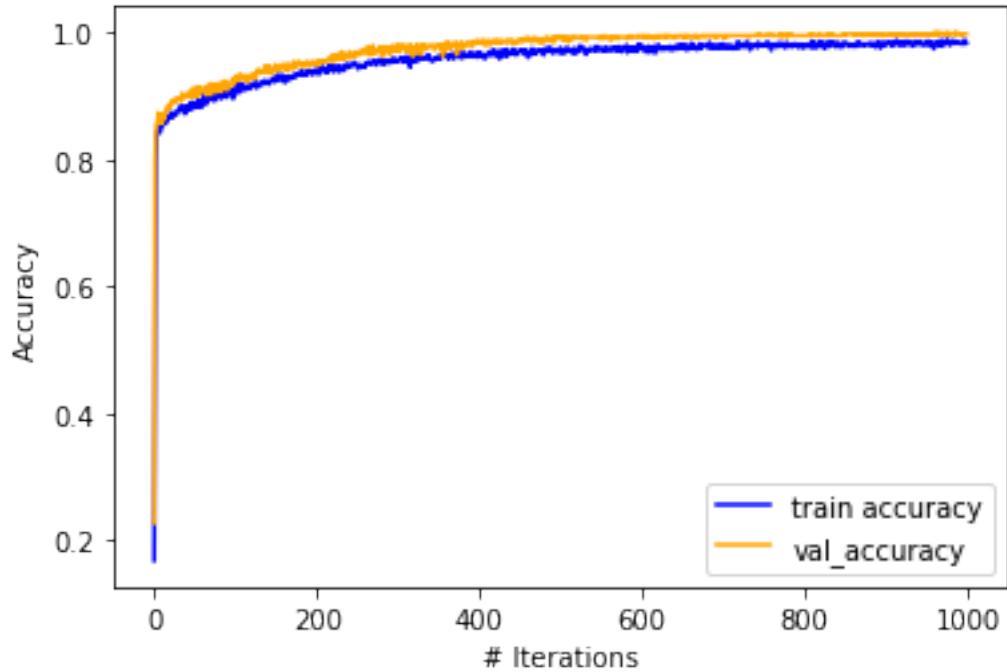
```
[ ]: evaluation
```

```
[ ]: [0.17401990294456482, 0.982367753982544, 0.9910156726837158]
```

```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Binary Cross entropy Loss')
plt.plot(history.history['loss'], color='blue',label = "train loss")
plt.plot(history.history['val_loss'], color='orange',label = "val_loss")
plt.savefig(folder + "loss.jpeg")
plt.legend()
# plt.grid()
plt.show()
```



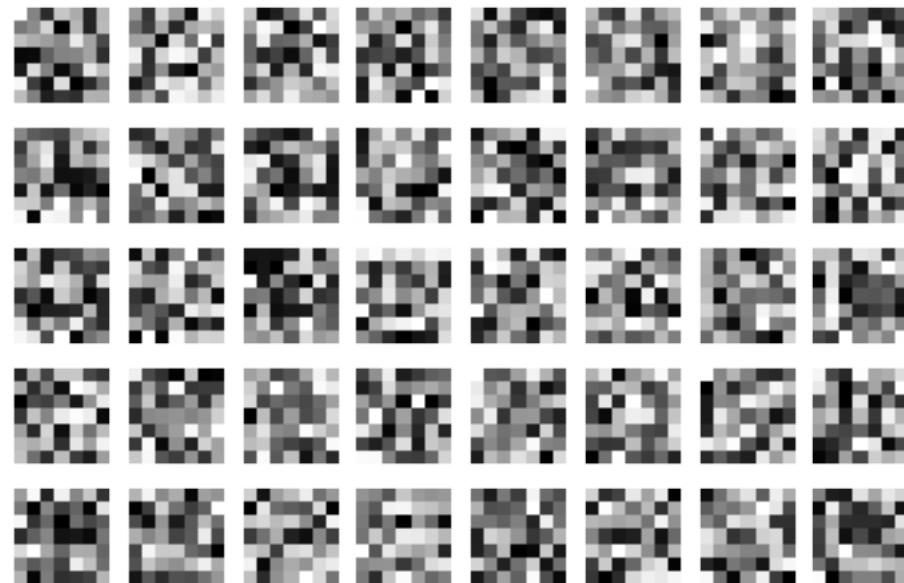
```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Accuracy')
plt.plot(history.history['accuracy'], color='blue',label = "train accuracy")
plt.plot(history.history['val_accuracy'], color='orange',label = "val_accuracy")
# plt.grid()
plt.legend()
plt.savefig(folder + "accuracy.jpeg")
plt.show()
```



```
[ ]: fig, ax = plt.subplots(5,8)

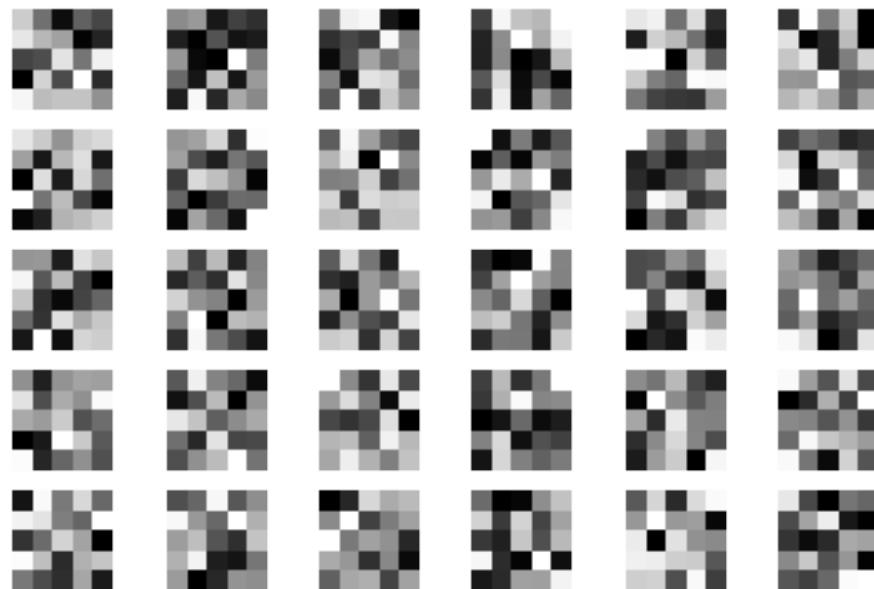
for i in range(40):
    ax[i%5, i//5].imshow(model.layers[0].weights[0] [:,:,0,i].numpy(),cmap='gray')
    ax[i%5, i//5].set_axis_off()
fig.suptitle('Layer 1')
plt.show()
```

Layer 1



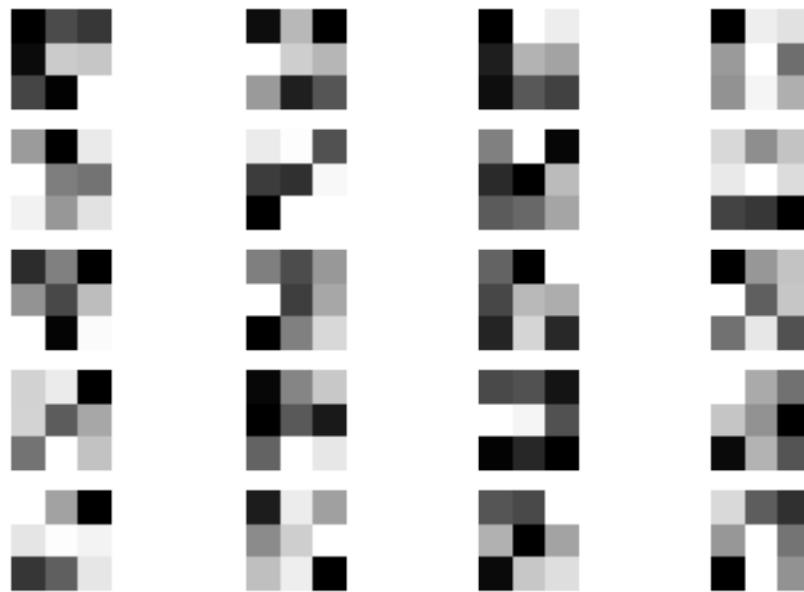
```
[ ]: fig, ax = plt.subplots(5,6, squeeze=False)
for i in range(30):
    ax[i%5, i//5].imshow(model.layers[3].weights[0] [:,:,0,i].numpy(), cmap = 'gray')
    ax[i%5, i//5].set_axis_off()
fig.suptitle('Layer 2')
plt.show()
```

Layer 2



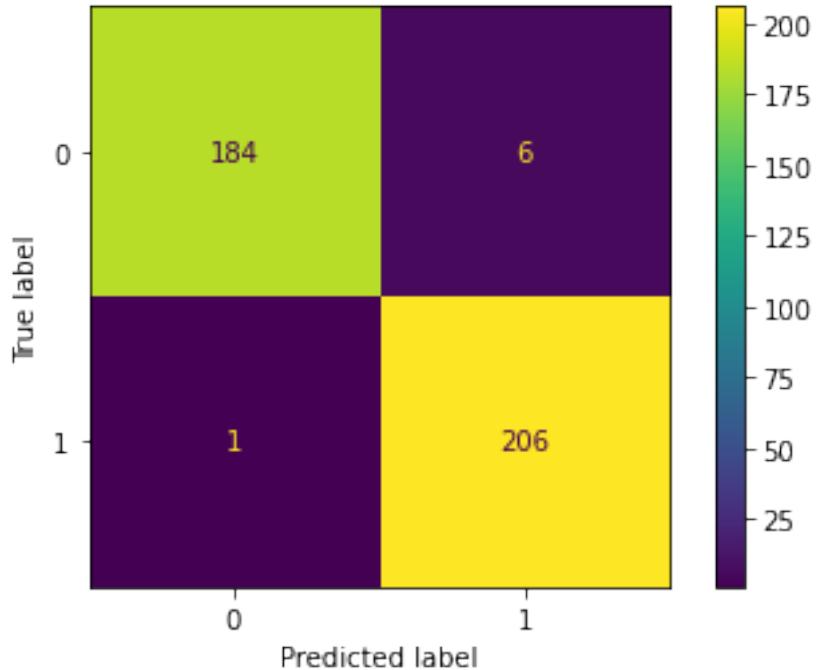
```
[ ]: fig, ax = plt.subplots(5, 4)
for i in range(20):
    ax[i%5, i//5].imshow(model.layers[6].weights[0] [:,:,0,i].numpy(), 'gray')
    ax[i%5, i//5].set_axis_off()
fig.suptitle('Layer 3')
plt.show()
```

Layer 3



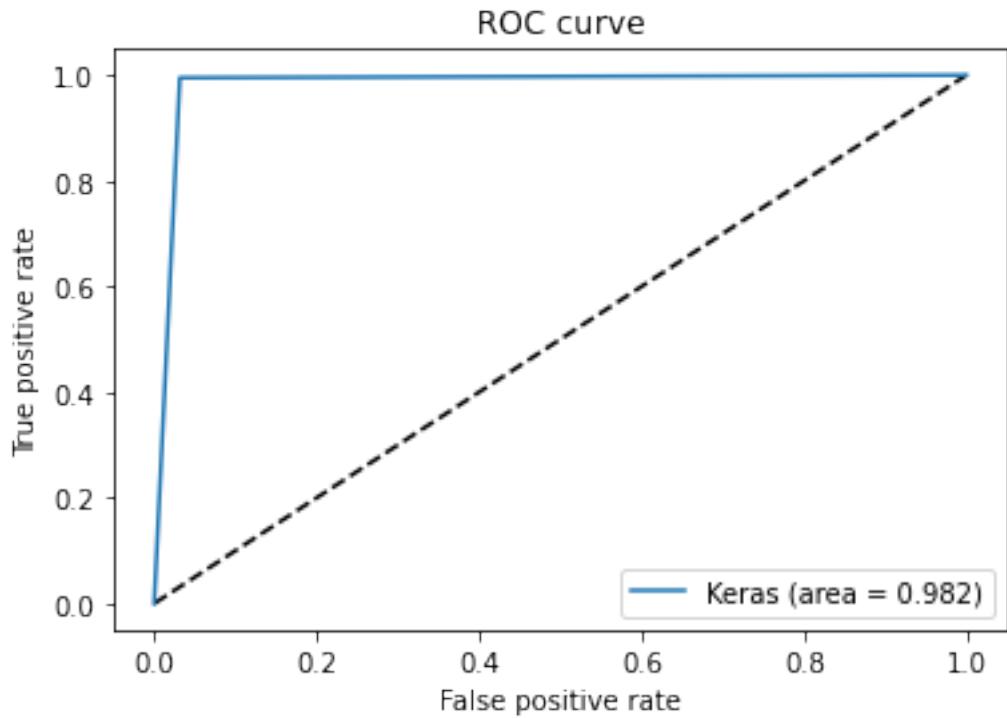
```
[ ]: ConfusionMatrixDisplay(confusion_matrix(tf.argmax(Y_test, axis = 1), tf.  
    ↪argmax(model.predict(X_test), axis = 1))).plot()  
plt.savefig( folder + "confusionmatrix.jpeg")  
plt.show()
```

13/13 [=====] - 0s 7ms/step



```
[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = tf.argmax(model.predict(X_test),axis = 1)
fpr_keras, tpr_keras, thresholds_keras = roc_curve(tf.argmax(Y_test,axis = 1),y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.jpeg")
plt.show()
```

13/13 [=====] - 0s 7ms/step



```
[ ]: model
```

```
[ ]: import pandas as pd
compare_results = pd.DataFrame([model_0_results,
                                 model_1_results,
                                 model_2_results])
compare_results
```

```
[ ]:
```

Model2_SigNetSiamese_BhSig260HindiTraining

November 14, 2024

```
#Model 2 (SigNet)
```

```
[ ]: from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
import os
from tensorflow.keras.initializers import RandomNormal, Constant
import tensorflow as tf
from tensorflow.keras.backend import max, mean, sqrt, square, sum
import seaborn as sns
from tensorflow.keras.optimizers import Adam, RMSprop
from keras.models import Model

from tensorflow import keras
from keras.layers import LeakyReLU, Softmax
from keras.layers import Conv2D, Activation, Input, Dropout, Lambda, Flatten, Dense
from keras.layers import Dense, Flatten, Reshape, Activation
from keras.layers import BatchNormalization, ZeroPadding2D

from keras.layers import MaxPooling2D
from keras.layers import Concatenate
from keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
from keras.layers import Layer, InputSpec
from keras.regularizers import l2
from keras import backend as K
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from tensorflow import one_hot, reshape
```

```
[ ]: import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

Found GPU at: /device:GPU:0

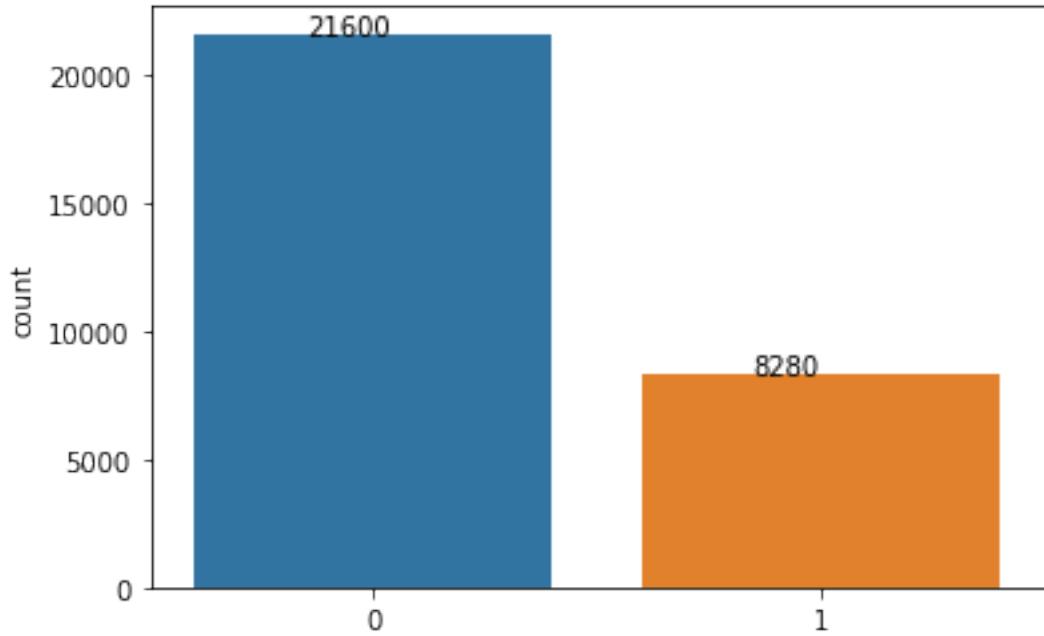
```
[ ]: from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ]: with tf.device('/device:GPU:0'):
    colab = True
    # database = 'bigsig260_224x224_siamese_preprocessed.h5'
    if colab:
        # from google.colab import drive
        # drive.mount('/content/gdrive')
        file = '/content/drive/MyDrive/BHsigHindi224x224_siamese_preprocessed.h5'
    else:
        file = os.path.join(os.getcwd(), 'RoboticLab', database)
    print(file)
    with File(file, 'r') as hdf:
        S1 = np.array(hdf.get('S1'))
        S2 = np.array(hdf.get('S2'))
        Y = np.array(hdf.get('Y'))
    print(S1.shape)
    print(S2.shape)
    print(Y.shape)

/content/drive/MyDrive/BHsigHindi224x224_siamese_preprocessed.h5
(29880, 224, 224, 1)
(29880, 224, 224, 1)
(29880, 1)
```

```
[ ]: ax = sns.countplot(x = Y.reshape(Y.shape[0]))
for p in ax.patches:
    ax.annotate('{:}.'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.
    ↵01))
```



```
[ ]: import sklearn
class_weights = sklearn.utils.class_weight.compute_class_weight(class_weight = "balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict
```

```
[ ]: {0: 0.6916666666666667, 1: 1.8043478260869565}
```

```
[ ]: Y = Y/1.0
```

```
[ ]: with tf.device('/device:GPU:0'):
    seed=randint(10)
    print('seed=' +str(seed))
    indices = permutation(Y.shape[0])
    m = int(0.70 * Y.shape[0])
    n = int(0.15 * Y.shape[0])
    training_id, validation_id, test_id = indices[:m], indices[m: m + n], indices[m+n:]
    S1_train, S1_test, S1_validate = S1[training_id], S1[test_id], S1[validation_id]
    S2_train, S2_test, S2_validate = S2[training_id], S2[test_id], S2[validation_id]
    Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]
    print(S1_train.shape)
    print(S2_train.shape)
```

```

print(Y_train.shape)

def S1,S2,Y

seed=1
(20916, 224, 224, 1)
(20916, 224, 224, 1)
(20916, 1)

[ ]: #One hot Encoding
Y_train = one_hot(Y_train, depth=2)
Y_train = reshape(Y_train, (-1, 2))

Y_test = one_hot(Y_test, depth=2)
Y_test = reshape(Y_test, (-1, 2))

Y_validate = one_hot(Y_validate, depth=2)
Y_validate = reshape(Y_validate, (-1, 2))

[ ]: with tf.device('/device:GPU:0'):
    def SigNet(input_shape):

        seq = keras.Sequential()
        seq.add(Conv2D(96, kernel_size = 11, strides = 1, activation='relu',  

        ↪name='conv1_1', input_shape= input_shape,  

            ↪  

        ↪kernel_initializer='glorot_uniform',data_format='channels_first',padding =  

        ↪"same"))
        seq.add(BatchNormalization(epsilon=1e-04, axis=1, momentum=0.9))
        seq.add(MaxPooling2D(3, strides=2,padding = "same"))

        seq.add(Conv2D(256, kernel_size = 5,strides = 1, activation='relu',  

        ↪name='conv2_1', kernel_initializer='glorot_uniform'  

            ,data_format='channels_first',padding = "same"))
        seq.add(BatchNormalization(epsilon=1e-04 , momentum=0.9))
        seq.add(MaxPooling2D(3, strides=2,padding = "same"))
        seq.add(Dropout(0.3))

        seq.add(Conv2D(384, kernel_size = 3,strides = 3, activation='relu',  

        ↪name='conv3_1', kernel_initializer='glorot_uniform'  

            ,data_format='channels_first',padding = "same"))

        seq.add(Conv2D(256, kernel_size = 3, strides = 3, activation='relu',  

        ↪name='conv3_2', kernel_initializer='glorot_uniform'  

            ,data_format='channels_first',padding = "same"))
        seq.add(MaxPooling2D(3, strides=2,padding = "same"))
        seq.add(Dropout(0.5))

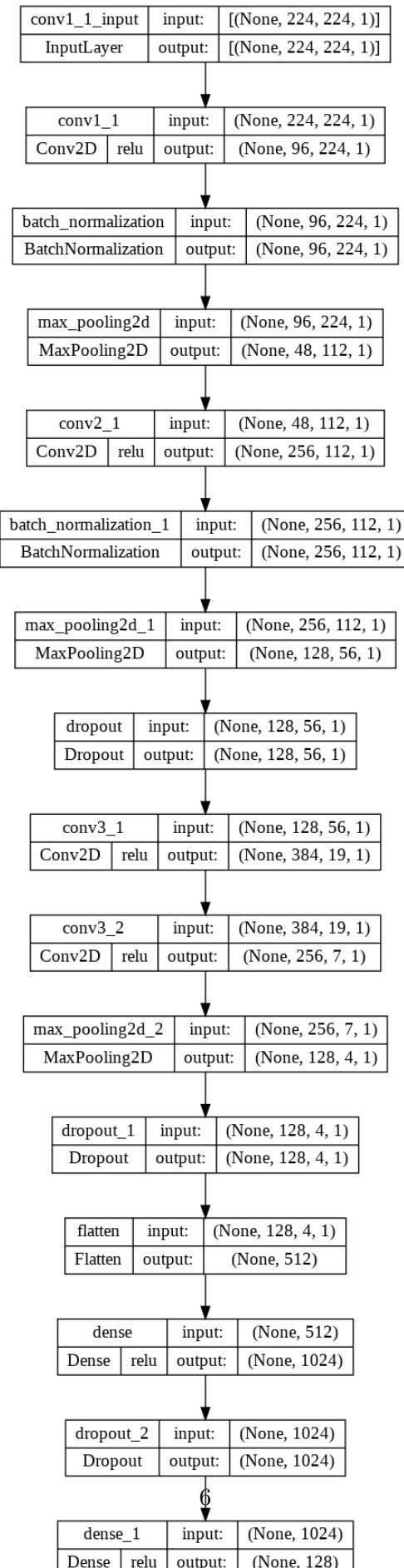
```

```
    seq.add(Flatten(name='flatten'))
    seq.add(Dense(1024 , activation='relu', ↴
    ↪kernel_initializer='glorot_uniform'))
    seq.add(Dropout(0.3))
    seq.add(Dense(128, activation='relu', kernel_initializer='glorot_uniform')) ↴
    ↪# softmax changed to relu

    return seq
```

```
[ ]: plot_model(SigNet(input_shape=(224, 224, 1)), ↴
    ↪show_shapes=True, show_layer_activations = True)
```

```
[ ]:
```



```
[ ]: with tf.device('/device:GPU:0'):
    inputShape = S1_train.shape[1:]
    f = SigNet(inputShape)

[ ]: def euclidean_distance(S):
    X = S['S1']
    Y = S['S2']
    return K.sqrt(K.maximum(K.sum(K.square(X - Y), axis=1, keepdims=True), K.
                           epsilon()))

[ ]: with tf.device('/device:GPU:0'):
    input_a = Input(shape=inputShape, name = 'S1')
    input_b = Input(shape=inputShape, name = 'S2')
    encoded_s1 = f(input_a)
    encoded_s2 = f(input_b)
    distance = Lambda(euclidean_distance)({'S1': encoded_s1, 'S2': encoded_s2})
    model = Model(inputs = [input_a, input_b], outputs = distance)

[ ]: def constructive_loss(y_true, y_pred):
    margin = 1
    y_true = tf.cast(y_true, y_pred.dtype)
    squaredPreds = K.square(y_pred)
    squaredMargin = K.square(K.maximum(margin - y_pred, 0))
    loss = K.mean(y_true * squaredMargin + (1 - y_true) * squaredPreds)
    return loss
```

0.1 HYPER PARAMETER TUNING

0.2 BATCH SIZE TUNING

```
[ ]: # with tf.device('/device:GPU:0'):
#   for i in np.arange(0.0010,0.0101,0.0010):
#     adam = Adam(learning_rate = 0.001,epsilon = 1e-08)
#     model.compile(loss = constructive_loss, optimizer = adam,metrics=['BinaryAccuracy'])
#     print("For Learning Rate = {LR} ".format(LR = i))

[ ]: with tf.device('/device:GPU:0'):
    adam = Adam(learning_rate = 0.001,epsilon = 1e-08)
    model.compile(loss = constructive_loss, optimizer = adam,metrics=['BinaryAccuracy'])

[ ]: with tf.device('/device:GPU:0'):
    batch_Size = 8
    print("For batch Size : ",batch_Size)
```

```

model.fit(x = [S1_train, S2_train],y = Y_train,
           validation_data = [
             ([S1_validate,S2_validate],Y_validate),
               epochs = 1,batch_size = batch_Size,,class_weight = [
                 class_weight_dict,shuffle = True
               ]
)

```

For batch Size : 8
2802/2802 [=====] - 213s 72ms/step - loss: 0.0934 -
binary_accuracy: 0.9104 - val_loss: 0.0813 - val_binary_accuracy: 0.9186

```
[ ]: with tf.device('/device:GPU:0'):
    batch_Size = 16
    print("For batch Size : ",batch_Size)
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = [
                 ([S1_validate,S2_validate],Y_validate),
                   epochs = 1,batch_size = batch_Size,class_weight = [
                     class_weight_dict,shuffle = True
                   ]
)

```

For batch Size : 16
1401/1401 [=====] - 160s 107ms/step - loss: 0.0904 -
binary_accuracy: 0.9103 - val_loss: 0.0883 - val_binary_accuracy: 0.9116

```
[ ]: with tf.device('/device:GPU:0'):
    batch_Size = 32
    print("For batch Size : ",batch_Size)
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = [
                 ([S1_validate,S2_validate],Y_validate),
                   epochs = 1,batch_size = batch_Size,class_weight = [
                     class_weight_dict,shuffle = True
                   ]
)

```

For batch Size : 32
701/701 [=====] - 132s 172ms/step - loss: 0.0979 -
binary_accuracy: 0.9093 - val_loss: 0.0795 - val_binary_accuracy: 0.9205

```
[ ]: with tf.device('/device:GPU:0'):
    batch_Size = 64
    print("For batch Size : ",batch_Size)
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = [
                 ([S1_validate,S2_validate],Y_validate),
                   epochs = 1,batch_size = batch_Size,class_weight = [
                     class_weight_dict,shuffle = True
                   ]
)

```

```
For batch Size : 64
351/351 [=====] - 120s 311ms/step - loss: 0.0908 -
binary_accuracy: 0.9108 - val_loss: 0.0768 - val_binary_accuracy: 0.9232
```

```
[ ]: with tf.device('/device:GPU:0'):
    batch_Size = 128
    print("For batch Size : ",batch_Size)
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = [
                   ([S1_validate,S2_validate],Y_validate),
                   epochs = 1,batch_size = batch_Size,class_weight = [
                       class_weight_dict,shuffle = True
                   ]
               )
```

```
For batch Size : 128
176/176 [=====] - 118s 605ms/step - loss: 0.0928 -
binary_accuracy: 0.9084 - val_loss: 0.0779 - val_binary_accuracy: 0.9221
```

THE BEST ACCURACY IS FOR BATCH SIZE = 64

0.3 LEARNING RATE

```
[ ]: batch_Size = 32

[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-04
    print("For learning Rate = ",lr)
    SGD = tf.keras.optimizers.SGD(learning_rate = lr,momentum = 0.9)
    model.compile(loss = constructive_loss, optimizer = SGD,
                   metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = [
                   ([S1_validate,S2_validate],Y_validate),
                   epochs = 5,batch_size = batch_Size,class_weight = [
                       class_weight_dict,shuffle = True
                   ]
               )
```

```
For learning Rate =  0.0001
Epoch 1/5
229/654 [=====>...] - ETA: 33s - loss: 0.2544 -
binary_accuracy: 0.5000
```

```
-----
KeyboardInterrupt                                     Traceback (most recent call last)
<ipython-input-21-ee42594bbc27> in <module>
      6     model.fit(x = [S1_train, S2_train],y = Y_train,
      7                 validation_data = [
      8                   ([S1_validate,S2_validate],Y_validate),
-----> 8                   epochs = 5,batch_size = batch_Size,shuffle = True
```

```

9
)

/usr/local/lib/python3.7/dist-packages/keras/utils/traceback_utils.py in
↳ error_handler(*args, **kwargs)
62     filtered_tb = None
63     try:
--> 64         return fn(*args, **kwargs)
65     except Exception as e: # pylint: disable=broad-except
66         filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.7/dist-packages/keras/engine/training.py in fit(self, x,
↳ y, batch_size, epochs, verbose, callbacks, validation_split, validation_data,
↳ shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch, u
↳ validation_steps, validation_batch_size, validation_freq, max_queue_size, u
↳ workers, use_multiprocessing)
1407             _r=1):
1408                 callbacks.on_train_batch_begin(step)
-> 1409                 tmp_logs = self.train_function(iterator)
1410                 if data_handler.should_sync:
1411                     context.async_wait()

/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/traceback_utils.p
↳ in error_handler(*args, **kwargs)
148     filtered_tb = None
149     try:
--> 150         return fn(*args, **kwargs)
151     except Exception as e:
152         filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/def_function.py
↳ in __call__(self, *args, **kwds)
913
914     with OptionalXlaContext(self._jit_compile):
--> 915         result = self._call(*args, **kwds)
916
917     new_tracing_count = self.experimental_get_tracing_count()

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/def_function.py
↳ in _call(self, *args, **kwds)
945     # In this case we have created variables on the first call, so we
↳ run the
946     # defunned version which is guaranteed to never create variables.
--> 947     return self._stateless_fn(*args, **kwds) # pylint:u
↳ disable=not-callable
948     elif self._stateful_fn is not None:
949         # Release the lock early so that multiple threads can perform the
↳ call

```

```

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/function.py in __call__(self, *args, **kwargs)
 2452         filtered_flat_args) = self._maybe_define_function(args, kwargs)
 2453     return graph_function._call_flat(
-> 2454         filtered_flat_args, captured_inputs=graph_function.
 2455         _captured_inputs) # pylint: disable=protected-access
 2455
 2456     @property

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/function.py in __call_flat(self, args, captured_inputs, cancellation_manager)
 1859         # No tape is watching; skip to running the function.
 1860         return self._build_call_outputs(self._inference_function.call(
-> 1861             ctx, args, cancellation_manager=cancellation_manager))
 1862         forward_backward = self._select_forward_and_backward_functions(
 1863             args,

```

```

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/function.py in call(self, ctx, args, cancellation_manager)
 500             inputs=args,
 501             attrs=attrs,
-> 502             ctx=ctx)
 503         else:
 504             outputs = execute.execute_with_cancellation(

```

```

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
 53     ctx.ensure_initialized()
 54     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,
-> 55             op_name,
---> 55                                         inputs, attrs, num_outputs)
 56     except core._NotOkStatusException as e:
 57         if name is not None:

```

KeyboardInterrupt:

```

[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-03
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
-> metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = [
-> ([S1_validate,S2_validate],Y_validate),

```

```
        epochs = 5,batch_size = batch_Size,class_weight = None
    ↵class_weight_dict,shuffle = True
    )
```

```
For learning Rate = 0.001
Epoch 1/5
654/654 [=====] - 36s 39ms/step - loss: 0.2937 -
binary_accuracy: 0.7216 - val_loss: 0.2698 - val_binary_accuracy: 0.7300
Epoch 2/5
654/654 [=====] - 24s 37ms/step - loss: 0.2827 -
binary_accuracy: 0.7234 - val_loss: 0.2698 - val_binary_accuracy: 0.7300
Epoch 3/5
654/654 [=====] - 25s 38ms/step - loss: 0.2834 -
binary_accuracy: 0.7234 - val_loss: 0.2698 - val_binary_accuracy: 0.7300
Epoch 4/5
654/654 [=====] - 24s 37ms/step - loss: 0.2838 -
binary_accuracy: 0.7234 - val_loss: 0.2698 - val_binary_accuracy: 0.7300
Epoch 5/5
654/654 [=====] - 24s 37ms/step - loss: 0.2837 -
binary_accuracy: 0.7234 - val_loss: 0.2698 - val_binary_accuracy: 0.7300
```

```
[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-04
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
    ↵metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = None
    ↵([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight = None
    ↵class_weight_dict,shuffle = True
    )
```

```
For learning Rate = 0.0001
Epoch 1/5
654/654 [=====] - 36s 39ms/step - loss: 0.2232 -
binary_accuracy: 0.7001 - val_loss: 0.2445 - val_binary_accuracy: 0.7129
Epoch 2/5
654/654 [=====] - 25s 38ms/step - loss: 0.2047 -
binary_accuracy: 0.7146 - val_loss: 0.2358 - val_binary_accuracy: 0.7140
Epoch 3/5
654/654 [=====] - 25s 38ms/step - loss: 0.2008 -
binary_accuracy: 0.7189 - val_loss: 0.2318 - val_binary_accuracy: 0.7115
Epoch 4/5
654/654 [=====] - 24s 37ms/step - loss: 0.1992 -
binary_accuracy: 0.7188 - val_loss: 0.2352 - val_binary_accuracy: 0.7146
Epoch 5/5
```

```
654/654 [=====] - 24s 37ms/step - loss: 0.1935 -  
binary_accuracy: 0.7259 - val_loss: 0.2279 - val_binary_accuracy: 0.7149
```

```
[ ]: with tf.device('/device:GPU:0'):  
    lr = 1e-05  
    print("For learning Rate = ",lr)  
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)  
    model.compile(loss = constructive_loss, optimizer = rms,  
    ↵metrics=['BinaryAccuracy'])  
    model.fit(x = [S1_train, S2_train],y = Y_train,  
              validation_data =  
    ↵([S1_validate,S2_validate],Y_validate),  
              epochs = 5,batch_size = batch_Size,class_weight =  
    ↵class_weight_dict,shuffle = True  
    )
```

```
For learning Rate =  1e-05  
Epoch 1/5  
654/654 [=====] - 33s 35ms/step - loss: 0.2488 -  
binary_accuracy: 0.7070 - val_loss: 0.2451 - val_binary_accuracy: 0.7135  
Epoch 2/5  
654/654 [=====] - 22s 34ms/step - loss: 0.1971 -  
binary_accuracy: 0.7212 - val_loss: 0.2483 - val_binary_accuracy: 0.7135  
Epoch 3/5  
654/654 [=====] - 23s 35ms/step - loss: 0.1953 -  
binary_accuracy: 0.7240 - val_loss: 0.2338 - val_binary_accuracy: 0.7135  
Epoch 4/5  
654/654 [=====] - 22s 34ms/step - loss: 0.1947 -  
binary_accuracy: 0.7232 - val_loss: 0.2389 - val_binary_accuracy: 0.7124  
Epoch 5/5  
654/654 [=====] - 22s 34ms/step - loss: 0.1917 -  
binary_accuracy: 0.7255 - val_loss: 0.2385 - val_binary_accuracy: 0.7135
```

```
[ ]: with tf.device('/device:GPU:0'):  
    lr = 1e-06  
    print("For learning Rate = ",lr)  
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)  
    model.compile(loss = constructive_loss, optimizer = rms,  
    ↵metrics=['BinaryAccuracy'])  
    model.fit(x = [S1_train, S2_train],y = Y_train,  
              validation_data =  
    ↵([S1_validate,S2_validate],Y_validate),  
              epochs = 5,batch_size = batch_Size,class_weight =  
    ↵class_weight_dict,shuffle = True  
    )
```

```
For learning Rate =  1e-06  
Epoch 1/5
```

```

654/654 [=====] - 26s 36ms/step - loss: 0.2219 -
binary_accuracy: 0.6782 - val_loss: 0.2373 - val_binary_accuracy: 0.7160
Epoch 2/5
654/654 [=====] - 22s 34ms/step - loss: 0.1969 -
binary_accuracy: 0.7241 - val_loss: 0.2315 - val_binary_accuracy: 0.7160
Epoch 3/5
654/654 [=====] - 22s 34ms/step - loss: 0.1910 -
binary_accuracy: 0.7257 - val_loss: 0.2267 - val_binary_accuracy: 0.7160
Epoch 4/5
654/654 [=====] - 25s 39ms/step - loss: 0.1865 -
binary_accuracy: 0.7283 - val_loss: 0.2198 - val_binary_accuracy: 0.7160
Epoch 5/5
202/654 [=====>...] - ETA: 14s - loss: 0.1819 -
binary_accuracy: 0.7333

```

```

-----
KeyboardInterrupt                                     Traceback (most recent call last)
<ipython-input-18-447822a95911> in <module>
      6     model.fit(x = [S1_train, S2_train],y = Y_train,
      7                     validation_data =_
<__main__.NumpyArrayDataset object at 0x7f3a2d1a1a0>,
-> 8                         epochs = 5,batch_size = batch_Size,class_weight=
<__main__.NumpyArrayDataset object at 0x7f3a2d1a1a0>,
      9                     shuffle = True

```

```

/usr/local/lib/python3.7/dist-packages/keras/utils/traceback_utils.py in_
<error_handler(*args, **kwargs)
      62     filtered_tb = None
      63     try:
-> 64         return fn(*args, **kwargs)
      65     except Exception as e: # pylint: disable=broad-except
      66         filtered_tb = _process_traceback_frames(e.__traceback__)

```

```

/usr/local/lib/python3.7/dist-packages/keras/engine/training.py in fit(self, x,
      67     validation_split=0.0, validation_steps=None, validation_freq=1,
      68     workers=1, use_multiprocessing=False):
-> 69         logs = self._log_fn(step, metrics)
      70         if self.stop_training:
      71             break

```

```

/usr/local/lib/python3.7/dist-packages/keras/callbacks.py in_
<on_train_batch_end(self, batch, logs)
      436     """
-> 437     if self._should_call_train_batch_hooks:

```

```

--> 438         self._call_batch_hook(ModeKeys.TRAIN, 'end', batch, logs)
439
440     def on_test_batch_begin(self, batch, logs=None):
441
442         /usr/local/lib/python3.7/dist-packages/keras/callbacks.py in _call_batch_hook(self, mode, hook, batch, logs)
443             self._call_batch_begin_hook(mode, batch, logs)
444         elif hook == 'end':
--> 445             self._call_batch_end_hook(mode, batch, logs)
446         else:
447             raise ValueError(
448
449         /usr/local/lib/python3.7/dist-packages/keras/callbacks.py in _call_batch_end_hook(self, mode, batch, logs)
450             self._batch_times.append(batch_time)
451
452         /usr/local/lib/python3.7/dist-packages/keras/callbacks.py in _call_batch_hook_helper(self, hook_name, batch, logs)
453             self._call_batch_hook_helper(self, hook_name, batch, logs)
454             for callback in self.callbacks:
455                 hook = getattr(callback, hook_name)
--> 456             hook(batch, logs)
457
458         if self._check_timing:
459
460         /usr/local/lib/python3.7/dist-packages/keras/callbacks.py in on_train_batch_end(self, batch, logs)
461             self._batch_update_progbar(batch, logs)
462
463     def on_train_batch_end(self, batch, logs=None):
464
465         /usr/local/lib/python3.7/dist-packages/keras/callbacks.py in _batch_update_progbar(self, batch, logs)
466             if self.verbose == 1:
467                 # Only block async when verbose = 1.
468             logs = tf_utils.sync_to_numpy_or_python_type(logs)
--> 469             self.progbar.update(self.seen, list(logs.items()), finalize=False)
470
471         1104     if self.verbose == 1:
472             # Only block async when verbose = 1.
473             logs = tf_utils.sync_to_numpy_or_python_type(logs)
474             self.progbar.update(self.seen, list(logs.items()), finalize=False)
475
476     /usr/local/lib/python3.7/dist-packages/keras/utils/tf_utils.py in sync_to_numpy_or_python_type(tensors)
477             return t.item() if np.ndim(t) == 0 else t
478
479         605     return t.item() if np.ndim(t) == 0 else t
480

```

```

--> 607    return tf.nest.map_structure(_to_single_numpy_or_python_type, tensors
 608
 609

/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/nest.py in
`map_structure(func, *structure, **kwargs)
 914
 915    return pack_sequence_as(
--> 916        structure[0], [func(*x) for x in entries],
 917        expand_composites=expand_composites)
 918

/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/nest.py in
`<listcomp>(.0)
 914
 915    return pack_sequence_as(
--> 916        structure[0], [func(*x) for x in entries],
 917        expand_composites=expand_composites)
 918

/usr/local/lib/python3.7/dist-packages/keras/utils/tf_utils.py in
`_to_single_numpy_or_python_type(t)
 599    # Don't turn ragged or sparse tensors to NumPy.
 600    if isinstance(t, tf.Tensor):
--> 601        t = t.numpy()
 602    # Strings, ragged and sparse tensors don't have .item(). Return them
`as-is.
 603    if not isinstance(t, (np.ndarray, np.generic)):

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/ops.py in
`numpy(self)
 1157    """
 1158    # TODO(slebedev): Consider avoiding a copy for non-CPU or remote
`tensors.
-> 1159    maybe_arr = self._numpy() # pylint: disable=protected-access
 1160    return maybe_arr.copy() if isinstance(maybe_arr, np.ndarray) else
`maybe_arr
 1161

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/ops.py in
`_numpy(self)
 1123    def _numpy(self):
 1124        try:
-> 1125            return self._numpy_internal()
 1126        except core._NotOkStatusException as e: # pylint: disable=
`disable=protected-access
 1127            raise core._status_to_exception(e) from None # pylint: disable=
`disable=protected-access

```

KeyboardInterrupt:

```
[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-07
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
    ↪metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = ↪
    ↪([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight = ↪
    ↪class_weight_dict,shuffle = True
               )
```

For learning Rate = 1e-07
Epoch 1/5
654/654 [=====] - 32s 37ms/step - loss: 2.3212 -
binary_accuracy: 0.2760 - val_loss: 0.2162 - val_binary_accuracy: 0.7115
Epoch 2/5
654/654 [=====] - 23s 34ms/step - loss: 0.4720 -
binary_accuracy: 0.3013 - val_loss: 0.2243 - val_binary_accuracy: 0.7135
Epoch 3/5
654/654 [=====] - 22s 34ms/step - loss: 0.2376 -
binary_accuracy: 0.6079 - val_loss: 0.2404 - val_binary_accuracy: 0.7135
Epoch 4/5
654/654 [=====] - 22s 34ms/step - loss: 0.2098 -
binary_accuracy: 0.7163 - val_loss: 0.2480 - val_binary_accuracy: 0.7135
Epoch 5/5
654/654 [=====] - 22s 34ms/step - loss: 0.2044 -
binary_accuracy: 0.7230 - val_loss: 0.2509 - val_binary_accuracy: 0.7135

```
[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-08
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
    ↪metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = ↪
    ↪([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight = ↪
    ↪class_weight_dict,shuffle = True
               )
```

For learning Rate = 1e-08

```

Epoch 1/5
654/654 [=====] - 29s 42ms/step - loss: 0.1911 -
binary_accuracy: 0.7238 - val_loss: 0.2165 - val_binary_accuracy: 0.7320
Epoch 2/5
654/654 [=====] - 25s 38ms/step - loss: 0.1907 -
binary_accuracy: 0.7241 - val_loss: 0.2162 - val_binary_accuracy: 0.7320
Epoch 3/5
654/654 [=====] - 25s 38ms/step - loss: 0.1910 -
binary_accuracy: 0.7238 - val_loss: 0.2154 - val_binary_accuracy: 0.7320
Epoch 4/5
654/654 [=====] - 25s 38ms/step - loss: 0.1914 -
binary_accuracy: 0.7238 - val_loss: 0.2154 - val_binary_accuracy: 0.7320
Epoch 5/5
654/654 [=====] - 25s 38ms/step - loss: 0.1908 -
binary_accuracy: 0.7241 - val_loss: 0.2160 - val_binary_accuracy: 0.7320

```

0.4 FINAL TRAINING

BEST LEARNING RATE IS 1e-04

```

[ ]: Batch_size = 32
lr = 1e-04
Epochs = 60

[ ]: with tf.device('/device:GPU:0'):
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,metrics=['BinaryAccuracy','AUC'])

[ ]: with tf.device('/device:GPU:0'):
    callbacks = [
        EarlyStopping(patience = 6, verbose = 1),
        ModelCheckpoint('/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/
        bhsig260-{epoch:03d}.h5'
        , verbose=1,save_best_only = True)
    ]

[ ]: with tf.device('/device:GPU:0'):
    results = model.fit(x = [S1_train, S2_train],y = Y_train,
                         validation_data = ([S1_validate,S2_validate],Y_validate),
                         epochs = Epochs,callbacks = callbacks,batch_size = Batch_size
                         ,class_weight = class_weight_dict,shuffle = True
                         )

```

```

Epoch 1/60
654/654 [=====] - ETA: 0s - loss: 0.2307 -
binary_accuracy: 0.6978

```

```
Epoch 1: val_loss improved from inf to 0.23711, saving model to
/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/bhsig260-001.h5
654/654 [=====] - 34s 39ms/step - loss: 0.2307 -
binary_accuracy: 0.6978 - val_loss: 0.2371 - val_binary_accuracy: 0.7164
Epoch 2/60
654/654 [=====] - ETA: 0s - loss: 0.2083 -
binary_accuracy: 0.7125
Epoch 2: val_loss did not improve from 0.23711
654/654 [=====] - 25s 38ms/step - loss: 0.2083 -
binary_accuracy: 0.7125 - val_loss: 0.2376 - val_binary_accuracy: 0.7153
Epoch 3/60
654/654 [=====] - ETA: 0s - loss: 0.2024 -
binary_accuracy: 0.7154
Epoch 3: val_loss improved from 0.23711 to 0.23053, saving model to
/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/bhsig260-003.h5
654/654 [=====] - 25s 39ms/step - loss: 0.2024 -
binary_accuracy: 0.7154 - val_loss: 0.2305 - val_binary_accuracy: 0.7166
Epoch 4/60
653/654 [=====>.] - ETA: 0s - loss: 0.1986 -
binary_accuracy: 0.7197
Epoch 4: val_loss did not improve from 0.23053
654/654 [=====] - 25s 39ms/step - loss: 0.1986 -
binary_accuracy: 0.7197 - val_loss: 0.2324 - val_binary_accuracy: 0.7164
Epoch 5/60
653/654 [=====>.] - ETA: 0s - loss: 0.1951 -
binary_accuracy: 0.7243
Epoch 5: val_loss did not improve from 0.23053
654/654 [=====] - 25s 39ms/step - loss: 0.1951 -
binary_accuracy: 0.7243 - val_loss: 0.2333 - val_binary_accuracy: 0.7162
Epoch 6/60
653/654 [=====>.] - ETA: 0s - loss: 0.1919 -
binary_accuracy: 0.7284
Epoch 6: val_loss did not improve from 0.23053
654/654 [=====] - 25s 38ms/step - loss: 0.1919 -
binary_accuracy: 0.7283 - val_loss: 0.2356 - val_binary_accuracy: 0.7164
Epoch 7/60
654/654 [=====] - ETA: 0s - loss: 0.1941 -
binary_accuracy: 0.7233
Epoch 7: val_loss did not improve from 0.23053
654/654 [=====] - 25s 39ms/step - loss: 0.1941 -
binary_accuracy: 0.7233 - val_loss: 0.2318 - val_binary_accuracy: 0.7126
Epoch 8/60
653/654 [=====>.] - ETA: 0s - loss: 0.1921 -
binary_accuracy: 0.7256
Epoch 8: val_loss improved from 0.23053 to 0.21936, saving model to
/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/bhsig260-008.h5
654/654 [=====] - 26s 39ms/step - loss: 0.1921 -
binary_accuracy: 0.7255 - val_loss: 0.2194 - val_binary_accuracy: 0.7166
```

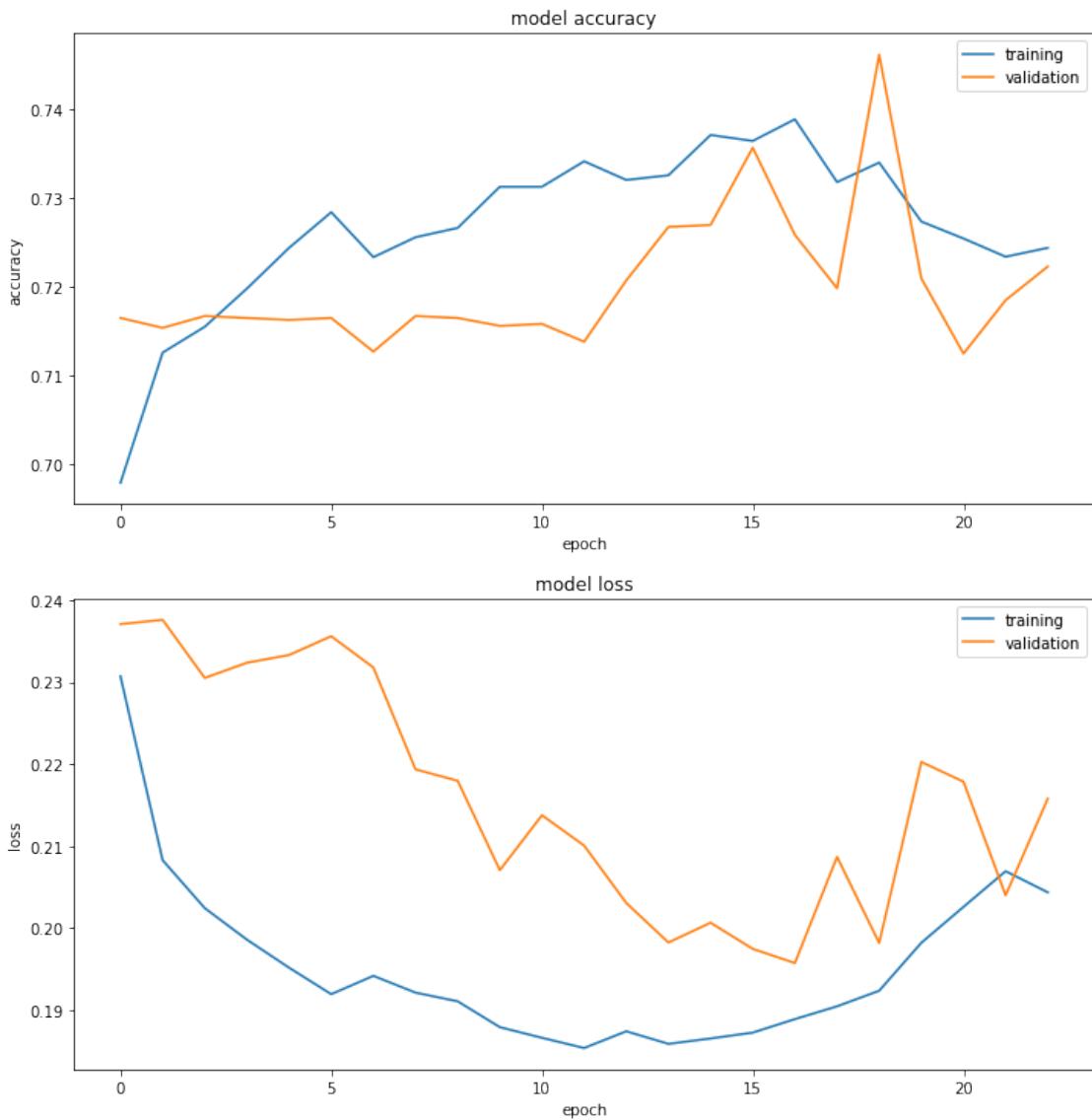
```
Epoch 9/60
653/654 [=====>.] - ETA: 0s - loss: 0.1910 -
binary_accuracy: 0.7266
Epoch 9: val_loss improved from 0.21936 to 0.21797, saving model to
/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/bhsig260-009.h5
654/654 [=====] - 25s 39ms/step - loss: 0.1910 -
binary_accuracy: 0.7266 - val_loss: 0.2180 - val_binary_accuracy: 0.7164
Epoch 10/60
653/654 [=====>.] - ETA: 0s - loss: 0.1879 -
binary_accuracy: 0.7311
Epoch 10: val_loss improved from 0.21797 to 0.20705, saving model to
/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/bhsig260-010.h5
654/654 [=====] - 25s 39ms/step - loss: 0.1879 -
binary_accuracy: 0.7312 - val_loss: 0.2070 - val_binary_accuracy: 0.7155
Epoch 11/60
653/654 [=====>.] - ETA: 0s - loss: 0.1865 -
binary_accuracy: 0.7312
Epoch 11: val_loss did not improve from 0.20705
654/654 [=====] - 25s 39ms/step - loss: 0.1866 -
binary_accuracy: 0.7312 - val_loss: 0.2138 - val_binary_accuracy: 0.7158
Epoch 12/60
653/654 [=====>.] - ETA: 0s - loss: 0.1853 -
binary_accuracy: 0.7341
Epoch 12: val_loss did not improve from 0.20705
654/654 [=====] - 25s 39ms/step - loss: 0.1853 -
binary_accuracy: 0.7341 - val_loss: 0.2101 - val_binary_accuracy: 0.7137
Epoch 13/60
653/654 [=====>.] - ETA: 0s - loss: 0.1873 -
binary_accuracy: 0.7321
Epoch 13: val_loss improved from 0.20705 to 0.20304, saving model to
/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/bhsig260-013.h5
654/654 [=====] - 25s 39ms/step - loss: 0.1874 -
binary_accuracy: 0.7320 - val_loss: 0.2030 - val_binary_accuracy: 0.7207
Epoch 14/60
653/654 [=====>.] - ETA: 0s - loss: 0.1859 -
binary_accuracy: 0.7324
Epoch 14: val_loss improved from 0.20304 to 0.19821, saving model to
/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/bhsig260-014.h5
654/654 [=====] - 25s 39ms/step - loss: 0.1858 -
binary_accuracy: 0.7325 - val_loss: 0.1982 - val_binary_accuracy: 0.7267
Epoch 15/60
653/654 [=====>.] - ETA: 0s - loss: 0.1865 -
binary_accuracy: 0.7369
Epoch 15: val_loss did not improve from 0.19821
654/654 [=====] - 25s 39ms/step - loss: 0.1865 -
binary_accuracy: 0.7370 - val_loss: 0.2006 - val_binary_accuracy: 0.7269
Epoch 16/60
653/654 [=====>.] - ETA: 0s - loss: 0.1872 -
```

```
binary_accuracy: 0.7364
Epoch 16: val_loss improved from 0.19821 to 0.19743, saving model to
/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/bhsig260-016.h5
654/654 [=====] - 25s 39ms/step - loss: 0.1872 -
binary_accuracy: 0.7364 - val_loss: 0.1974 - val_binary_accuracy: 0.7356
Epoch 17/60
653/654 [=====>.] - ETA: 0s - loss: 0.1888 -
binary_accuracy: 0.7388
Epoch 17: val_loss improved from 0.19743 to 0.19570, saving model to
/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/bhsig260-017.h5
654/654 [=====] - 25s 39ms/step - loss: 0.1888 -
binary_accuracy: 0.7388 - val_loss: 0.1957 - val_binary_accuracy: 0.7258
Epoch 18/60
653/654 [=====>.] - ETA: 0s - loss: 0.1904 -
binary_accuracy: 0.7318
Epoch 18: val_loss did not improve from 0.19570
654/654 [=====] - 25s 39ms/step - loss: 0.1904 -
binary_accuracy: 0.7317 - val_loss: 0.2087 - val_binary_accuracy: 0.7198
Epoch 19/60
653/654 [=====>.] - ETA: 0s - loss: 0.1923 -
binary_accuracy: 0.7339
Epoch 19: val_loss did not improve from 0.19570
654/654 [=====] - 26s 39ms/step - loss: 0.1923 -
binary_accuracy: 0.7339 - val_loss: 0.1982 - val_binary_accuracy: 0.7461
Epoch 20/60
653/654 [=====>.] - ETA: 0s - loss: 0.1981 -
binary_accuracy: 0.7274
Epoch 20: val_loss did not improve from 0.19570
654/654 [=====] - 26s 40ms/step - loss: 0.1982 -
binary_accuracy: 0.7273 - val_loss: 0.2203 - val_binary_accuracy: 0.7209
Epoch 21/60
653/654 [=====>.] - ETA: 0s - loss: 0.2026 -
binary_accuracy: 0.7254
Epoch 21: val_loss did not improve from 0.19570
654/654 [=====] - 26s 39ms/step - loss: 0.2026 -
binary_accuracy: 0.7254 - val_loss: 0.2179 - val_binary_accuracy: 0.7124
Epoch 22/60
653/654 [=====>.] - ETA: 0s - loss: 0.2070 -
binary_accuracy: 0.7232
Epoch 22: val_loss did not improve from 0.19570
654/654 [=====] - 26s 40ms/step - loss: 0.2069 -
binary_accuracy: 0.7233 - val_loss: 0.2040 - val_binary_accuracy: 0.7184
Epoch 23/60
653/654 [=====>.] - ETA: 0s - loss: 0.2043 -
binary_accuracy: 0.7243
Epoch 23: val_loss did not improve from 0.19570
654/654 [=====] - 26s 40ms/step - loss: 0.2043 -
binary_accuracy: 0.7243 - val_loss: 0.2158 - val_binary_accuracy: 0.7222
```

```
Epoch 23: early stopping
```

```
[ ]: import pickle
[ ]: with open('/content/drive/MyDrive/HindiSigNet(1e-04)DataEpochs60/
   ↪trainHistoryDict', 'wb') as file_pi:
      # pickle.dump(results.history, file_pi)
[ ]: import pickle
history = pickle.load(open('/content/drive/MyDrive/
   ↪HindiSigNet(1e-04)DataEpochs60/trainHistoryDict', "rb"))
[ ]: history['val_loss'].index(min(history['val_loss'])) + 1
[ ]: 17
[ ]: def display_training_curves(training, validation, title, subplot):
    ax = plt.subplot(subplot)
    ax.plot(training)
    ax.plot(validation)
    ax.set_title('model ' + title)
    ax.set_ylabel(title)
    ax.set_xlabel('epoch')
    ax.legend(['training', 'validation'])

    plt.subplots(figsize=(10,10))
    plt.tight_layout()
    display_training_curves(history['binary_accuracy'], ↪
       ↪history['val_binary_accuracy'], 'accuracy', 211)
    display_training_curves(history['loss'], history['val_loss'], 'loss', 212)
```



```
[ ]: ConfusionMatrixDisplay(confusion_matrix(Y_test, tf.argmax(model.predict(X_test), axis = 1))).plot()
plt.savefig(folder + "confusionmatrix.jpeg")
plt.show()
```

Model3_Resnet50_BhSig260BengaliTraining

November 14, 2024

1 Model - 3 (ResNet)

This model is a ResNet Model for the task of automatic verification of offline signatures.

```
[ ]: from tensorflow.keras.layers import Activation, Add, AvgPool2D, ↵BatchNormalization, Conv2D, Dense, Flatten, Input, MaxPool2D, ZeroPadding2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.initializers import random_uniform, glorot_uniform, ↵constant
from tensorflow.keras.utils import plot_model
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
```

```
[ ]: def identity_block(X, f, filters, training=True, initializer=random_uniform, u
↳activation='relu', seed=0):
    F1, F2, F3 = filters
    Y = X

    Y = Conv2D(filters=F1, kernel_size=1, strides=1, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F2, kernel_size=f, strides=1, padding='same', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F3, kernel_size=1, strides=1, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)

    Y = Add() ([Y, X])
    Y = Activation(activation = activation) (Y)

    return Y
```

```
[ ]: def convolutional_block(X, f, filters, s=2, training=True, u
↳initializer=glorot_uniform, activation='relu', seed=0):
    F1, F2, F3 = filters
    Y = X

    Y = Conv2D(filters=F1, kernel_size=1, strides=s, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F2, kernel_size=f, strides=1, padding='same', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F3, kernel_size=1, strides=1, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)

    X = Conv2D(filters=F3, kernel_size=1, strides=s, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (X)
    X = BatchNormalization() (X, training=training)
```

```

Y = Add() ([Y, X])
Y = Activation(activation = activation) (Y)

return Y

```

[]:

```

def ResNet50(input_shape = (512, 512, 3), classes = 2, activation='relu', ↴
    seed=0):
    Y = Input(input_shape)
    X = Y
    X = ZeroPadding2D(3)(X)

    X = Conv2D(filters=64, kernel_size = 7, strides = 2, kernel_initializer = glorot_uniform(seed=seed)) (X)
    X = BatchNormalization() (X)
    X = Activation(activation=activation) (X)
    X = MaxPool2D((3, 3), strides=(2, 2)) (X)

    X = convolutional_block(X, f = 3, filters = [64, 64, 256], s = 1, seed=seed)
    X = identity_block(X, 3, [64, 64, 256], seed=seed)
    X = identity_block(X, 3, [64, 64, 256], seed=seed)

    X = convolutional_block(X, f = 3, filters = [128, 128, 512], s = 2, ↴
        seed=seed)
    X = identity_block(X, 3, [128, 128, 512], seed=seed)
    X = identity_block(X, 3, [128, 128, 512], seed=seed)
    X = identity_block(X, 3, [128, 128, 512], seed=seed)

    X = convolutional_block(X, f = 3, filters = [256, 256, 1024], s = 2, ↴
        seed=seed)
    X = identity_block(X, 3, [256, 256, 1024], seed=seed)

    X = convolutional_block(X, f = 3, filters = [512, 512, 2048], s = 2, ↴
        seed=seed)
    X = identity_block(X, 3, [512, 512, 2048], seed=seed)
    X = identity_block(X, 3, [512, 512, 2048], seed=seed)

    X = AvgPool2D(pool_size=(2, 2))(X)
    X = Flatten()(X)
    X = Dense(classes, activation='softmax', kernel_initializer = glorot_uniform(seed=seed))(X)

model = Model(inputs = Y, outputs = X)

```

```

    return model

[ ]: plot_model(ResNet50(input_shape=(64, 128, 3)), show_shapes=True)

[ ]: from h5py import File
from matplotlib import pyplot
import numpy as np
import os
from tensorflow import one_hot, reshape

[ ]: from google.colab import drive
drive.mount('/content/drive')

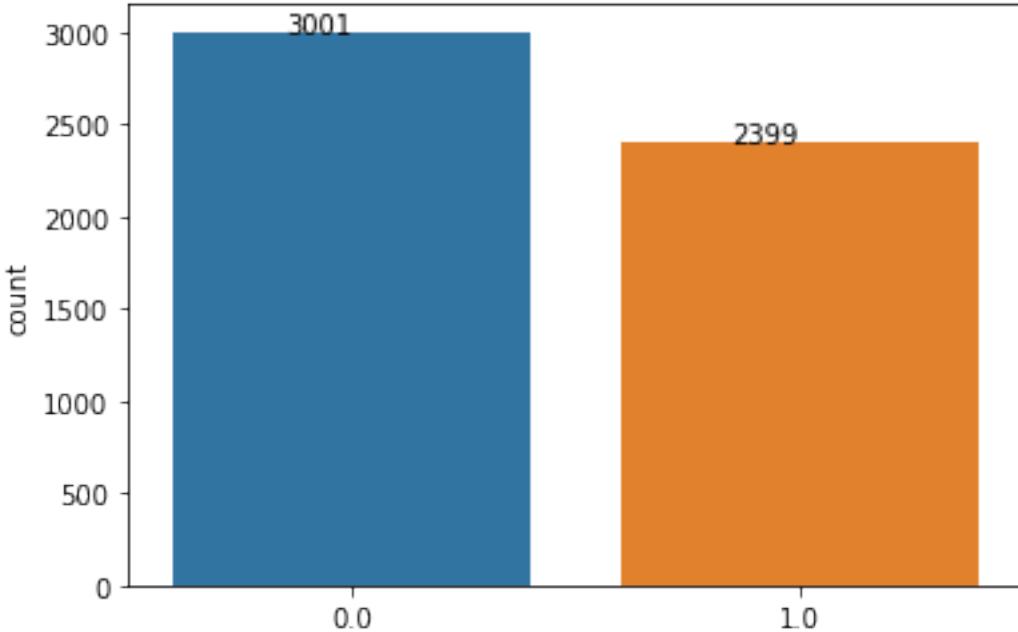
Mounted at /content/drive

[ ]: # database = input('Database Name: ')
# database = database.lower() + '_128x64.h5'
# file = os.path.join(os.getcwd(), 'database', database)
file = "/content/drive/MyDrive/bhsig260bengali_128x643.h5"
print(file)
try:
    with File(file, 'r') as hdf:
        X = np.array(hdf.get('X'))
        Y = np.array(hdf.get('Y'))
except Exception as ex:
    template = "An exception of type {0} occurred. Arguments:\n{1!r}"
    message = template.format(type(ex).__name__, ex.args)
    print(message)
X = X / 255.0
Y = Y * 1.0
# Y = one_hot(Y * 1.0, depth=2)
# Y = reshape(Y, (-1, 2))
print("Feature shape =", X.shape)
print("Label shape =", Y.shape)

/content/drive/MyDrive/bhsig260bengali_128x643.h5
Feature shape = (5400, 64, 128, 3)
Label shape = (5400, 1)

[ ]: ax = sns.countplot(x = Y.reshape(Y.shape[0]))
for p in ax.patches:
    ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))

```



```
[ ]: import sklearn
class_weights = sklearn.utils.class_weight.compute_class_weight(class_weight = "balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict
```

```
[ ]: {0: 0.8997000999666778, 1: 1.1254689453939142}
```

```
[ ]: seed=randint(10)
# metadata['split_seed']=seed
print('seed =', seed)
indices = permutation(X.shape[0])
# print(X)
same = True
if same:
    m = int(0.70 * X.shape[0])
    n = int(0.15 * X.shape[0])
    training_id, validation_id, test_id = indices[:m], indices[m: m + n], indices[m+n:]
    X_train, X_test, X_validate = X[training_id], X[test_id], X[validation_id]
    Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]
else:
    m = int(0.70 * X.shape[0])
    training_id, validation_id = indices[:m], indices[m:]
    X_train, X_validate = X[training_id], X[validation_id]
```

```
Y_train, Y_validate = Y[training_id], Y[validation_id]
print("Shape of Features in training set =", X_train.shape)
print("Shape of Labels in training set =", Y_train.shape)
```

```
del Y,X
```

```
seed = 3
Shape of Features in training set = (3779, 64, 128, 3)
Shape of Labels in training set = (3779, 1)
```

```
[ ]: #One hot Encoding
```

```
Y_train = one_hot(Y_train, depth=2)
Y_train = reshape(Y_train, (-1, 2))

Y_test = one_hot(Y_test, depth=2)
Y_test = reshape(Y_test, (-1, 2))

Y_validate = one_hot(Y_validate, depth=2)
Y_validate = reshape(Y_validate, (-1, 2))
```

```
[ ]: print("Shape of Labels in training set =", Y_train.shape)
```

```
Shape of Labels in training set = (3779, 2)
```

```
[ ]: with tf.device('/device:GPU:0'):
```

```
    model = ResNet50(input_shape=(64, 128, 3))
    SGD = tf.keras.optimizers.SGD(learning_rate = 1e-02,momentum = 0.9)
    model.compile(optimizer = SGD, loss='binary_crossentropy',  
      metrics=['accuracy'])
```

1.1 Model Training for 10 epochs

Optimizer - Adam (default hyperparameter)

Epochs - 10

```
[ ]: with tf.device('/device:GPU:0'):
    history = model.fit(X_train, Y_train, epochs = 10, batch_size = 32,  
      validation_data=(X_validate, Y_validate),shuffle=True
                  ,class_weight = class_weight_dict)
print('training done')
```

Epoch 1/10

```
119/119 [=====] - 16s 95ms/step - loss: 1.9024 -  
accuracy: 0.5578 - val_loss: 0.7199 - val_accuracy: 0.6704
```

Epoch 2/10

```
119/119 [=====] - 10s 88ms/step - loss: 1.1240 -  
accuracy: 0.6055 - val_loss: 0.6314 - val_accuracy: 0.6951
```

Epoch 3/10

```
119/119 [=====] - 10s 86ms/step - loss: 0.7371 -
```

```

accuracy: 0.6740 - val_loss: 0.6902 - val_accuracy: 0.6790
Epoch 4/10
119/119 [=====] - 10s 84ms/step - loss: 0.7697 -
accuracy: 0.6600 - val_loss: 0.6341 - val_accuracy: 0.6975
Epoch 5/10
119/119 [=====] - 10s 85ms/step - loss: 0.6997 -
accuracy: 0.6960 - val_loss: 0.5858 - val_accuracy: 0.7235
Epoch 6/10
119/119 [=====] - 10s 85ms/step - loss: 0.6792 -
accuracy: 0.7020 - val_loss: 0.7866 - val_accuracy: 0.5272
Epoch 7/10
119/119 [=====] - 10s 85ms/step - loss: 0.6327 -
accuracy: 0.7004 - val_loss: 0.5366 - val_accuracy: 0.7605
Epoch 8/10
119/119 [=====] - 10s 87ms/step - loss: 0.6312 -
accuracy: 0.7438 - val_loss: 0.7071 - val_accuracy: 0.6531
Epoch 9/10
119/119 [=====] - 10s 83ms/step - loss: 0.6014 -
accuracy: 0.7505 - val_loss: 0.4950 - val_accuracy: 0.7802
Epoch 10/10
119/119 [=====] - 10s 83ms/step - loss: 0.4857 -
accuracy: 0.7804 - val_loss: 0.6288 - val_accuracy: 0.7333
training done

```

```
[ ]: import pickle
folder = "/content/drive/MyDrive/BengaliResNetEPOCHS10/"
```

```
[ ]: with open(folder + 'trainHistoryDict', 'wb') as file_pi:
    pickle.dump(history, file_pi)
model.save(folder + 'model.h5')
```

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 53). These functions will not be directly callable after loading.

```
[ ]: history = pickle.load(open(folder + 'trainHistoryDict', "rb"))
model = tf.keras.models.load_model(folder + 'model.h5')
```

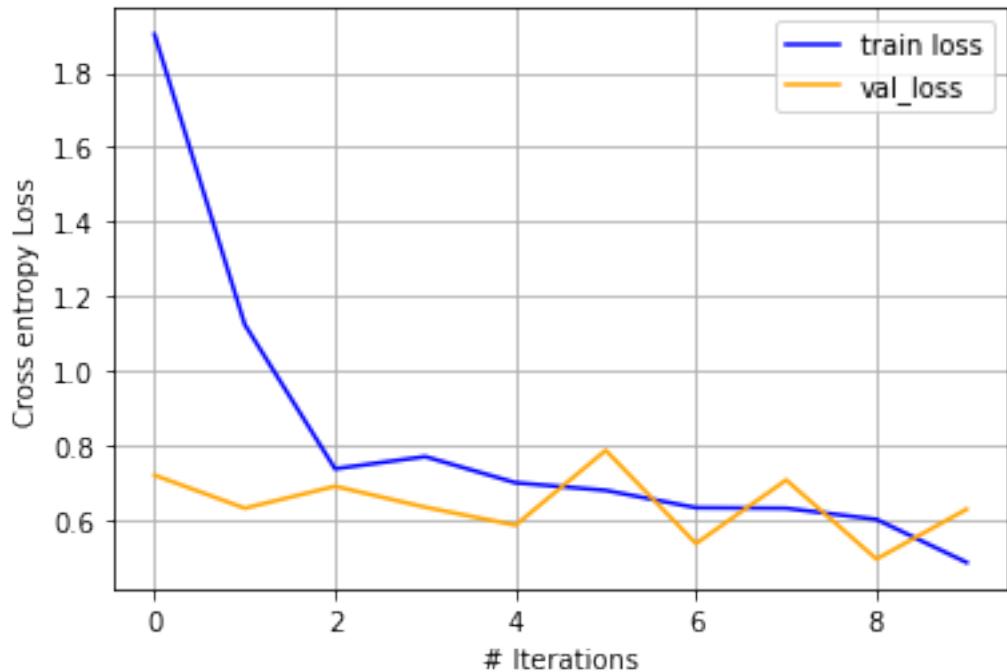
1.2 Plotting

Optimizer - Adam (default hyperparameter)

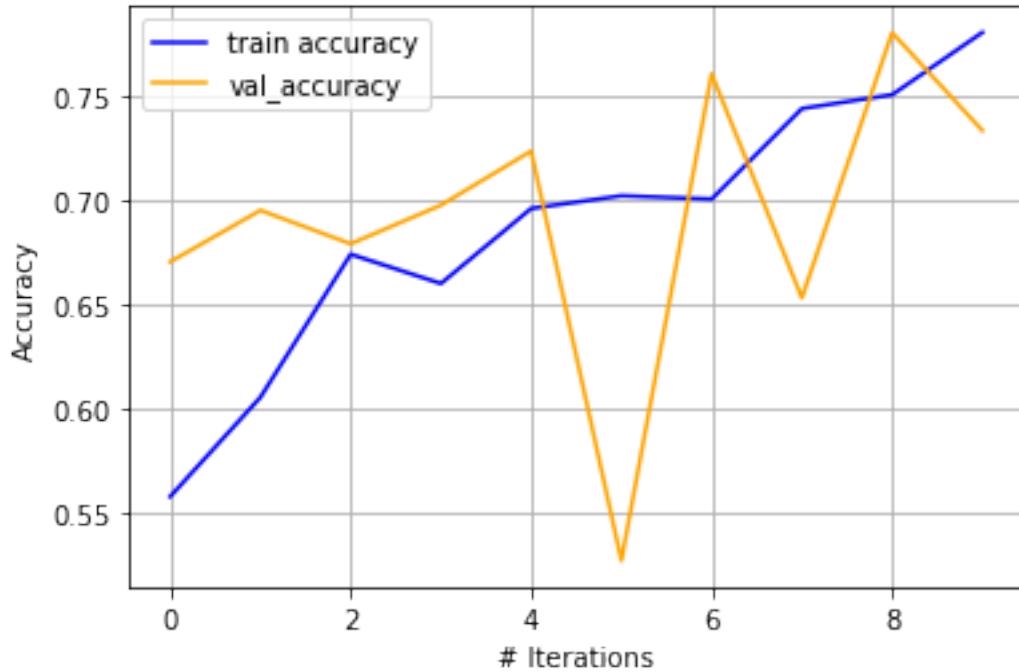
Epochs - 10

```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Cross entropy Loss')
plt.plot(history.history['loss'], color='blue',label = "train loss")
plt.plot(history.history['val_loss'], color='orange',label = "val_loss")
```

```
plt.legend()  
plt.grid()  
plt.savefig(folder + "loss.jpeg")  
plt.show()
```

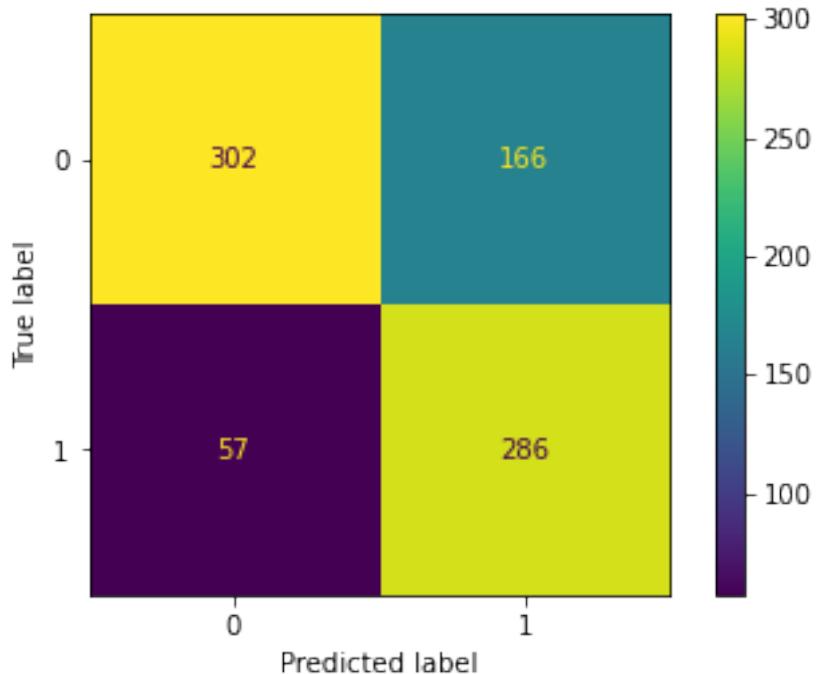


```
[ ]: plt.xlabel('# Iterations')  
plt.ylabel('Accuracy')  
plt.plot(history.history['accuracy'], color='blue',label = "train accuracy")  
plt.plot(history.history['val_accuracy'], color='orange',label = "val_accuracy")  
plt.grid()  
plt.legend()  
plt.savefig(folder + "accuracy.jpeg")  
plt.show()
```



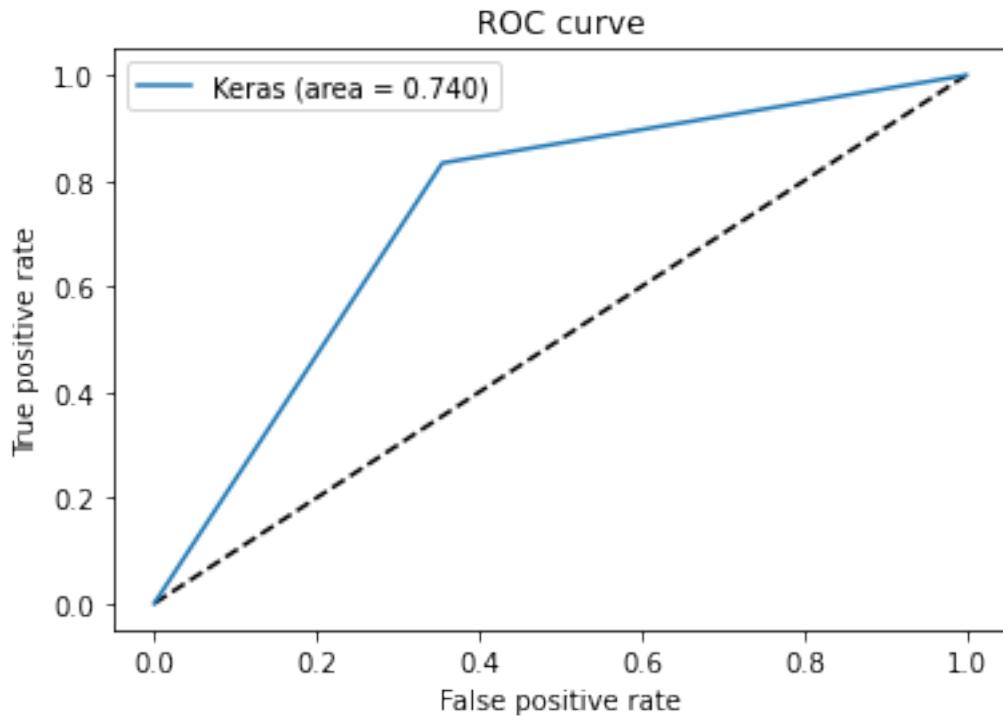
```
[ ]: ConfusionMatrixDisplay(confusion_matrix(tf.argmax(Y_test, axis = 1), tf.  
    ↪argmax(model.predict(X_test), axis = 1))).plot()  
plt.savefig( folder + "confusionmatrix.jpeg")  
plt.show()
```

26/26 [=====] - 2s 29ms/step



```
[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = tf.argmax(model.predict(X_test),axis = 1)
fpr_keras, tpr_keras, thresholds_keras = roc_curve(tf.argmax(Y_test,axis = 1),y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.jpeg")
plt.show()
```

26/26 [=====] - 1s 25ms/step



1.3 Model Training for 100 epochs

Optimizer - SGD (lr = 0.01)

Epochs - 100 (Early Stopping)

```
[ ]: import pickle
folder = "/content/drive/MyDrive/BengaliResNetEPOCHS100/"

[ ]: with tf.device('/device:GPU:0'):
    callbacks = [
        EarlyStopping(patience = 5, verbose = 1),
        ModelCheckpoint(folder + 'model-{epoch:03d}.h5'
            , verbose=1, save_best_only = True)
    ]

[ ]: with tf.device('/device:GPU:0'):
    history = model.fit(X_train, Y_train, epochs = 100, batch_size = 32,
        validation_data=(X_validate, Y_validate), shuffle=True
        , class_weight = class_weight_dict, callbacks = callbacks)
print('training done')
```

Epoch 1/100
119/119 [=====] - ETA: 0s - loss: 1.7476 - accuracy: 0.5335

```
Epoch 1: val_loss improved from inf to 0.85154, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-001.h5
119/119 [=====] - 26s 110ms/step - loss: 1.7476 -
accuracy: 0.5335 - val_loss: 0.8515 - val_accuracy: 0.5852
Epoch 2/100
119/119 [=====] - ETA: 0s - loss: 0.9917 - accuracy:
0.6031
Epoch 2: val_loss did not improve from 0.85154
119/119 [=====] - 10s 85ms/step - loss: 0.9917 -
accuracy: 0.6031 - val_loss: 0.9137 - val_accuracy: 0.5938
Epoch 3/100
118/119 [=====>.] - ETA: 0s - loss: 0.7554 - accuracy:
0.6552
Epoch 3: val_loss improved from 0.85154 to 0.66213, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-003.h5
119/119 [=====] - 11s 95ms/step - loss: 0.7550 -
accuracy: 0.6555 - val_loss: 0.6621 - val_accuracy: 0.7148
Epoch 4/100
118/119 [=====>.] - ETA: 0s - loss: 0.7654 - accuracy:
0.6547
Epoch 4: val_loss did not improve from 0.66213
119/119 [=====] - 10s 85ms/step - loss: 0.7688 -
accuracy: 0.6544 - val_loss: 0.7625 - val_accuracy: 0.5481
Epoch 5/100
118/119 [=====>.] - ETA: 0s - loss: 0.7498 - accuracy:
0.6602
Epoch 5: val_loss did not improve from 0.66213
119/119 [=====] - 10s 88ms/step - loss: 0.7502 -
accuracy: 0.6602 - val_loss: 0.6917 - val_accuracy: 0.6556
Epoch 6/100
118/119 [=====>.] - ETA: 0s - loss: 0.6501 - accuracy:
0.7124
Epoch 6: val_loss improved from 0.66213 to 0.64725, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-006.h5
119/119 [=====] - 11s 94ms/step - loss: 0.6522 -
accuracy: 0.7118 - val_loss: 0.6472 - val_accuracy: 0.6568
Epoch 7/100
118/119 [=====>.] - ETA: 0s - loss: 0.6646 - accuracy:
0.7013
Epoch 7: val_loss did not improve from 0.64725
119/119 [=====] - 10s 85ms/step - loss: 0.6645 -
accuracy: 0.7012 - val_loss: 0.9369 - val_accuracy: 0.5988
Epoch 8/100
118/119 [=====>.] - ETA: 0s - loss: 0.6576 - accuracy:
0.7100
Epoch 8: val_loss improved from 0.64725 to 0.63311, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-008.h5
119/119 [=====] - 11s 92ms/step - loss: 0.6580 -
```

```
accuracy: 0.7097 - val_loss: 0.6331 - val_accuracy: 0.7333
Epoch 9/100
118/119 [=====>.] - ETA: 0s - loss: 0.6322 - accuracy:
0.7338
Epoch 9: val_loss did not improve from 0.63311
119/119 [=====] - 10s 83ms/step - loss: 0.6325 -
accuracy: 0.7335 - val_loss: 0.9206 - val_accuracy: 0.5457
Epoch 10/100
118/119 [=====>.] - ETA: 0s - loss: 0.5946 - accuracy:
0.7381
Epoch 10: val_loss improved from 0.63311 to 0.58896, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-010.h5
119/119 [=====] - 11s 93ms/step - loss: 0.5949 -
accuracy: 0.7380 - val_loss: 0.5890 - val_accuracy: 0.7568
Epoch 11/100
118/119 [=====>.] - ETA: 0s - loss: 0.5800 - accuracy:
0.7418
Epoch 11: val_loss improved from 0.58896 to 0.57073, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-011.h5
119/119 [=====] - 12s 97ms/step - loss: 0.5802 -
accuracy: 0.7417 - val_loss: 0.5707 - val_accuracy: 0.7531
Epoch 12/100
119/119 [=====] - ETA: 0s - loss: 0.5221 - accuracy:
0.7716
Epoch 12: val_loss did not improve from 0.57073
119/119 [=====] - 10s 84ms/step - loss: 0.5221 -
accuracy: 0.7716 - val_loss: 0.5917 - val_accuracy: 0.7136
Epoch 13/100
118/119 [=====>.] - ETA: 0s - loss: 0.4638 - accuracy:
0.8053
Epoch 13: val_loss improved from 0.57073 to 0.47878, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-013.h5
119/119 [=====] - 11s 93ms/step - loss: 0.4634 -
accuracy: 0.8055 - val_loss: 0.4788 - val_accuracy: 0.7963
Epoch 14/100
118/119 [=====>.] - ETA: 0s - loss: 0.4172 - accuracy:
0.8220
Epoch 14: val_loss improved from 0.47878 to 0.47324, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-014.h5
119/119 [=====] - 11s 95ms/step - loss: 0.4170 -
accuracy: 0.8222 - val_loss: 0.4732 - val_accuracy: 0.8049
Epoch 15/100
118/119 [=====>.] - ETA: 0s - loss: 0.3735 - accuracy:
0.8419
Epoch 15: val_loss improved from 0.47324 to 0.46428, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-015.h5
119/119 [=====] - 11s 97ms/step - loss: 0.3743 -
accuracy: 0.8415 - val_loss: 0.4643 - val_accuracy: 0.8136
```

```
Epoch 16/100
118/119 [=====>.] - ETA: 0s - loss: 0.3894 - accuracy:
0.8403
Epoch 16: val_loss did not improve from 0.46428
119/119 [=====] - 10s 84ms/step - loss: 0.3891 -
accuracy: 0.8404 - val_loss: 0.5115 - val_accuracy: 0.7877
Epoch 17/100
118/119 [=====>.] - ETA: 0s - loss: 0.3111 - accuracy:
0.8776
Epoch 17: val_loss improved from 0.46428 to 0.45888, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-017.h5
119/119 [=====] - 11s 93ms/step - loss: 0.3124 -
accuracy: 0.8775 - val_loss: 0.4589 - val_accuracy: 0.7975
Epoch 18/100
118/119 [=====>.] - ETA: 0s - loss: 0.3301 - accuracy:
0.8628
Epoch 18: val_loss did not improve from 0.45888
119/119 [=====] - 10s 84ms/step - loss: 0.3299 -
accuracy: 0.8629 - val_loss: 0.5235 - val_accuracy: 0.7988
Epoch 19/100
118/119 [=====>.] - ETA: 0s - loss: 0.2622 - accuracy:
0.8922
Epoch 19: val_loss did not improve from 0.45888
119/119 [=====] - 10s 83ms/step - loss: 0.2643 -
accuracy: 0.8920 - val_loss: 0.7233 - val_accuracy: 0.7173
Epoch 20/100
118/119 [=====>.] - ETA: 0s - loss: 0.5069 - accuracy:
0.7728
Epoch 20: val_loss did not improve from 0.45888
119/119 [=====] - 10s 83ms/step - loss: 0.5070 -
accuracy: 0.7727 - val_loss: 0.4629 - val_accuracy: 0.8148
Epoch 21/100
118/119 [=====>.] - ETA: 0s - loss: 0.3212 - accuracy:
0.8713
Epoch 21: val_loss improved from 0.45888 to 0.42735, saving model to
/content/drive/MyDrive/BengaliResNetEPOCHS100/model-021.h5
119/119 [=====] - 11s 93ms/step - loss: 0.3213 -
accuracy: 0.8711 - val_loss: 0.4273 - val_accuracy: 0.8222
Epoch 22/100
118/119 [=====>.] - ETA: 0s - loss: 0.2878 - accuracy:
0.8896
Epoch 22: val_loss did not improve from 0.42735
119/119 [=====] - 10s 84ms/step - loss: 0.2882 -
accuracy: 0.8894 - val_loss: 0.5692 - val_accuracy: 0.7741
Epoch 23/100
118/119 [=====>.] - ETA: 0s - loss: 0.2256 - accuracy:
0.9100
Epoch 23: val_loss did not improve from 0.42735
```

```
119/119 [=====] - 10s 83ms/step - loss: 0.2257 -  
accuracy: 0.9098 - val_loss: 0.4685 - val_accuracy: 0.8160  
Epoch 24/100  
118/119 [=====>.] - ETA: 0s - loss: 0.2302 - accuracy:  
0.9115  
Epoch 24: val_loss did not improve from 0.42735  
119/119 [=====] - 10s 83ms/step - loss: 0.2305 -  
accuracy: 0.9114 - val_loss: 0.5148 - val_accuracy: 0.8309  
Epoch 25/100  
118/119 [=====>.] - ETA: 0s - loss: 0.1787 - accuracy:  
0.9314  
Epoch 25: val_loss did not improve from 0.42735  
119/119 [=====] - 10s 83ms/step - loss: 0.1790 -  
accuracy: 0.9312 - val_loss: 0.6064 - val_accuracy: 0.8111  
Epoch 26/100  
118/119 [=====>.] - ETA: 0s - loss: 0.1661 - accuracy:  
0.9431  
Epoch 26: val_loss did not improve from 0.42735  
119/119 [=====] - 10s 84ms/step - loss: 0.1660 -  
accuracy: 0.9431 - val_loss: 0.6085 - val_accuracy: 0.8173  
Epoch 26: early stopping  
training done
```

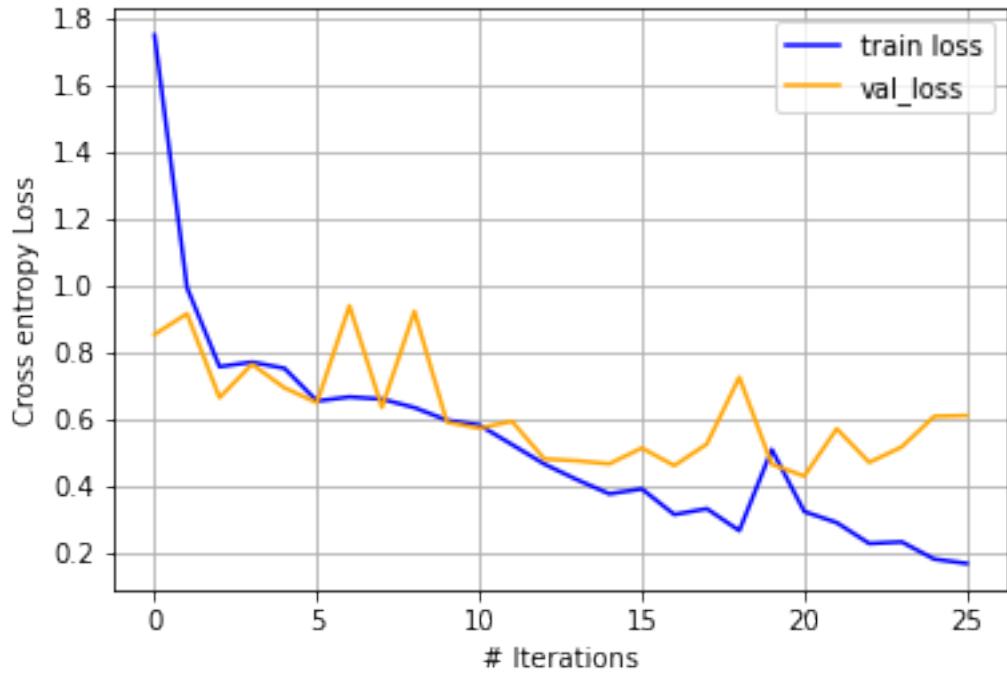
```
[ ]:
```

1.4 Plotting

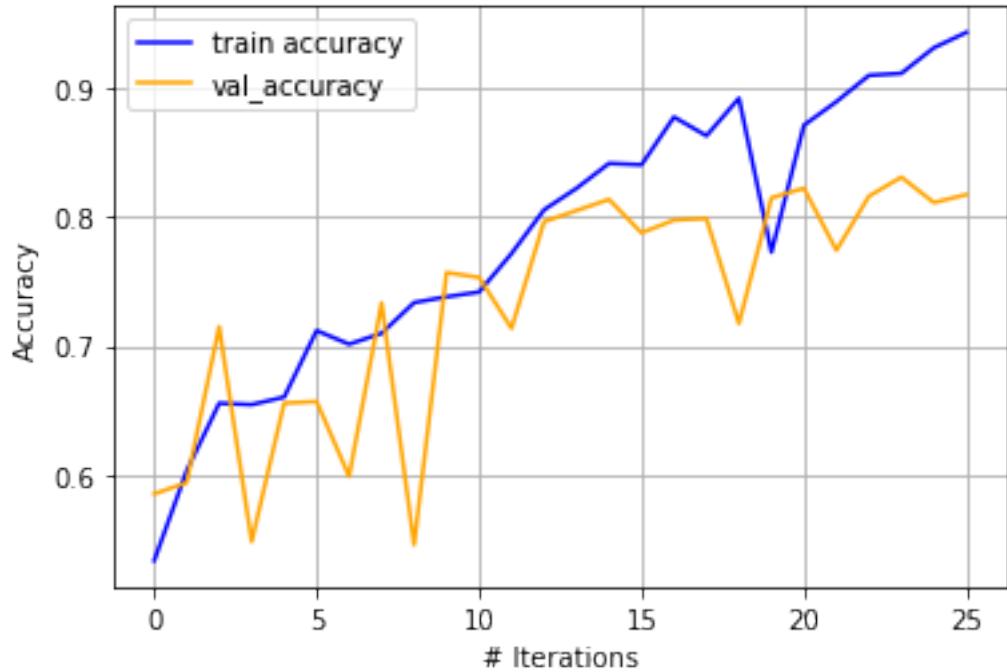
Optimizer - SGD (lr = 0.01)

Epochs - 100 (Early Stopping)

```
[ ]: plt.xlabel('# Iterations')  
plt.ylabel('Cross entropy Loss')  
plt.plot(history.history['loss'], color='blue',label = "train loss")  
plt.plot(history.history['val_loss'], color='orange',label = "val_loss")  
plt.legend()  
plt.grid()  
plt.savefig(folder + "loss.jpeg")  
plt.show()
```

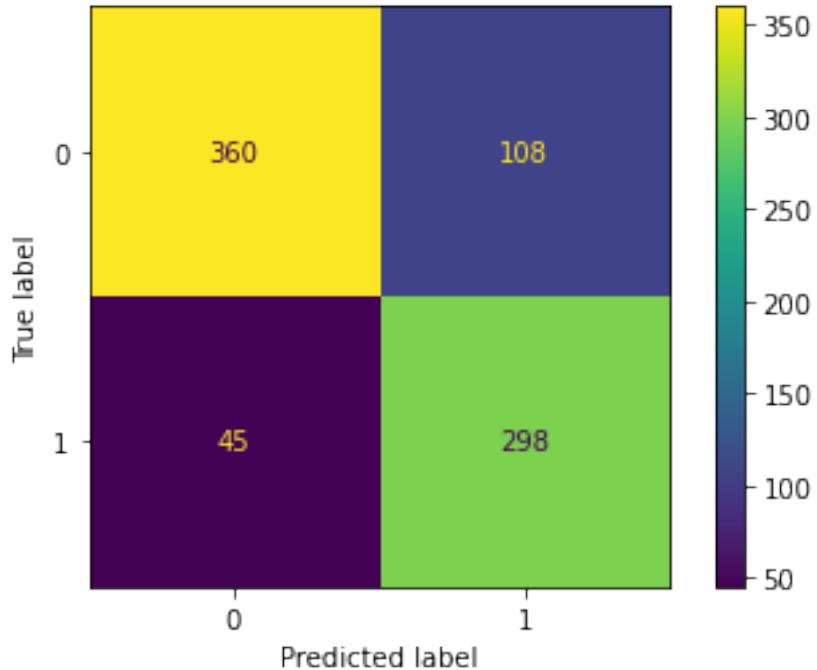


```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Accuracy')
plt.plot(history.history['accuracy'], color='blue',label = "train accuracy")
plt.plot(history.history['val_accuracy'], color='orange',label = "val_accuracy")
plt.grid()
plt.legend()
plt.savefig(folder + "accuracy.jpeg")
plt.show()
```



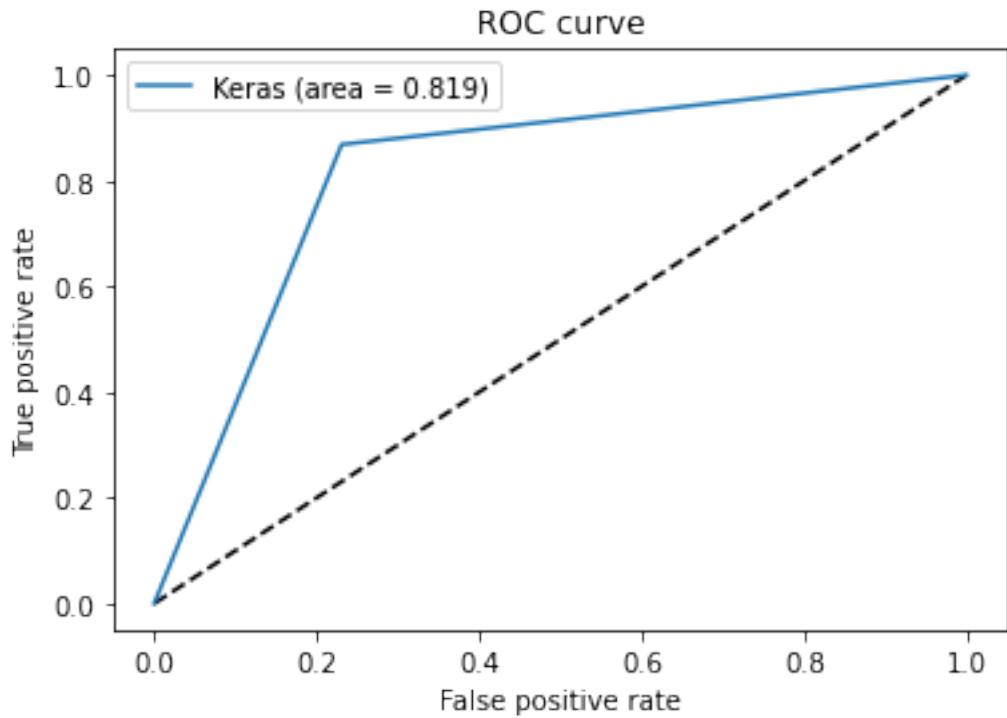
```
[ ]: ConfusionMatrixDisplay(confusion_matrix(tf.argmax(Y_test, axis = 1), tf.  
    ↪argmax(model.predict(X_test), axis = 1))).plot()  
plt.savefig( folder + "confusionmatrix.jpeg")  
plt.show()
```

26/26 [=====] - 1s 25ms/step



```
[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = tf.argmax(model.predict(X_test),axis = 1)
fpr_keras, tpr_keras, thresholds_keras = roc_curve(tf.argmax(Y_test,axis = 1),y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.jpeg")
plt.show()
```

26/26 [=====] - 1s 25ms/step



[]:

Model3_Resnet50_BhSig260HindiTraining

November 14, 2024

1 Model - 3 (ResNet)

This model is a ResNet Model for the task of automatic verification of offline signatures.

```
[ ]: from tensorflow.keras.layers import Activation, Add, AvgPool2D, ↵BatchNormalization, Conv2D, Dense, Flatten, Input, MaxPool2D, ZeroPadding2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.initializers import random_uniform, glorot_uniform, ↵constant
from tensorflow.keras.utils import plot_model
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
```

```
[ ]: def identity_block(X, f, filters, training=True, initializer=random_uniform, u
↳activation='relu', seed=0):
    F1, F2, F3 = filters
    Y = X

    Y = Conv2D(filters=F1, kernel_size=1, strides=1, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F2, kernel_size=f, strides=1, padding='same', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F3, kernel_size=1, strides=1, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)

    Y = Add() ([Y, X])
    Y = Activation(activation = activation) (Y)

    return Y
```

```
[ ]: def convolutional_block(X, f, filters, s=2, training=True, u
↳initializer=glorot_uniform, activation='relu', seed=0):
    F1, F2, F3 = filters
    Y = X

    Y = Conv2D(filters=F1, kernel_size=1, strides=s, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F2, kernel_size=f, strides=1, padding='same', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F3, kernel_size=1, strides=1, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)

    X = Conv2D(filters=F3, kernel_size=1, strides=s, padding='valid', u
↳kernel_initializer=initializer(seed=seed)) (X)
    X = BatchNormalization() (X, training=training)
```

```

Y = Add() ([Y, X])
Y = Activation(activation = activation) (Y)

return Y

```

[]:

```

def ResNet50(input_shape = (512, 512, 3), classes = 2, activation='relu',
    ↪seed=0):
    Y = Input(input_shape)
    X = Y
    X = ZeroPadding2D(3)(X)

    X = Conv2D(filters=64, kernel_size = 7, strides = 2, kernel_initializer =
        ↪glorot_uniform(seed=seed)) (X)
    X = BatchNormalization() (X)
    X = Activation(activation=activation) (X)
    X = MaxPool2D((3, 3), strides=(2, 2)) (X)

    X = convolutional_block(X, f = 3, filters = [64, 64, 256], s = 1, seed=seed)
    X = identity_block(X, 3, [64, 64, 256], seed=seed)
    X = identity_block(X, 3, [64, 64, 256], seed=seed)

    X = convolutional_block(X, f = 3, filters = [128, 128, 512], s = 2,
        ↪seed=seed)
    X = identity_block(X, 3, [128, 128, 512], seed=seed)
    X = identity_block(X, 3, [128, 128, 512], seed=seed)
    X = identity_block(X, 3, [128, 128, 512], seed=seed)

    X = convolutional_block(X, f = 3, filters = [256, 256, 1024], s = 2,
        ↪seed=seed)
    X = identity_block(X, 3, [256, 256, 1024], seed=seed)
    X = identity_block(X, 3, [256, 256, 1024], seed=seed)
    X = identity_block(X, 3, [256, 256, 1024], seed=seed)
    X = identity_block(X, 3, [256, 256, 1024], seed=seed)

    X = convolutional_block(X, f = 3, filters = [512, 512, 2048], s = 2,
        ↪seed=seed)
    X = identity_block(X, 3, [512, 512, 2048], seed=seed)
    X = identity_block(X, 3, [512, 512, 2048], seed=seed)

    X = AvgPool2D(pool_size=(2, 2))(X)
    X = Flatten()(X)
    X = Dense(classes, activation='softmax', kernel_initializer =
        ↪glorot_uniform(seed=seed))(X)

model = Model(inputs = Y, outputs = X)

```

```

    return model

[ ]: plot_model(ResNet50(input_shape=(64, 128, 3)), show_shapes=True)

[ ]: from h5py import File
from matplotlib import pyplot
import numpy as np
import os
from tensorflow import one_hot, reshape

[ ]: from google.colab import drive
drive.mount('/content/drive')

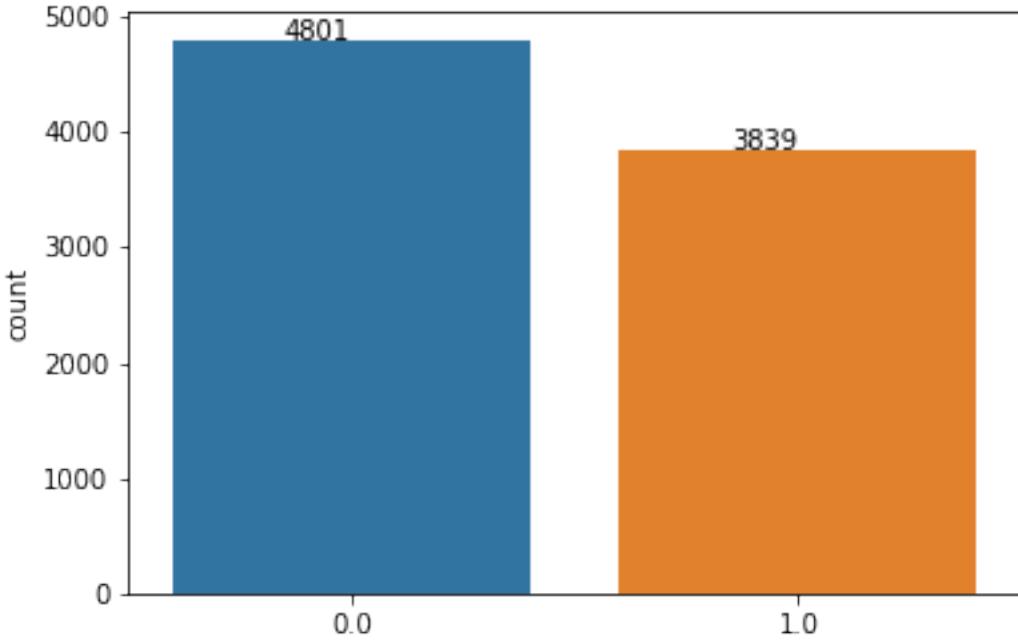
Mounted at /content/drive

[ ]: # database = input('Database Name: ')
# database = database.lower() + '_128x64.h5'
# file = os.path.join(os.getcwd(), 'database', database)
file = "/content/drive/MyDrive/bhsig260hindi_128x643.h5"
print(file)
try:
    with File(file, 'r') as hdf:
        X = np.array(hdf.get('X'))
        Y = np.array(hdf.get('Y'))
except Exception as ex:
    template = "An exception of type {0} occurred. Arguments:\n{1!r}"
    message = template.format(type(ex).__name__, ex.args)
    print(message)
X = X / 255.0
Y = Y * 1.0
# Y = one_hot(Y * 1.0, depth=2)
# Y = reshape(Y, (-1, 2))
print("Feature shape =", X.shape)
print("Label shape =", Y.shape)

/content/drive/MyDrive/bhsig260hindi_128x643.h5
Feature shape = (8640, 64, 128, 3)
Label shape = (8640, 1)

[ ]: ax = sns.countplot(x = Y.reshape(Y.shape[0]))
for p in ax.patches:
    ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))

```



```
[ ]: import sklearn
class_weights = sklearn.utils.class_weight.compute_class_weight(class_weight = "balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict
```

```
[ ]: {0: 0.8998125390543636, 1: 1.1252930450638188}
```

```
[ ]: seed=randint(10)
# metadata['split_seed']=seed
print('seed =', seed)
indices = permutation(X.shape[0])
# print(X)
same = True
if same:
    m = int(0.70 * X.shape[0])
    n = int(0.15 * X.shape[0])
    training_id, validation_id, test_id = indices[:m], indices[m: m + n], indices[m+n:]
    X_train, X_test, X_validate = X[training_id], X[test_id], X[validation_id]
    Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]
else:
    m = int(0.70 * X.shape[0])
    training_id, validation_id = indices[:m], indices[m:]
    X_train, X_validate = X[training_id], X[validation_id]
```

```
Y_train, Y_validate = Y[training_id], Y[validation_id]
print("Shape of Features in training set =", X_train.shape)
print("Shape of Labels in training set =", Y_train.shape)
```

```
del Y,X
```

```
seed = 2
Shape of Features in training set = (6048, 64, 128, 3)
Shape of Labels in training set = (6048, 1)
```

```
[ ]: #One hot Encoding
```

```
Y_train = one_hot(Y_train, depth=2)
Y_train = reshape(Y_train, (-1, 2))
```

```
Y_test = one_hot(Y_test, depth=2)
Y_test = reshape(Y_test, (-1, 2))
```

```
Y_validate = one_hot(Y_validate, depth=2)
Y_validate = reshape(Y_validate, (-1, 2))
```

```
[ ]: print("Shape of Labels in training set =", Y_train.shape)
```

```
Shape of Labels in training set = (6048, 2)
```

```
[ ]: with tf.device('/device:GPU:0'):
```

```
    model = ResNet50(input_shape=(64, 128, 3))
    SGD = tf.keras.optimizers.SGD(learning_rate = 1e-02,momentum = 0.9)
    model.compile(optimizer = SGD, loss='binary_crossentropy',  
      metrics=['accuracy'])
```

1.1 Model Training for 10 epochs

Optimizer - Adam (default hyperparameter)

Epochs - 10

```
[ ]: with tf.device('/device:GPU:0'):
    history = model.fit(X_train, Y_train, epochs = 20, batch_size = 32,  
      validation_data=(X_validate, Y_validate),shuffle=True
                  ,class_weight = class_weight_dict)
    print('training done')
```

```
[ ]: import pickle
folder = "/content/drive/MyDrive/ResNetEPOCHS10/"
```

```
[ ]: with open(folder + 'trainHistoryDict', 'wb') as file_pi:
    pickle.dump(history, file_pi)
model.save(folder + 'model.h5')
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,  
_jit_compiled_convolution_op, _jit_compiled_convolution_op,  
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing  
5 of 53). These functions will not be directly callable after loading.
```

```
[ ]: history = pickle.load(open(folder + 'trainHistoryDict', "rb"))  
model = tf.keras.models.load_model(folder + 'model.h5')
```

1.2 Model Training for 100 epochs

Optimizer - SGD (lr = 0.01)

Epochs - 100 (Early Stopping)

```
[ ]: import pickle  
folder = "/content/drive/MyDrive/ResNetEPOCHS100/"
```

```
[ ]: with tf.device('/device:GPU:0'):  
    callbacks = [  
        EarlyStopping(patience = 5, verbose = 1),  
        ModelCheckpoint(folder + 'model-{epoch:03d}.h5'  
            , verbose=1, save_best_only = True)  
    ]
```

```
[ ]: with tf.device('/device:GPU:0'):  
    history = model.fit(X_train, Y_train, epochs = 100, batch_size = 32,  
    validation_data=(X_validate, Y_validate), shuffle=True  
                    , class_weight = class_weight_dict, callbacks = callbacks)  
print('training done')
```

```
Epoch 1/100  
189/189 [=====] - ETA: 0s - loss: 1.6307 - accuracy:  
0.5577  
Epoch 1: val_loss improved from inf to 0.83957, saving model to  
/content/drive/MyDrive/ResNetEPOCHS100/model-001.h5  
189/189 [=====] - 22s 93ms/step - loss: 1.6307 -  
accuracy: 0.5577 - val_loss: 0.8396 - val_accuracy: 0.6188  
Epoch 2/100  
189/189 [=====] - ETA: 0s - loss: 0.8441 - accuracy:  
0.6293  
Epoch 2: val_loss did not improve from 0.83957  
189/189 [=====] - 15s 81ms/step - loss: 0.8441 -  
accuracy: 0.6293 - val_loss: 1.0157 - val_accuracy: 0.5818  
Epoch 3/100  
189/189 [=====] - ETA: 0s - loss: 0.6793 - accuracy:  
0.6609  
Epoch 3: val_loss improved from 0.83957 to 0.60916, saving model to  
/content/drive/MyDrive/ResNetEPOCHS100/model-003.h5  
189/189 [=====] - 17s 92ms/step - loss: 0.6793 -
```

```
accuracy: 0.6609 - val_loss: 0.6092 - val_accuracy: 0.6914
Epoch 4/100
189/189 [=====] - ETA: 0s - loss: 0.6202 - accuracy: 0.7077
Epoch 4: val_loss did not improve from 0.60916
189/189 [=====] - 15s 82ms/step - loss: 0.6202 - accuracy: 0.7077 - val_loss: 0.6696 - val_accuracy: 0.7045
Epoch 5/100
189/189 [=====] - ETA: 0s - loss: 0.5734 - accuracy: 0.7404
Epoch 5: val_loss improved from 0.60916 to 0.55587, saving model to /content/drive/MyDrive/ResNetEPOCHS100/model-005.h5
189/189 [=====] - 17s 88ms/step - loss: 0.5734 - accuracy: 0.7404 - val_loss: 0.5559 - val_accuracy: 0.7431
Epoch 6/100
189/189 [=====] - ETA: 0s - loss: 0.5457 - accuracy: 0.7449
Epoch 6: val_loss did not improve from 0.55587
189/189 [=====] - 17s 90ms/step - loss: 0.5457 - accuracy: 0.7449 - val_loss: 0.5566 - val_accuracy: 0.7346
Epoch 7/100
189/189 [=====] - ETA: 0s - loss: 0.5266 - accuracy: 0.7553
Epoch 7: val_loss improved from 0.55587 to 0.52419, saving model to /content/drive/MyDrive/ResNetEPOCHS100/model-007.h5
189/189 [=====] - 18s 94ms/step - loss: 0.5266 - accuracy: 0.7553 - val_loss: 0.5242 - val_accuracy: 0.7647
Epoch 8/100
189/189 [=====] - ETA: 0s - loss: 0.5047 - accuracy: 0.7736
Epoch 8: val_loss improved from 0.52419 to 0.50631, saving model to /content/drive/MyDrive/ResNetEPOCHS100/model-008.h5
189/189 [=====] - 17s 92ms/step - loss: 0.5047 - accuracy: 0.7736 - val_loss: 0.5063 - val_accuracy: 0.7816
Epoch 9/100
189/189 [=====] - ETA: 0s - loss: 0.5064 - accuracy: 0.7783
Epoch 9: val_loss did not improve from 0.50631
189/189 [=====] - 16s 87ms/step - loss: 0.5064 - accuracy: 0.7783 - val_loss: 0.5561 - val_accuracy: 0.7762
Epoch 10/100
189/189 [=====] - ETA: 0s - loss: 0.4800 - accuracy: 0.8061
Epoch 10: val_loss improved from 0.50631 to 0.49927, saving model to /content/drive/MyDrive/ResNetEPOCHS100/model-010.h5
189/189 [=====] - 17s 92ms/step - loss: 0.4800 - accuracy: 0.8061 - val_loss: 0.4993 - val_accuracy: 0.7724
Epoch 11/100
```

```
189/189 [=====] - ETA: 0s - loss: 0.4342 - accuracy: 0.8209
Epoch 11: val_loss improved from 0.49927 to 0.48223, saving model to /content/drive/MyDrive/ResNetEPOCHS100/model-011.h5
189/189 [=====] - 18s 95ms/step - loss: 0.4342 - accuracy: 0.8209 - val_loss: 0.4822 - val_accuracy: 0.7832
Epoch 12/100
189/189 [=====] - ETA: 0s - loss: 0.4150 - accuracy: 0.8262
Epoch 12: val_loss improved from 0.48223 to 0.48145, saving model to /content/drive/MyDrive/ResNetEPOCHS100/model-012.h5
189/189 [=====] - 17s 91ms/step - loss: 0.4150 - accuracy: 0.8262 - val_loss: 0.4814 - val_accuracy: 0.7878
Epoch 13/100
189/189 [=====] - ETA: 0s - loss: 0.4070 - accuracy: 0.8294
Epoch 13: val_loss did not improve from 0.48145
189/189 [=====] - 16s 84ms/step - loss: 0.4070 - accuracy: 0.8294 - val_loss: 0.5036 - val_accuracy: 0.7932
Epoch 14/100
189/189 [=====] - ETA: 0s - loss: 0.4069 - accuracy: 0.8358
Epoch 14: val_loss improved from 0.48145 to 0.46418, saving model to /content/drive/MyDrive/ResNetEPOCHS100/model-014.h5
189/189 [=====] - 17s 90ms/step - loss: 0.4069 - accuracy: 0.8358 - val_loss: 0.4642 - val_accuracy: 0.8009
Epoch 15/100
189/189 [=====] - ETA: 0s - loss: 0.3482 - accuracy: 0.8603
Epoch 15: val_loss improved from 0.46418 to 0.44445, saving model to /content/drive/MyDrive/ResNetEPOCHS100/model-015.h5
189/189 [=====] - 17s 91ms/step - loss: 0.3482 - accuracy: 0.8603 - val_loss: 0.4444 - val_accuracy: 0.8187
Epoch 16/100
189/189 [=====] - ETA: 0s - loss: 0.3328 - accuracy: 0.8652
Epoch 16: val_loss did not improve from 0.44445
189/189 [=====] - 16s 87ms/step - loss: 0.3328 - accuracy: 0.8652 - val_loss: 0.4485 - val_accuracy: 0.8063
Epoch 17/100
189/189 [=====] - ETA: 0s - loss: 0.3062 - accuracy: 0.8824
Epoch 17: val_loss did not improve from 0.44445
189/189 [=====] - 16s 87ms/step - loss: 0.3062 - accuracy: 0.8824 - val_loss: 0.4660 - val_accuracy: 0.8086
Epoch 18/100
189/189 [=====] - ETA: 0s - loss: 0.2921 - accuracy: 0.8862
```

```

Epoch 18: val_loss did not improve from 0.44445
189/189 [=====] - 16s 84ms/step - loss: 0.2921 -
accuracy: 0.8862 - val_loss: 0.5169 - val_accuracy: 0.8063
Epoch 19/100
189/189 [=====] - ETA: 0s - loss: 0.2547 - accuracy:
0.9026
Epoch 19: val_loss did not improve from 0.44445
189/189 [=====] - 16s 84ms/step - loss: 0.2547 -
accuracy: 0.9026 - val_loss: 0.4752 - val_accuracy: 0.8032
Epoch 20/100
189/189 [=====] - ETA: 0s - loss: 0.2511 - accuracy:
0.9034
Epoch 20: val_loss did not improve from 0.44445
189/189 [=====] - 16s 84ms/step - loss: 0.2511 -
accuracy: 0.9034 - val_loss: 0.4838 - val_accuracy: 0.8110
Epoch 20: early stopping
training done

```

```
[ ]: # with open(folder + 'trainHistoryDict', 'wb') as file_pi:
#     pickle.dump(history, file_pi)
# model.save(folder + 'BestModel.h5')
```

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 53). These functions will not be directly callable after loading.

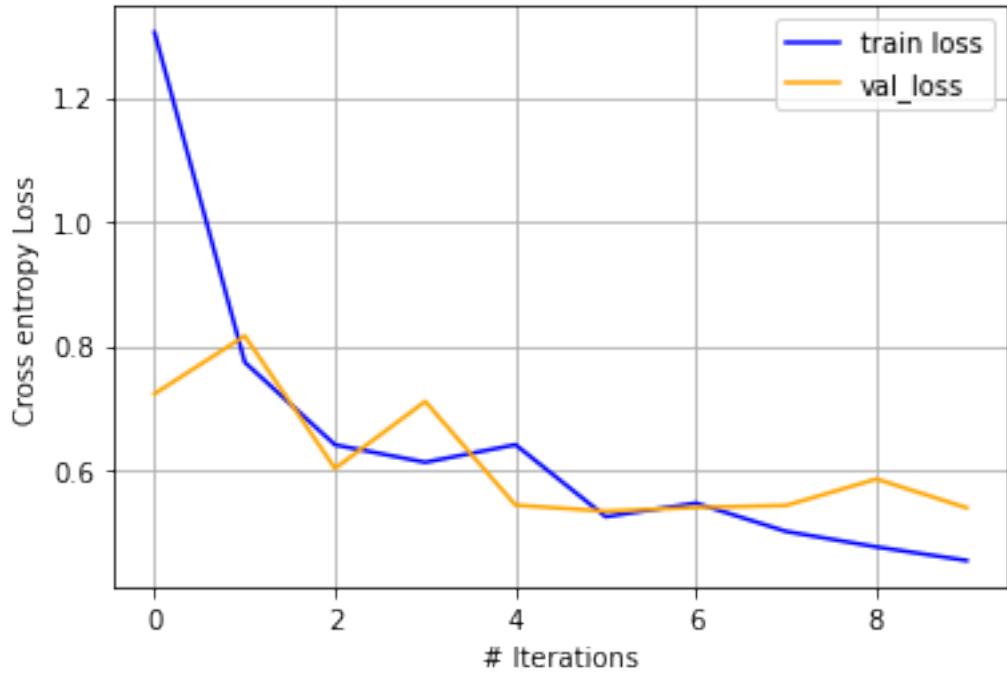
```
[ ]: history = pickle.load(open(folder + 'trainHistoryDict', "rb"))
# model = tf.keras.models.load_model(folder + 'BestModel.h5')
```

1.3 Plotting

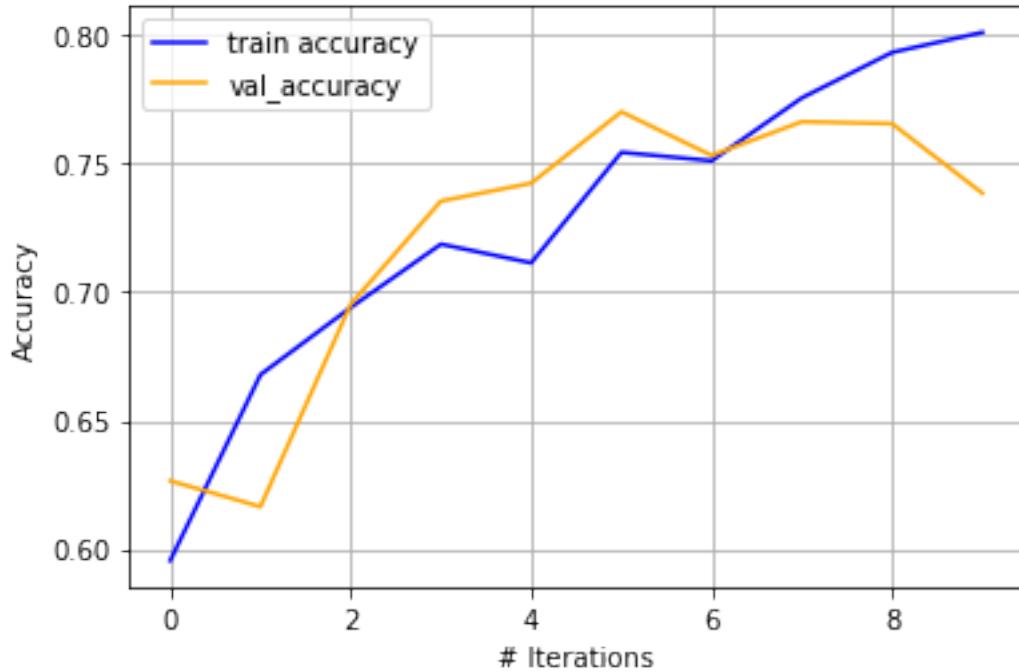
Optimizer - Adam (default hyperparameter)

Epochs - 10

```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Cross entropy Loss')
plt.plot(history.history['loss'], color='blue', label = "train loss")
plt.plot(history.history['val_loss'], color='orange', label = "val_loss")
plt.legend()
plt.grid()
plt.savefig(folder + "loss.jpeg")
plt.show()
```

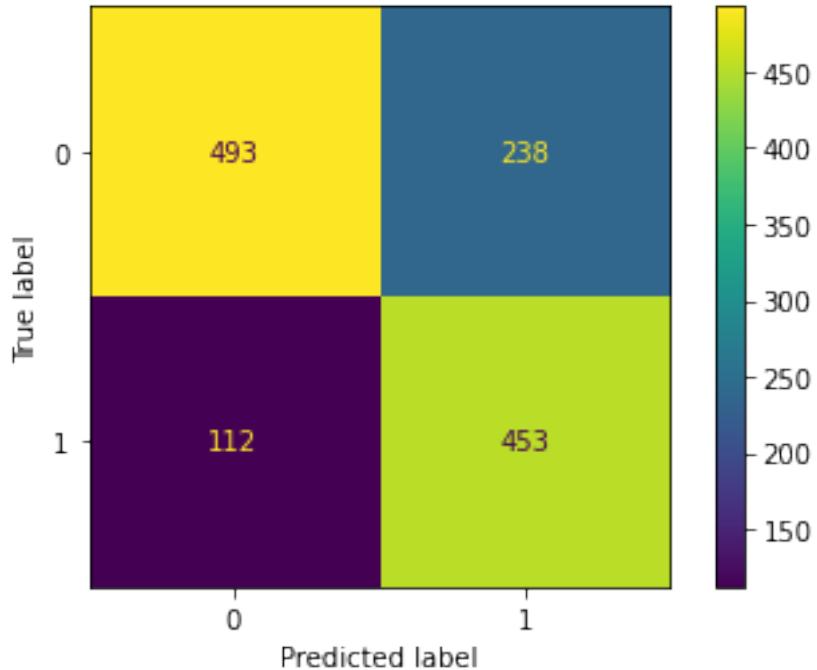


```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Accuracy')
plt.plot(history.history['accuracy'], color='blue',label = "train accuracy")
plt.plot(history.history['val_accuracy'], color='orange',label = "val_accuracy")
plt.grid()
plt.legend()
plt.savefig(folder + "accuracy.jpeg")
plt.show()
```



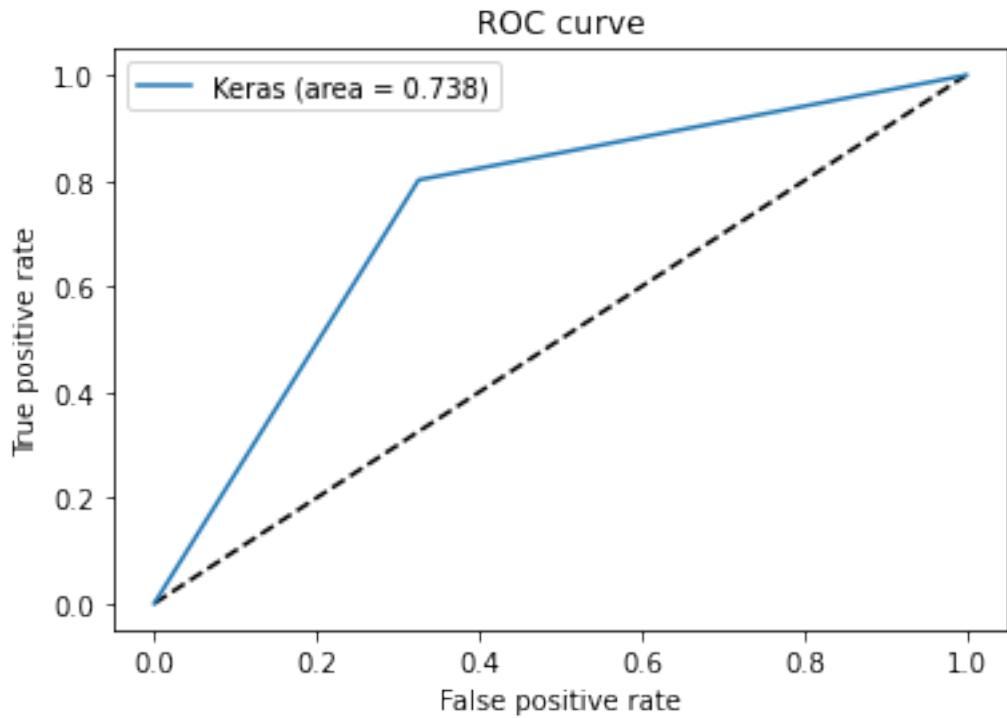
```
[ ]: ConfusionMatrixDisplay(confusion_matrix(tf.argmax(Y_test, axis = 1), tf.  
    ↪argmax(model.predict(X_test), axis = 1))).plot()  
plt.savefig( folder + "confusionmatrix.jpeg")  
plt.show()
```

41/41 [=====] - 2s 27ms/step



```
[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = tf.argmax(model.predict(X_test),axis = 1)
fpr_keras, tpr_keras, thresholds_keras = roc_curve(tf.argmax(Y_test,axis = 1),y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.jpeg")
plt.show()
```

41/41 [=====] - 1s 25ms/step

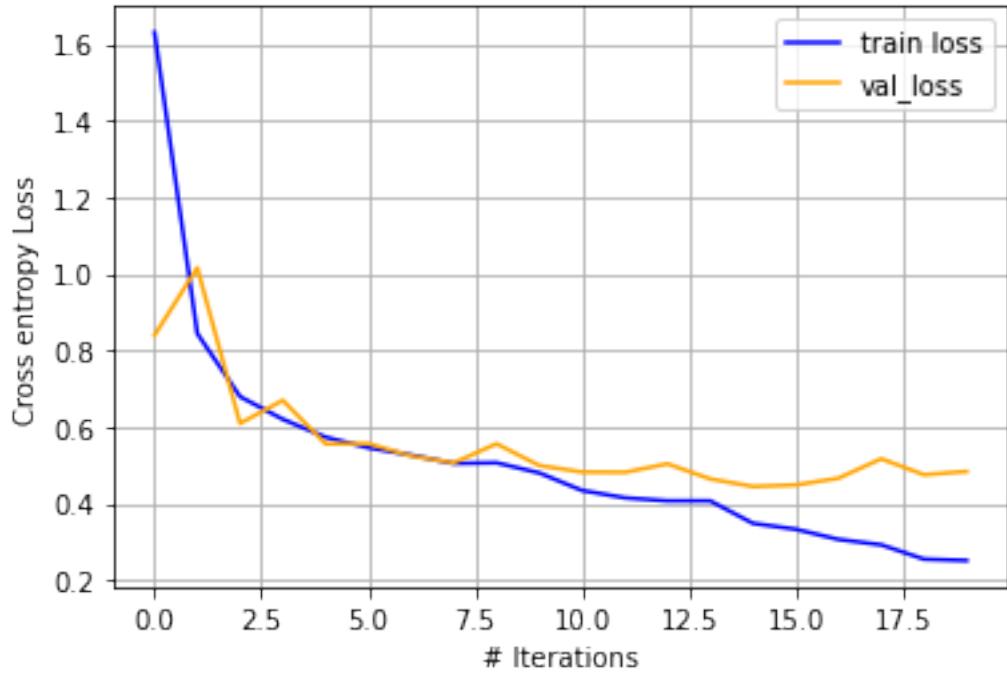


1.4 Plotting

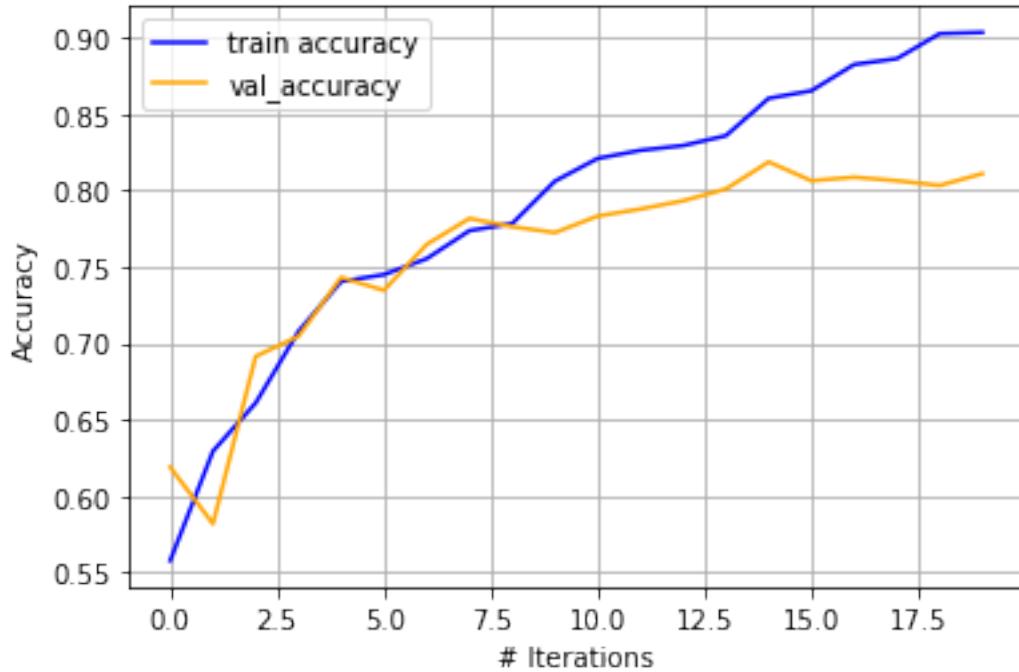
Optimizer - SGD (lr = 0.01)

Epochs - 100 (Early Stopping)

```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Cross entropy Loss')
plt.plot(history.history['loss'], color='blue',label = "train loss")
plt.plot(history.history['val_loss'], color='orange',label = "val_loss")
plt.legend()
plt.grid()
plt.savefig(folder + "loss.jpeg")
plt.show()
```

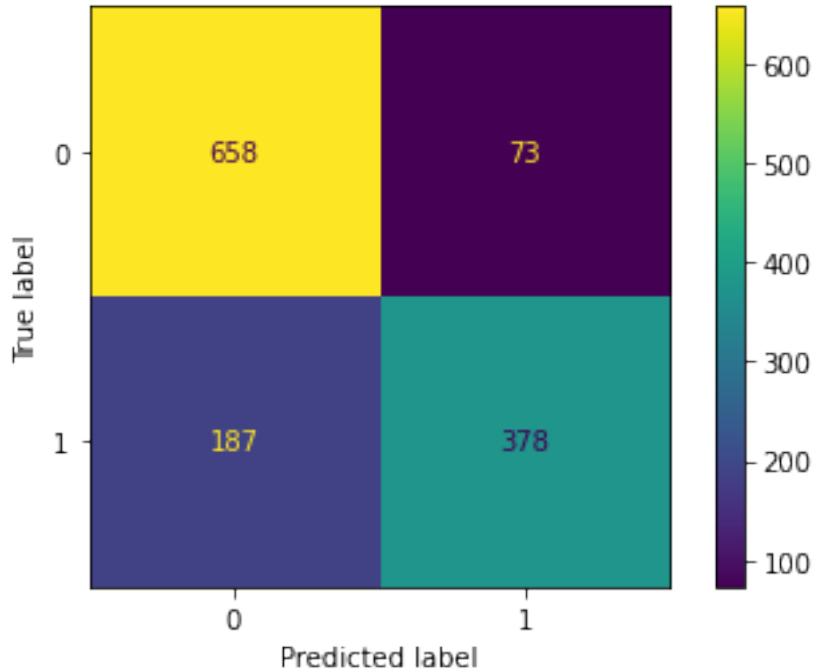


```
[ ]: plt.xlabel('# Iterations')
plt.ylabel('Accuracy')
plt.plot(history.history['accuracy'], color='blue',label = "train accuracy")
plt.plot(history.history['val_accuracy'], color='orange',label = "val_accuracy")
plt.grid()
plt.legend()
plt.savefig(folder + "accuracy.jpeg")
plt.show()
```



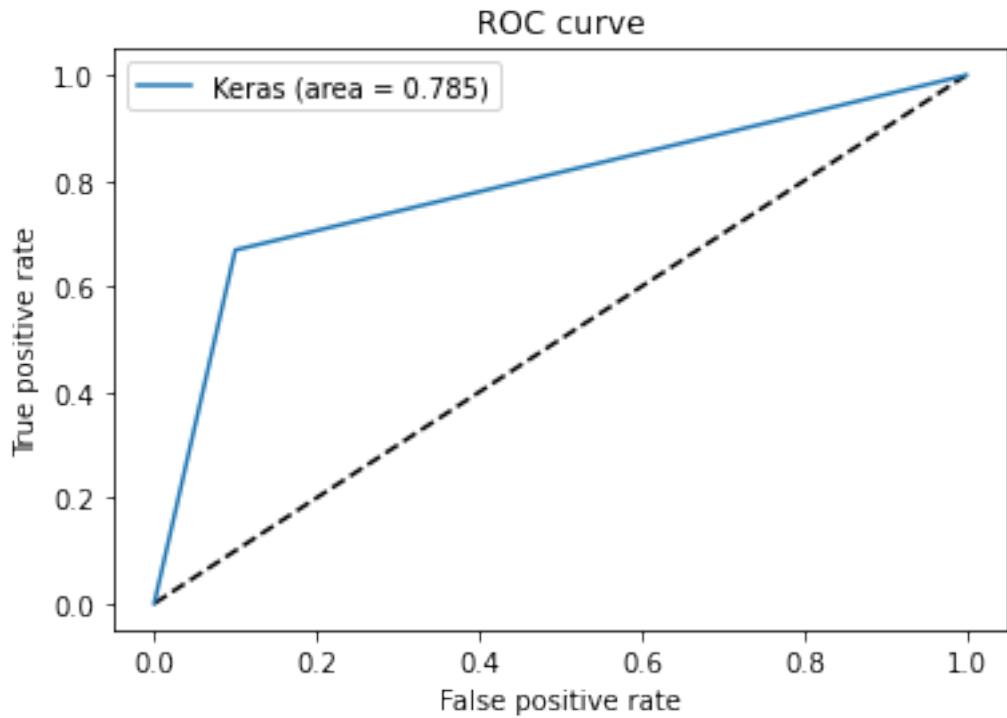
```
[ ]: ConfusionMatrixDisplay(confusion_matrix(tf.argmax(Y_test, axis = 1), tf.  
    ↪argmax(model.predict(X_test), axis = 1))).plot()  
plt.savefig( folder + "confusionmatrix.jpeg")  
plt.show()
```

41/41 [=====] - 2s 27ms/step



```
[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = tf.argmax(model.predict(X_test),axis = 1)
fpr_keras, tpr_keras, thresholds_keras = roc_curve(tf.argmax(Y_test,axis = 1),y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.jpeg")
plt.show()
```

41/41 [=====] - 1s 25ms/step



Model - 3 (ResNet)

This model is a ResNet Model for the task of automatic verification of offline signatures.

```
from tensorflow.keras.layers import Activation, Add, AvgPool2D,
BatchNormalization, Conv2D, Dense, Flatten, Input, MaxPool2D,
ZeroPadding2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.initializers import random_uniform,
glorot_uniform, constant
from tensorflow.keras.utils import plot_model
from keras.callbacks import EarlyStopping, ModelCheckpoint,
ReduceLROnPlateau

from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model

def identity_block(X, f, filters, training=True,
initializer=random_uniform, activation='relu', seed=0):
    F1, F2, F3 = filters
    Y = X

    Y = Conv2D(filters=F1, kernel_size=1, strides=1, padding='valid',
kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
```

```

Y = Activation(activation = activation) (Y)

Y = Conv2D(filters=F2, kernel_size=f, strides=1, padding='same',
kernel_initializer=initializer(seed=seed)) (Y)
Y = BatchNormalization() (Y, training=training)
Y = Activation(activation = activation) (Y)

Y = Conv2D(filters=F3, kernel_size=1, strides=1, padding='valid',
kernel_initializer=initializer(seed=seed)) (Y)
Y = BatchNormalization() (Y, training=training)

Y = Add() ([Y, X])
Y = Activation(activation = activation) (Y)

return Y

def convolutional_block(X, f, filters, s=2, training=True,
initializer=glorot_uniform, activation='relu', seed=0):
    F1, F2, F3 = filters
    Y = X

    Y = Conv2D(filters=F1, kernel_size=1, strides=s, padding='valid',
kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F2, kernel_size=f, strides=1, padding='same',
kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)
    Y = Activation(activation = activation) (Y)

    Y = Conv2D(filters=F3, kernel_size=1, strides=1, padding='valid',
kernel_initializer=initializer(seed=seed)) (Y)
    Y = BatchNormalization() (Y, training=training)

    X = Conv2D(filters=F3, kernel_size=1, strides=s, padding='valid',
kernel_initializer=initializer(seed=seed)) (X)
    X = BatchNormalization() (X, training=training)

    Y = Add() ([Y, X])
    Y = Activation(activation = activation) (Y)

return Y

def ResNet50(input_shape = (512, 512, 3), classes = 2,
activation='relu', seed=0):
    Y = Input(input_shape)
    X = Y
    X = ZeroPadding2D(3)(X)

    X = Conv2D(filters=64, kernel_size = 7, strides = 2,

```

```

kernel_initializer = glorot_uniform(seed=seed))(X)
X = BatchNormalization()(X)
X = Activation(activation=activation)(X)
X = MaxPool2D((3, 3), strides=(2, 2))(X)

X = convolutional_block(X, f = 3, filters = [64, 64, 256], s = 1,
seed=seed)
X = identity_block(X, 3, [64, 64, 256], seed=seed)
X = identity_block(X, 3, [64, 64, 256], seed=seed)

X = convolutional_block(X, f = 3, filters = [128, 128, 512], s =
2, seed=seed)
X = identity_block(X, 3, [128, 128, 512], seed=seed)
X = identity_block(X, 3, [128, 128, 512], seed=seed)
X = identity_block(X, 3, [128, 128, 512], seed=seed)

X = convolutional_block(X, f = 3, filters = [256, 256, 1024], s =
2, seed=seed)
X = identity_block(X, 3, [256, 256, 1024], seed=seed)

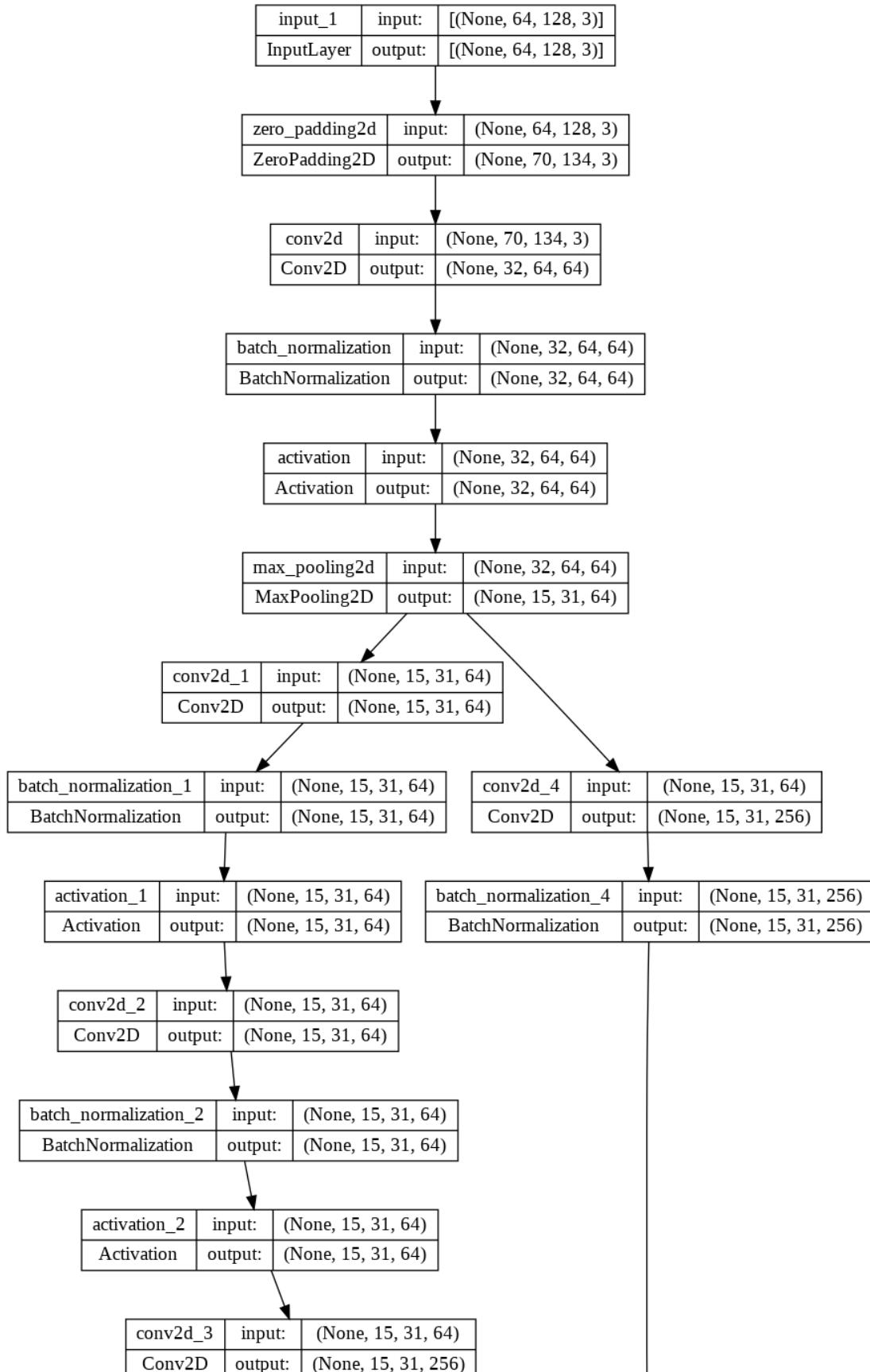
X = convolutional_block(X, f = 3, filters = [512, 512, 2048], s =
2, seed=seed)
X = identity_block(X, 3, [512, 512, 2048], seed=seed)
X = identity_block(X, 3, [512, 512, 2048], seed=seed)

X = AvgPool2D(pool_size=(2, 2))(X)
X = Flatten()(X)
X = Dense(classes, activation='softmax', kernel_initializer =
glorot_uniform(seed=seed))(X)

model = Model(inputs = Y, outputs = X)
return model

plot_model(ResNet50(input_shape=(64, 128, 3)), show_shapes=True)

```



```

from h5py import File
from matplotlib import pyplot
import numpy as np
import os
from tensorflow import one_hot, reshape

from google.colab import drive
drive.mount('/content/drive')

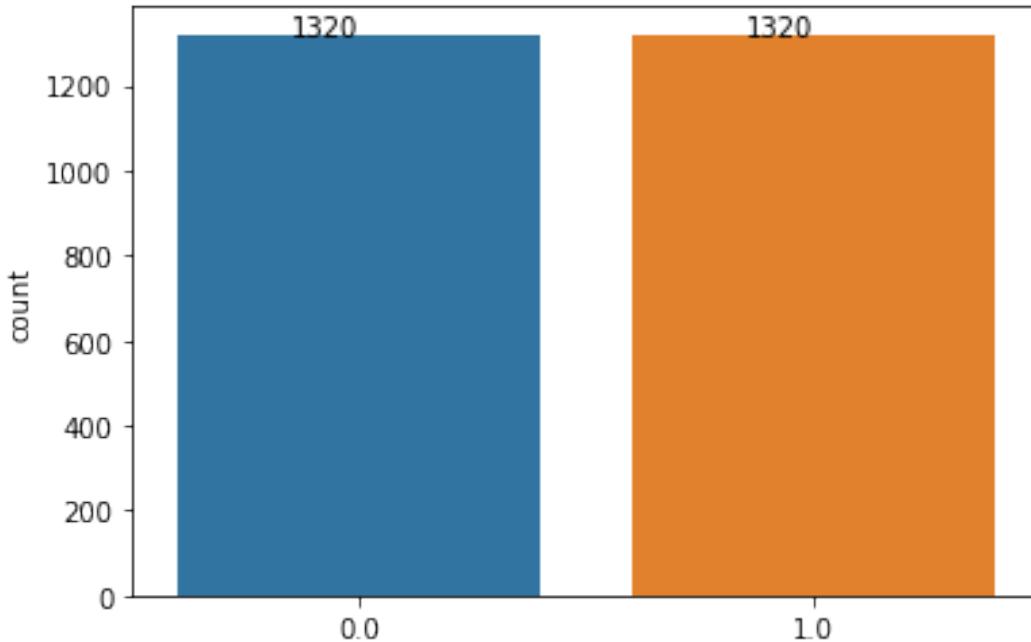
Mounted at /content/drive

# database = input('Database Name: ')
# database = database.lower() + '_128x64.h5'
# file = os.path.join(os.getcwd(), 'database', database)
file = "/content/drive/MyDrive/cedar_128x643.h5"
print(file)
try:
    with File(file, 'r') as hdf:
        X = np.array(hdf.get('X'))
        Y = np.array(hdf.get('Y'))
except Exception as ex:
    template = "An exception of type {0} occurred. Arguments:\n{1!r}"
    message = template.format(type(ex).__name__, ex.args)
    print(message)
X = X / 255.0
Y = Y * 1.0
# Y = one_hot(Y * 1.0, depth=2)
# Y = reshape(Y, (-1, 2))
print("Feature shape =", X.shape)
print("Label shape =", Y.shape)

/content/drive/MyDrive/cedar_128x643.h5
Feature shape = (2640, 64, 128, 3)
Label shape = (2640, 1)

ax = sns.countplot(x = Y.reshape(Y.shape[0]))
for p in ax.patches:
    ax.annotate('{:}'.format(p.get_height()), (p.get_x()+0.25,
p.get_height()+0.01))

```



```
import sklearn
class_weights =
sklearn.utils.class_weight.compute_class_weight(class_weight =
"balanced",classes = np.unique(Y),y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict

{0: 1.0, 1: 1.0}

seed=randint(10)
# metadata['split_seed']=seed
print('seed =', seed)
indices = permutation(X.shape[0])
# print(X)
same = True
if same:
    m = int(0.70 * X.shape[0])
    n = int(0.15 * X.shape[0])
    training_id, validation_id, test_id = indices[:m], indices[m: m + n], indices[m+n:]
    X_train, X_test, X_validate = X[training_id], X[test_id], X[validation_id]
    Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]
else:
    m = int(0.70 * X.shape[0])
    training_id, validation_id = indices[:m], indices[m:]
    X_train, X_validate = X[training_id], X[validation_id]
    Y_train, Y_validate = Y[training_id], Y[validation_id]
```

```

print("Shape of Features in training set =", X_train.shape)
print("Shape of Labels in training set =", Y_train.shape)

def Y,X

seed = 2
Shape of Features in training set = (1847, 64, 128, 3)
Shape of Labels in training set = (1847, 1)

#One hot Encoding
Y_train = one_hot(Y_train, depth=2)
Y_train = reshape(Y_train, (-1, 2))

Y_test = one_hot(Y_test, depth=2)
Y_test = reshape(Y_test, (-1, 2))

Y_validate = one_hot(Y_validate, depth=2)
Y_validate = reshape(Y_validate, (-1, 2))

print("Shape of Labels in training set =", Y_train.shape)

Shape of Labels in training set = (1847, 2)

with tf.device('/device:GPU:0'):
    model = ResNet50(input_shape=(64, 128, 3))
    SGD = tf.keras.optimizers.SGD(learning_rate = 1e-02,momentum = 0.9)
    model.compile(optimizer = SGD, loss='binary_crossentropy',
metrics=['accuracy'])

```

Model Training for 10 epochs

Optimizer - Adam (default hyperparameter)

Epochs - 10

```

with tf.device('/device:GPU:0'):
    history = model.fit(X_train, Y_train, epochs = 10, batch_size = 32,
validation_data=(X_validate, Y_validate),shuffle=True
                      ,class_weight = class_weight_dict)
print('training done')

Epoch 1/10
58/58 [=====] - 26s 127ms/step - loss: 0.5174
- accuracy: 0.8722 - val_loss: 0.0652 - val_accuracy: 0.9773
Epoch 2/10
58/58 [=====] - 5s 83ms/step - loss: 0.0208 -
accuracy: 0.9940 - val_loss: 0.0139 - val_accuracy: 0.9949
Epoch 3/10
58/58 [=====] - 5s 84ms/step - loss: 0.0082 -
accuracy: 0.9957 - val_loss: 0.0356 - val_accuracy: 0.9823

```

```

Epoch 4/10
58/58 [=====] - 5s 84ms/step - loss: 0.0336 - accuracy: 0.9897 - val_loss: 0.0055 - val_accuracy: 0.9975
Epoch 5/10
58/58 [=====] - 5s 85ms/step - loss: 0.0144 - accuracy: 0.9957 - val_loss: 0.0014 - val_accuracy: 1.0000
Epoch 6/10
58/58 [=====] - 5s 85ms/step - loss: 0.0139 - accuracy: 0.9968 - val_loss: 4.0775e-04 - val_accuracy: 1.0000
Epoch 7/10
58/58 [=====] - 5s 89ms/step - loss: 7.6360e-04 - accuracy: 1.0000 - val_loss: 1.7100e-04 - val_accuracy: 1.0000
Epoch 8/10
58/58 [=====] - 5s 83ms/step - loss: 0.0041 - accuracy: 0.9995 - val_loss: 2.5264e-04 - val_accuracy: 1.0000
Epoch 9/10
58/58 [=====] - 5s 83ms/step - loss: 9.3196e-04 - accuracy: 1.0000 - val_loss: 3.0111e-04 - val_accuracy: 1.0000
Epoch 10/10
58/58 [=====] - 6s 100ms/step - loss: 0.0020 - accuracy: 0.9995 - val_loss: 3.2433e-04 - val_accuracy: 1.0000
training done

import pickle
folder = "/content/drive/MyDrive/CedarResNetEPOCHS10/"

with open(folder + 'trainHistoryDict', 'wb') as file_pi:
    pickle.dump(history, file_pi)
model.save(folder + 'model.h5')

WARNING:absl:Found untraced functions such as
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op while saving (showing 5 of 53). These
functions will not be directly callable after loading.

history = pickle.load(open(folder + 'trainHistoryDict', "rb"))
model = tf.keras.models.load_model(folder + 'model.h5')

```

Plotting

Optimizer - Adam (default hyperparameter)

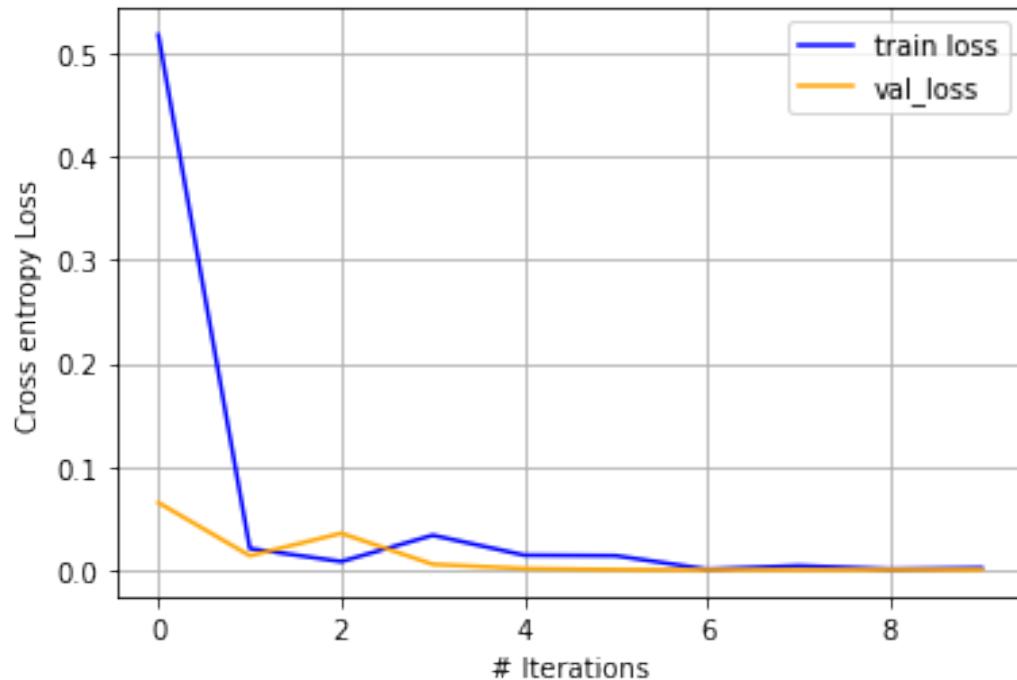
Epochs - 10

```

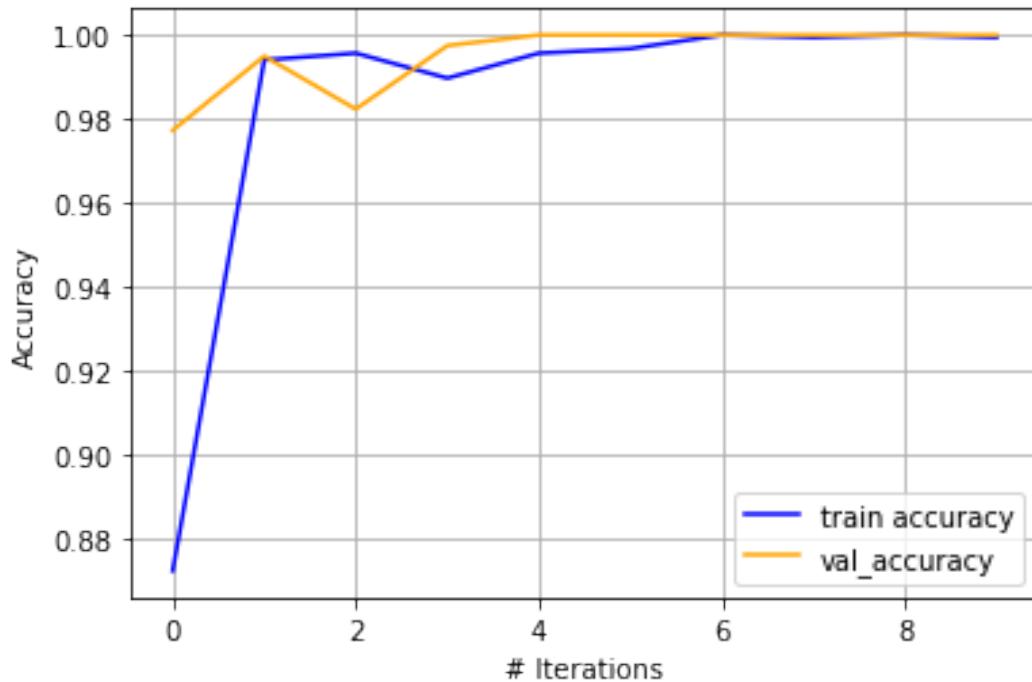
plt.xlabel('# Iterations')
plt.ylabel('Cross entropy Loss')
plt.plot(history.history['loss'], color='blue',label = "train loss")
plt.plot(history.history['val_loss'], color='orange',label = "val_loss")

```

```
plt.legend()  
plt.grid()  
plt.savefig(folder + "loss.jpeg")  
plt.show()
```

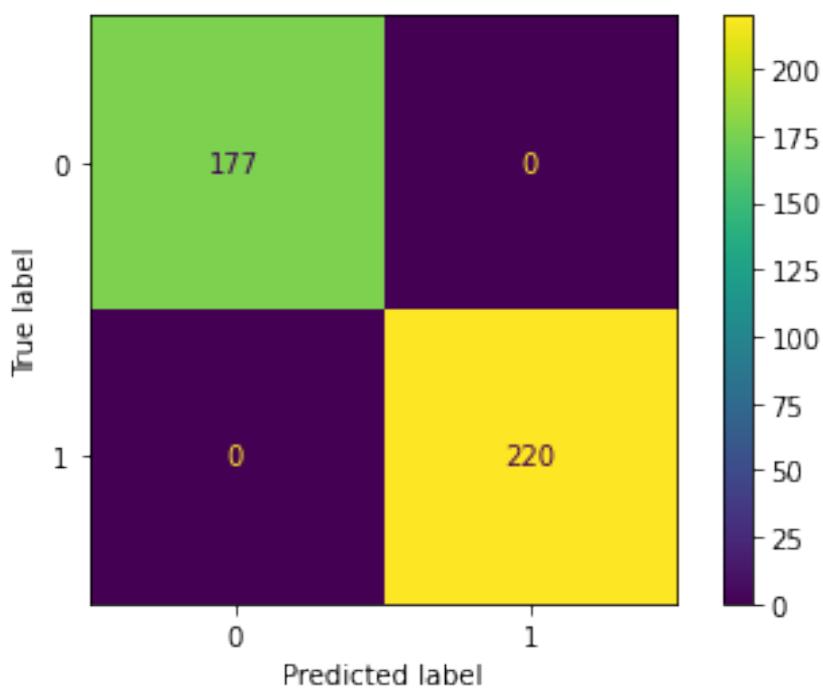


```
plt.xlabel('# Iterations')  
plt.ylabel('Accuracy')  
plt.plot(history.history['accuracy'], color='blue',label = "train accuracy")  
plt.plot(history.history['val_accuracy'], color='orange',label = "val_accuracy")  
plt.grid()  
plt.legend()  
plt.savefig(folder + "accuracy.jpeg")  
plt.show()
```



```
ConfusionMatrixDisplay(confusion_matrix(tf.argmax(Y_test, axis = 1),  
tf.argmax(model.predict(X_test), axis = 1))).plot()  
plt.savefig( folder + "confusionmatrix.jpeg")  
plt.show()
```

13/13 [=====] - 2s 59ms/step

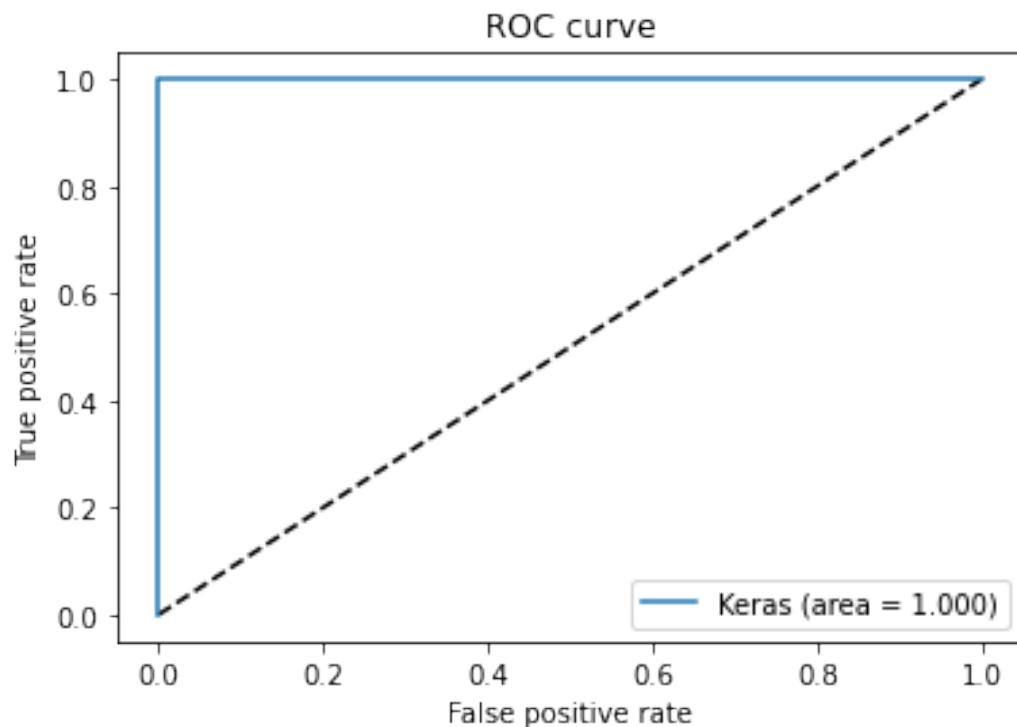


```

from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = tf.argmax(model.predict(X_test), axis = 1)
fpr_keras, tpr_keras, thresholds_keras =
roc_curve(tf.argmax(Y_test, axis = 1), y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.jpeg")
plt.show()

```

13/13 [=====] - 0s 24ms/step



Model Training for 100 epochs

Optimizer - SGD ($\text{lr} = 0.01$)

Epochs - 100 (Early Stopping)

```
import pickle
folder = "/content/drive/MyDrive/CedarResNetEPOCHS100/"

with tf.device('/device:GPU:0'):
    callbacks = [
        EarlyStopping(patience = 5, verbose = 1),
        ModelCheckpoint(folder + 'model-{epoch:03d}.h5',
                        verbose=1, save_best_only = True)
    ]

    with tf.device('/device:GPU:0'):
        history = model.fit(X_train, Y_train, epochs = 100, batch_size = 32,
                             validation_data=(X_validate, Y_validate), shuffle=True
                             , class_weight = class_weight_dict, callbacks =
        callbacks)
    print('training done')

Epoch 1/100
58/58 [=====] - ETA: 0s - loss: 0.8786 -
accuracy: 0.8581
Epoch 1: val_loss improved from inf to 0.13165, saving model to
/content/drive/MyDrive/CedarResNetEPOCHS100/model-001.h5
58/58 [=====] - 12s 119ms/step - loss: 0.8786
- accuracy: 0.8581 - val_loss: 0.1317 - val_accuracy: 0.9823
Epoch 2/100
58/58 [=====] - ETA: 0s - loss: 0.0965 -
accuracy: 0.9881
Epoch 2: val_loss improved from 0.13165 to 0.01450, saving model to
/content/drive/MyDrive/CedarResNetEPOCHS100/model-002.h5
58/58 [=====] - 6s 102ms/step - loss: 0.0965
- accuracy: 0.9881 - val_loss: 0.0145 - val_accuracy: 0.9949
Epoch 3/100
58/58 [=====] - ETA: 0s - loss: 0.0391 -
accuracy: 0.9908
Epoch 3: val_loss improved from 0.01450 to 0.01326, saving model to
/content/drive/MyDrive/CedarResNetEPOCHS100/model-003.h5
58/58 [=====] - 6s 103ms/step - loss: 0.0391
- accuracy: 0.9908 - val_loss: 0.0133 - val_accuracy: 0.9975
Epoch 4/100
58/58 [=====] - ETA: 0s - loss: 0.0289 -
accuracy: 0.9968
Epoch 4: val_loss did not improve from 0.01326
58/58 [=====] - 5s 84ms/step - loss: 0.0289 -
accuracy: 0.9968 - val_loss: 0.0296 - val_accuracy: 0.9924
Epoch 5/100
58/58 [=====] - ETA: 0s - loss: 0.0302 -
accuracy: 0.9962
Epoch 5: val_loss improved from 0.01326 to 0.00035, saving model to
/content/drive/MyDrive/CedarResNetEPOCHS100/model-005.h5
58/58 [=====] - 6s 105ms/step - loss: 0.0302
```

```
- accuracy: 0.9962 - val_loss: 3.4722e-04 - val_accuracy: 1.0000
Epoch 6/100
58/58 [=====] - ETA: 0s - loss: 0.0220 - accuracy: 0.9946
Epoch 6: val_loss did not improve from 0.00035
58/58 [=====] - 5s 84ms/step - loss: 0.0220 - accuracy: 0.9946 - val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 7/100
58/58 [=====] - ETA: 0s - loss: 0.0159 - accuracy: 0.9968
Epoch 7: val_loss did not improve from 0.00035
58/58 [=====] - 5s 85ms/step - loss: 0.0159 - accuracy: 0.9968 - val_loss: 0.0429 - val_accuracy: 0.9924
Epoch 8/100
58/58 [=====] - ETA: 0s - loss: 0.0133 - accuracy: 0.9968
Epoch 8: val_loss improved from 0.00035 to 0.00024, saving model to /content/drive/MyDrive/CedarResNetEPOCHS100/model-008.h5
58/58 [=====] - 6s 104ms/step - loss: 0.0133 - accuracy: 0.9968 - val_loss: 2.3619e-04 - val_accuracy: 1.0000
Epoch 9/100
58/58 [=====] - ETA: 0s - loss: 5.8528e-04 - accuracy: 1.0000
Epoch 9: val_loss improved from 0.00024 to 0.00004, saving model to /content/drive/MyDrive/CedarResNetEPOCHS100/model-009.h5
58/58 [=====] - 6s 105ms/step - loss: 5.8528e-04 - accuracy: 1.0000 - val_loss: 4.2090e-05 - val_accuracy: 1.0000
Epoch 10/100
58/58 [=====] - ETA: 0s - loss: 0.0012 - accuracy: 0.9989
Epoch 10: val_loss did not improve from 0.00004
58/58 [=====] - 5s 83ms/step - loss: 0.0012 - accuracy: 0.9989 - val_loss: 4.7575e-05 - val_accuracy: 1.0000
Epoch 11/100
58/58 [=====] - ETA: 0s - loss: 0.0016 - accuracy: 0.9989
Epoch 11: val_loss did not improve from 0.00004
58/58 [=====] - 5s 88ms/step - loss: 0.0016 - accuracy: 0.9989 - val_loss: 9.0890e-05 - val_accuracy: 1.0000
Epoch 12/100
58/58 [=====] - ETA: 0s - loss: 5.0120e-04 - accuracy: 1.0000
Epoch 12: val_loss did not improve from 0.00004
58/58 [=====] - 5s 83ms/step - loss: 5.0120e-04 - accuracy: 1.0000 - val_loss: 1.0217e-04 - val_accuracy: 1.0000
Epoch 13/100
58/58 [=====] - ETA: 0s - loss: 3.6526e-04 - accuracy: 1.0000
```

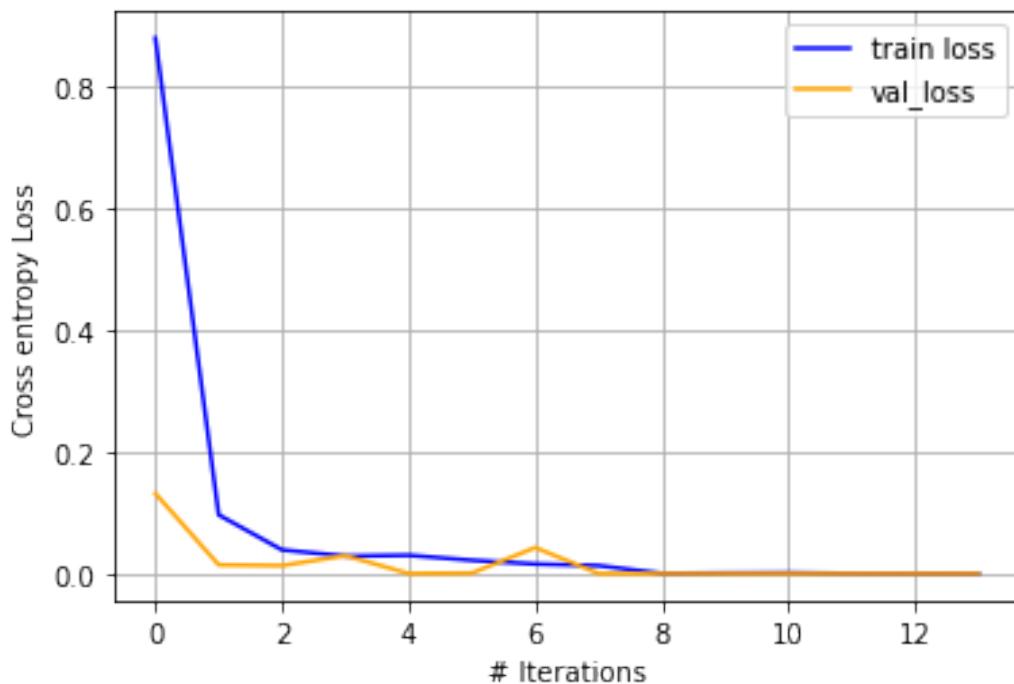
```
Epoch 13: val_loss did not improve from 0.00004
58/58 [=====] - 5s 83ms/step - loss: 3.6526e-04 - accuracy: 1.0000 - val_loss: 5.0152e-05 - val_accuracy: 1.0000
Epoch 14/100
58/58 [=====] - ETA: 0s - loss: 3.6564e-05 - accuracy: 1.0000
Epoch 14: val_loss did not improve from 0.00004
58/58 [=====] - 5s 82ms/step - loss: 3.6564e-05 - accuracy: 1.0000 - val_loss: 4.4532e-05 - val_accuracy: 1.0000
Epoch 14: early stopping
training done
```

Plotting

Optimizer - SGD ($\text{lr} = 0.01$)

Epochs - 100 (Early Stopping)

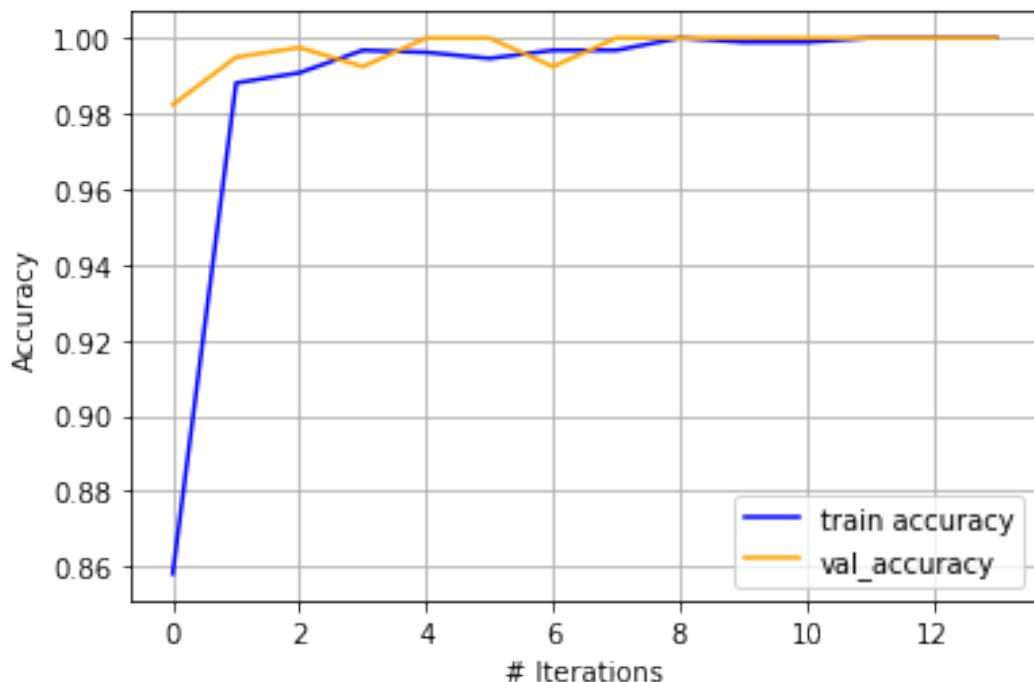
```
plt.xlabel('# Iterations')
plt.ylabel('Cross entropy Loss')
plt.plot(history.history['loss'], color='blue', label = "train loss")
plt.plot(history.history['val_loss'], color='orange', label = "val_loss")
plt.legend()
plt.grid()
plt.savefig(folder + "loss.jpeg")
plt.show()
```



```

plt.xlabel('# Iterations')
plt.ylabel('Accuracy')
plt.plot(history.history['accuracy'], color='blue',label = "train accuracy")
plt.plot(history.history['val_accuracy'], color='orange',label = "val_accuracy")
plt.grid()
plt.legend()
plt.savefig(folder + "accuracy.jpeg")
plt.show()

```

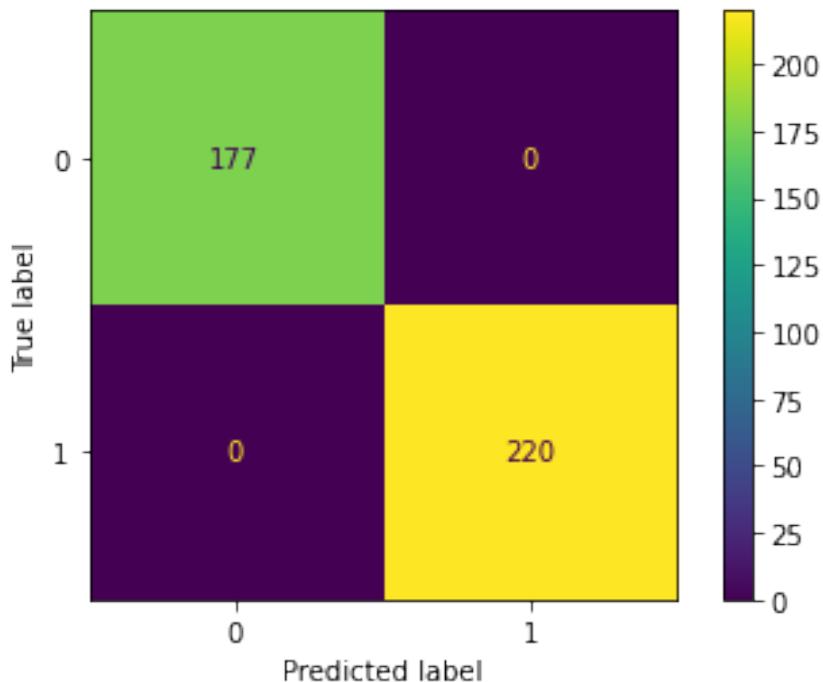


```

ConfusionMatrixDisplay(confusion_matrix(tf.argmax(Y_test,axis = 1),
tf.argmax(model.predict(X_test),axis = 1))).plot()
plt.savefig( folder + "confusionmatrix.jpeg")
plt.show()

```

13/13 [=====] - 1s 24ms/step

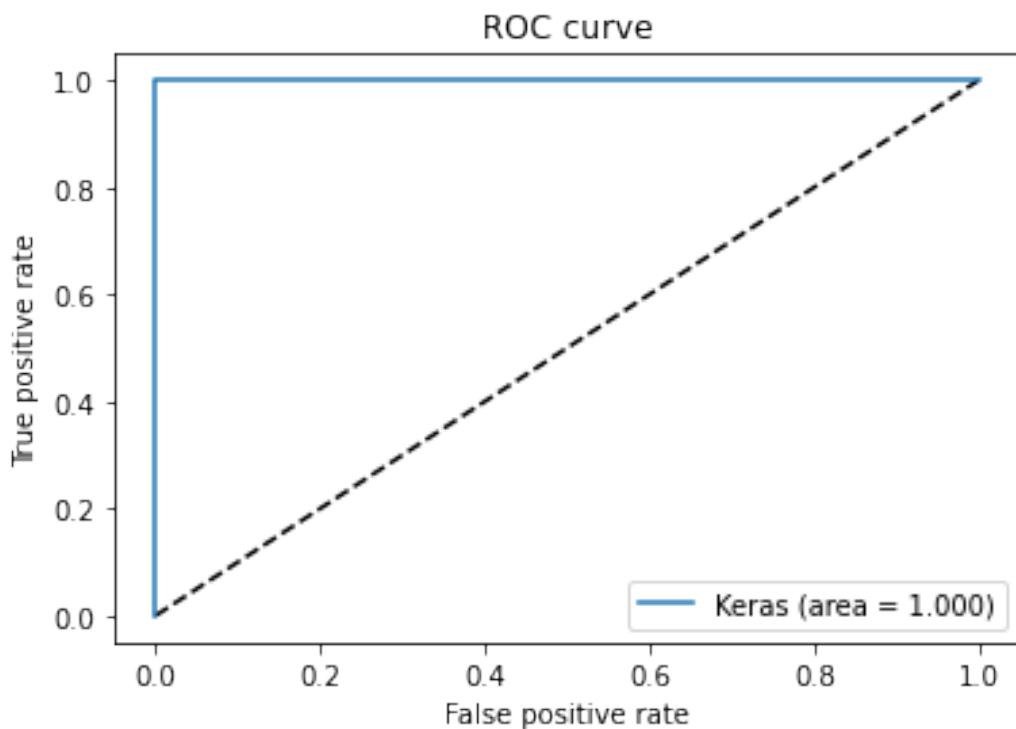


```

from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = tf.argmax(model.predict(X_test), axis = 1)
fpr_keras, tpr_keras, thresholds_keras =
roc_curve(tf.argmax(Y_test, axis = 1), y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.jpeg")
plt.show()

```

13/13 [=====] - 0s 24ms/step



Model4_Resnet50Siamese_BhSig260HindiTraining

November 14, 2024

```
#Model 4 (ResNet50 + Siamese)
```

```
[ ]: from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
import os
from tensorflow.keras.initializers import RandomNormal, Constant
import tensorflow as tf
from tensorflow.keras.backend import max, mean, sqrt, square, sum

import seaborn as sns
from tensorflow.keras.optimizers import Adam,RMSprop
from keras.models import Model

from tensorflow import keras
from keras.layers import LeakyReLU, Softmax
from keras.layers import Conv2D, Activation, Input,Dropout,Lambda,Flatten, Dense
from keras.layers import Dense, Flatten, Reshape, Activation
from keras.layers import BatchNormalization ,ZeroPadding2D

from keras.layers import MaxPooling2D
from keras.layers import Concatenate
from keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
from keras.layers import Layer, InputSpec
from keras.regularizers import l2
from keras import backend as K
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from tensorflow import one_hot, reshape

from tensorflow.keras.layers import Activation, Add, AvgPool2D, ↴
    ↴BatchNormalization, Conv2D, Dense, Flatten, Input, MaxPool2D, ZeroPadding2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.initializers import random_uniform, glorot_uniform, ↴
    ↴constant
from tensorflow.keras.utils import plot_model
```

```

from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model

```

```
[ ]: import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
[ ]: from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[ ]: with tf.device('/device:GPU:0'):
    colab = True
    # database = 'bigsig260_224x224_siamese_preprocessed.h5'
    if colab:
        # from google.colab import drive
        # drive.mount('/content/gdrive')
        file = '/content/drive/MyDrive/bigsig260155x220x1_siamese_preprocessed.h5'
    else:
        file = os.path.join(os.getcwd(), 'RoboticLab', database)
```

```

print(file)
with File(file, 'r') as hdf:
    S1 = np.array(hdf.get('S1'))
    S2 = np.array(hdf.get('S2'))
    Y = np.array(hdf.get('Y'))
print(S1.shape)
print(S2.shape)
print(Y.shape)

```

```
/content/drive/MyDrive/bigsig260155x220x1_siamese_preprocessed.h5
(29880, 155, 220, 1)
(29880, 155, 220, 1)
(29880, 1)
```

```
[ ]: res_net = ResNet18(input_shape = (155, 220, 1),input_tensor = S1[1]
,weights='imagenet',classes = 2)
```

```
-----
AttributeError Traceback (most recent call last)
<ipython-input-13-c2a6c5eebc21> in <module>
----> 1 res_net = ResNet18(input_shape = (155, 220, 1),input_tensor = S1[1]
,weights='imagenet',classes = 2)

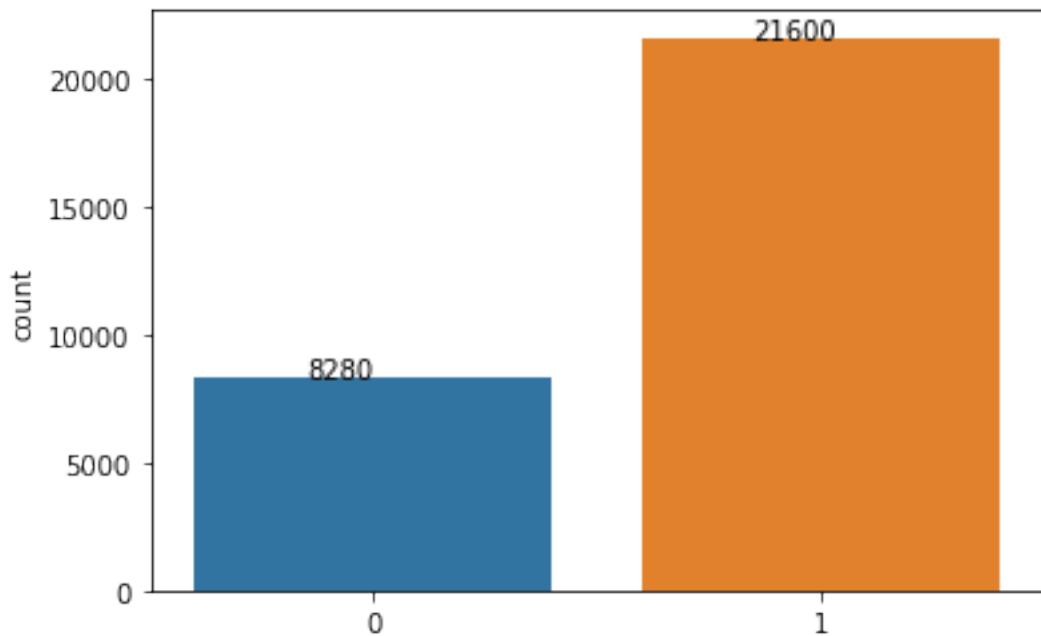
/content/classification_models/models/resnet.py in ResNet18(input_shape,
   input_tensor, weights, classes, include_top, **kwargs)
    300
    301 def ResNet18(input_shape=None, input_tensor=None, weights=None,
   classes=1000, include_top=True, **kwargs):
--> 302     return ResNet(
    303         MODELS_PARAMS['resnet18'],
    304         input_shape=input_shape,

/content/classification_models/models/resnet.py in ResNet(model_params,
   input_shape, input_tensor, include_top, classes, weights, **kwargs)
    210         img_input = layers.Input(shape=input_shape, name='data')
    211     else:
--> 212         if not backend.is_keras_tensor(input_tensor):
    213             img_input = layers.Input(tensor=input_tensor,
   shape=input_shape)
    214     else:

AttributeError: 'NoneType' object has no attribute 'is_keras_tensor'
```

```
[ ]: ax = sns.countplot(x = Y.reshape(Y.shape[0]))
for p in ax.patches:
```

```
    ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x() + 0.25, p.get_height() + 0.01))
```



```
[ ]: import sklearn
class_weights = sklearn.utils.class_weight.compute_class_weight(class_weight = "balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict
```

```
[ ]: {0: 1.8043478260869565, 1: 0.6916666666666667}
```

```
[ ]: Y = Y/1.0
```

```
[ ]: S1.dtype
```

```
[ ]: dtype('uint8')
```

```
[ ]: with tf.device('/device:GPU:0'):
    seed=randint(10)
    print('seed='+str(seed))
    indices = permutation(Y.shape[0])
    m = int(0.70 * Y.shape[0])
    n = int(0.15 * Y.shape[0])
    training_id, validation_id, test_id = indices[:m], indices[m: m + n], indices[m+n:]
```

```

S1_train, S1_test, S1_validate = S1[training_id], S1[test_id],  

    ↪S1[validation_id]  

S2_train, S2_test, S2_validate = S2[training_id], S2[test_id],  

    ↪S2[validation_id]  

Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]  

print(S1_train.shape)  

print(S2_train.shape)  

print(Y_train.shape)

del S1,S2,Y

```

```

seed=4  

(20916, 155, 220, 1)  

(20916, 155, 220, 1)  

(20916, 1)

```

```

[ ]: fig = plt.figure(figsize = (16,16))
rows,cols = 4,4
i = 1
while(i < range(1,rows*cols)):
    random_idx = randint(0,len(S1_train))
    if(Y_train[random_idx] == 0):
        Label = "Genuine Pair"
    else:
        Label = "Forged Pair"
    img1 = S1_train[random_idx]
    img2 = S2_train[random_idx]
    fig.add_subplot(rows,cols,i)
    plt.imshow(img1.squeeze(),cmap = "gray");
    plt.axis(False);
    i += 1
    fig.add_subplot(rows,cols,i)
    plt.imshow(img2.squeeze(),cmap = "gray");
    plt.axis(False);
    plt.text(0.5, 0.5, Label,  

    ↪horizontalalignment='center',verticalalignment='center',fontsize=15,fontweight  

    ↪= 1000);
    i += 1

```

Forged Pair	Forged Pair
परमिंद्र सिंह	परमिंद्र बिंदु
मानेत धनेश	मनित धनेश

Forged Pair	Genuine Pair
सिमरण जीत कौर	सिमरण जीत कौर
विशाल गुप्ता	विशाल गुप्ता

Forged Pair	Forged Pair
जीत पट्टनाथक	जीत पट्टनाथक
विवेणीप कौर	विवेणीप कौर

Forged Pair	Forged Pair
विशाल गुप्ता	विशाल गुप्ता
दिपालवीत साधनी	दिपालवीत साधनी

```
[ ]: # #One hot Encoding
# Y_train = one_hot(Y_train, depth=2)
# Y_train = reshape(Y_train, (-1, 2))

# Y_test = one_hot(Y_test, depth=2)
# Y_test = reshape(Y_test, (-1, 2))

# Y_validate = one_hot(Y_validate, depth=2)
# Y_validate = reshape(Y_validate, (-1, 2))
```

```
[ ]: def identity_block(X, f, filters, training=True, initializer=random_uniform,
activation='relu', seed=0):
    F1, F2, F3 = filters
    Y = X
```

```

Y = Conv2D(filters=F1, kernel_size=1, strides=1, padding='valid',  

kernel_initializer=initializer(seed=seed)) (Y)  

Y = BatchNormalization() (Y, training=training)  

Y = Activation(activation = activation) (Y)  
  

Y = Conv2D(filters=F2, kernel_size=f, strides=1, padding='same',  

kernel_initializer=initializer(seed=seed)) (Y)  

Y = BatchNormalization() (Y, training=training)  

Y = Activation(activation = activation) (Y)  
  

Y = Conv2D(filters=F3, kernel_size=1, strides=1, padding='valid',  

kernel_initializer=initializer(seed=seed)) (Y)  

Y = BatchNormalization() (Y, training=training)  
  

Y = Add() ([Y, X])  

Y = Activation(activation = activation) (Y)  
  

return Y

```

```

[ ]: def convolutional_block(X, f, filters, s=2, training=True,  

kernel_initializer=glorot_uniform, activation='relu', seed=0):  

    F1, F2, F3 = filters  

    Y = X  
  

    Y = Conv2D(filters=F1, kernel_size=1, strides=s, padding='valid',  

kernel_initializer=initializer(seed=seed)) (Y)  

    Y = BatchNormalization() (Y, training=training)  

    Y = Activation(activation = activation) (Y)  
  

    Y = Conv2D(filters=F2, kernel_size=f, strides=1, padding='same',  

kernel_initializer=initializer(seed=seed)) (Y)  

    Y = BatchNormalization() (Y, training=training)  

    Y = Activation(activation = activation) (Y)  
  

    Y = Conv2D(filters=F3, kernel_size=1, strides=1, padding='valid',  

kernel_initializer=initializer(seed=seed)) (Y)  

    Y = BatchNormalization() (Y, training=training)  
  

    X = Conv2D(filters=F3, kernel_size=1, strides=s, padding='valid',  

kernel_initializer=initializer(seed=seed)) (X)  

    X = BatchNormalization() (X, training=training)  
  

    Y = Add() ([Y, X])  

    Y = Activation(activation = activation) (Y)  
  

return Y

```

```
[ ]: def ResNet50(input_shape = (512, 512, 3), classes = 2, activation='relu', u
↳seed=0):
    Y = Input(input_shape)
    X = Y
    X = ZeroPadding2D(3)(X)

    X = Conv2D(filters=64, kernel_size = 7, strides = 2, kernel_initializer = u
↳glorot_uniform(seed=seed))(X)
    X = BatchNormalization()(X)
    X = Activation(activation=activation)(X)
    X = MaxPool2D((3, 3), strides=(2, 2))(X)

    X = convolutional_block(X, f = 3, filters = [64, 64, 256], s = 1, seed=seed)
    X = identity_block(X, 3, [64, 64, 256], seed=seed)
    X = identity_block(X, 3, [64, 64, 256], seed=seed)

    X = convolutional_block(X, f = 3, filters = [128, 128, 512], s = 2, u
↳seed=seed)
    X = identity_block(X, 3, [128, 128, 512], seed=seed)
    X = identity_block(X, 3, [128, 128, 512], seed=seed)
    X = identity_block(X, 3, [128, 128, 512], seed=seed)

    X = convolutional_block(X, f = 3, filters = [256, 256, 1024], s = 2, u
↳seed=seed)
    X = identity_block(X, 3, [256, 256, 1024], seed=seed)
    X = identity_block(X, 3, [256, 256, 1024], seed=seed)
    X = identity_block(X, 3, [256, 256, 1024], seed=seed)
    X = identity_block(X, 3, [256, 256, 1024], seed=seed)

    X = convolutional_block(X, f = 3, filters = [512, 512, 2048], s = 2, u
↳seed=seed)
    X = identity_block(X, 3, [512, 512, 2048], seed=seed)
    X = identity_block(X, 3, [512, 512, 2048], seed=seed)

    X = AvgPool2D(pool_size=(2, 2))(X)
    X = Flatten()(X)
    X = Dense(classes, activation='sigmoid', kernel_initializer = u
↳glorot_uniform(seed=seed))(X)

    model = Model(inputs = Y, outputs = X)
    return model
```

```
[ ]: input_shape=(155, 220,1)
      input_shape
```

```
[ ]: (155, 220, 1)
```

```
[ ]: plot_model(ResNet50(input_shape = input_shape), show_shapes=True)

[ ]: with tf.device('/device:GPU:0'):
    inputShape = S1_train.shape[1:]
    f = ResNet50(inputShape)

[ ]: def euclidean_distance(S):
    X = S['S1']
    Y = S['S2']
    return K.sqrt(K.maximum(K.sum(K.square(X - Y), axis=1, keepdims=True), K.
                           epsilon()))

[ ]: input_a = Input(shape=inputShape, name = 'S1')
input_b = Input(shape=inputShape, name = 'S2')
encoded_s1 = f(input_a)
encoded_s2 = f(input_b)
distance = Lambda(euclidean_distance)({'S1': encoded_s1, 'S2': encoded_s2})
outputs = Dense(1, activation="sigmoid")(distance)
model = Model(inputs = [input_a, input_b], outputs = outputs)

[ ]: def constructive_loss(y_true, y_pred):
    margin = 1
    y_true = tf.cast(y_true, y_pred.dtype)
    squaredPreds = K.square(y_pred)
    squaredMargin = K.square(K.maximum(margin - y_pred, 0))
    loss = K.mean(y_true * squaredMargin + (1 - y_true) * squaredPreds)
    return loss
```

0.1 HYPER PARAMETER TUNING

0.2 BATCH SIZE TUNING

```
[ ]: # with tf.device('/device:GPU:0'):
#   for i in np.arange(0.0010,0.0101,0.0010):
#     adam = Adam(learning_rate = 0.001,epsilon = 1e-08)
#     model.compile(loss = constructive_loss, optimizer = adam,metrics=['BinaryAccuracy'])
#     print("For Learning Rate = {LR}".format(LR = i))

[ ]: with tf.device('/device:GPU:0'):
    adam = Adam(learning_rate = 0.001,epsilon = 1e-08)
    model.compile(loss = constructive_loss, optimizer = adam,metrics=['BinaryAccuracy'])

[ ]: with tf.device('/device:GPU:0'):
    batch_Size = 8
    print("For batch Size : ",batch_Size)
    model.fit(x = [S1_train, S2_train],y = Y_train,
```

```

        validation_data = u
        ([S1_validate,S2_validate],Y_validate),
            epochs = 1,batch_size = batch_Size,class_weight = u
        class_weight_dict,shuffle = True
    )
)

```

For batch Size : 8
2802/2802 [=====] - 213s 72ms/step - loss: 0.0934 -
binary_accuracy: 0.9104 - val_loss: 0.0813 - val_binary_accuracy: 0.9186

```
[ ]: with tf.device('/device:GPU:0'):
    batch_Size = 16
    print("For batch Size : ",batch_Size)
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = u
               ([S1_validate,S2_validate],Y_validate),
                   epochs = 1,batch_size = batch_Size,class_weight = u
               class_weight_dict,shuffle = True
    )
)
```

For batch Size : 16
1401/1401 [=====] - 160s 107ms/step - loss: 0.0904 -
binary_accuracy: 0.9103 - val_loss: 0.0883 - val_binary_accuracy: 0.9116

```
[ ]: with tf.device('/device:GPU:0'):
    batch_Size = 32
    print("For batch Size : ",batch_Size)
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = u
               ([S1_validate,S2_validate],Y_validate),
                   epochs = 1,batch_size = batch_Size,class_weight = u
               class_weight_dict,shuffle = True
    )
)
```

For batch Size : 32
701/701 [=====] - 132s 172ms/step - loss: 0.0979 -
binary_accuracy: 0.9093 - val_loss: 0.0795 - val_binary_accuracy: 0.9205

```
[ ]: with tf.device('/device:GPU:0'):
    batch_Size = 64
    print("For batch Size : ",batch_Size)
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = u
               ([S1_validate,S2_validate],Y_validate),
                   epochs = 1,batch_size = batch_Size,class_weight = u
               class_weight_dict,shuffle = True
    )
)
```

For batch Size : 64

```
351/351 [=====] - 120s 311ms/step - loss: 0.0908 -  
binary_accuracy: 0.9108 - val_loss: 0.0768 - val_binary_accuracy: 0.9232
```

```
[ ]: with tf.device('/device:GPU:0'):  
    batch_Size = 128  
    print("For batch Size : ",batch_Size)  
    model.fit(x = [S1_train, S2_train],y = Y_train,  
              validation_data =[  
    ([S1_validate,S2_validate],Y_validate),  
    epochs = 1,batch_size = batch_Size,class_weight =  
    class_weight_dict,shuffle = True  
    )
```

For batch Size : 128

```
176/176 [=====] - 118s 605ms/step - loss: 0.0928 -  
binary_accuracy: 0.9084 - val_loss: 0.0779 - val_binary_accuracy: 0.9221
```

THE BEST ACCURACY IS FOR BATCH SIZE = 64

0.3 LEARNING RATE

```
[ ]: batch_Size = 32
```

```
[ ]: with tf.device('/device:GPU:0'):  
    lr = 1e-03  
    print("For learning Rate = ",lr)  
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)  
    model.compile(loss = constructive_loss, optimizer = rms,  
    metrics=['BinaryAccuracy'])  
    model.fit(x = [S1_train, S2_train],y = Y_train,  
              validation_data =[  
    ([S1_validate,S2_validate],Y_validate),  
    epochs = 5,batch_size = batch_Size,class_weight =  
    class_weight_dict,shuffle = True  
    )
```

For learning Rate = 0.001

Epoch 1/5

```
654/654 [=====] - 327s 463ms/step - loss: 0.2045 -  
binary_accuracy: 0.7242 - val_loss: 0.2017 - val_binary_accuracy: 0.7200
```

Epoch 2/5

```
654/654 [=====] - 301s 460ms/step - loss: 0.2020 -  
binary_accuracy: 0.7248 - val_loss: 0.2018 - val_binary_accuracy: 0.7200
```

Epoch 3/5

```
654/654 [=====] - 321s 491ms/step - loss: 0.2021 -  
binary_accuracy: 0.7248 - val_loss: 0.2016 - val_binary_accuracy: 0.7200
```

Epoch 4/5

```
654/654 [=====] - 301s 460ms/step - loss: 0.2023 -  
binary_accuracy: 0.7248 - val_loss: 0.2017 - val_binary_accuracy: 0.7200
```

```
Epoch 5/5
654/654 [=====] - 301s 460ms/step - loss: 0.2022 -
binary_accuracy: 0.7248 - val_loss: 0.2016 - val_binary_accuracy: 0.7200
```

```
[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-04
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
    ↵metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = [
    ↵([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight =
    ↵class_weight_dict,shuffle = True
            )
```

```
For learning Rate =  0.0001
Epoch 1/5
654/654 [=====] - 36s 39ms/step - loss: 0.2232 -
binary_accuracy: 0.7001 - val_loss: 0.2445 - val_binary_accuracy: 0.7129
Epoch 2/5
654/654 [=====] - 25s 38ms/step - loss: 0.2047 -
binary_accuracy: 0.7146 - val_loss: 0.2358 - val_binary_accuracy: 0.7140
Epoch 3/5
654/654 [=====] - 25s 38ms/step - loss: 0.2008 -
binary_accuracy: 0.7189 - val_loss: 0.2318 - val_binary_accuracy: 0.7115
Epoch 4/5
654/654 [=====] - 24s 37ms/step - loss: 0.1992 -
binary_accuracy: 0.7188 - val_loss: 0.2352 - val_binary_accuracy: 0.7146
Epoch 5/5
654/654 [=====] - 24s 37ms/step - loss: 0.1935 -
binary_accuracy: 0.7259 - val_loss: 0.2279 - val_binary_accuracy: 0.7149
```

```
[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-05
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
    ↵metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = [
    ↵([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight =
    ↵class_weight_dict,shuffle = True
            )
```

```
For learning Rate =  1e-05
```

```

Epoch 1/5
654/654 [=====] - 33s 35ms/step - loss: 0.2488 -
binary_accuracy: 0.7070 - val_loss: 0.2451 - val_binary_accuracy: 0.7135
Epoch 2/5
654/654 [=====] - 22s 34ms/step - loss: 0.1971 -
binary_accuracy: 0.7212 - val_loss: 0.2483 - val_binary_accuracy: 0.7135
Epoch 3/5
654/654 [=====] - 23s 35ms/step - loss: 0.1953 -
binary_accuracy: 0.7240 - val_loss: 0.2338 - val_binary_accuracy: 0.7135
Epoch 4/5
654/654 [=====] - 22s 34ms/step - loss: 0.1947 -
binary_accuracy: 0.7232 - val_loss: 0.2389 - val_binary_accuracy: 0.7124
Epoch 5/5
654/654 [=====] - 22s 34ms/step - loss: 0.1917 -
binary_accuracy: 0.7255 - val_loss: 0.2385 - val_binary_accuracy: 0.7135

```

```
[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-06
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
    ↪metrics=['BinaryAccuracy' ])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = ↪
    ↪([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight = ↪
    ↪class_weight_dict,shuffle = True
               )
```

```

For learning Rate =  1e-06
Epoch 1/5
654/654 [=====] - 26s 36ms/step - loss: 0.2219 -
binary_accuracy: 0.6782 - val_loss: 0.2373 - val_binary_accuracy: 0.7160
Epoch 2/5
654/654 [=====] - 22s 34ms/step - loss: 0.1969 -
binary_accuracy: 0.7241 - val_loss: 0.2315 - val_binary_accuracy: 0.7160
Epoch 3/5
654/654 [=====] - 22s 34ms/step - loss: 0.1910 -
binary_accuracy: 0.7257 - val_loss: 0.2267 - val_binary_accuracy: 0.7160
Epoch 4/5
654/654 [=====] - 25s 39ms/step - loss: 0.1865 -
binary_accuracy: 0.7283 - val_loss: 0.2198 - val_binary_accuracy: 0.7160
Epoch 5/5
202/654 [=====>...] - ETA: 14s - loss: 0.1819 -
binary_accuracy: 0.7333

```

KeyboardInterrupt

Traceback (most recent call last)

```

<ipython-input-18-447822a95911> in <module>
      6     model.fit(x = [S1_train, S2_train],y = Y_train,
      7                 validation_data =_
<__main__.tuple object at 0x7f3e0d0000>,
--> 8                     epochs = 5,batch_size = batch_Size,class_weight_
      9             = class_weight_dict,shuffle = True
      9             )

/usr/local/lib/python3.7/dist-packages/keras/utils/traceback_utils.py in_
<__main__.error_handler(*args, **kwargs)>
      62     filtered_tb = None
      63     try:
--> 64         return fn(*args, **kwargs)
      65     except Exception as e: # pylint: disable=broad-except
      66         filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.7/dist-packages/keras/engine/training.py in fit(self, x,_
      67     y, batch_size, epochs, verbose, callbacks, validation_split, validation_data,_
      68     shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch,_
      69     validation_steps, validation_batch_size, validation_freq, max_queue_size,_
      70     workers, use_multiprocessing)
1412             logs = tmp_logs # No error, now safe to assign to logs.
1413             end_step = step + data_handler.step_increment
-> 1414             callbacks.on_train_batch_end(end_step, logs)
1415             if self.stop_training:
1416                 break

/usr/local/lib/python3.7/dist-packages/keras/callbacks.py in_
<__main__.on_train_batch_end(self, batch, logs)>
      436     """
      437     if self._should_call_train_batch_hooks:
--> 438         self._call_batch_hook(ModeKeys.TRAIN, 'end', batch, logs=logs)
      439
      440     def on_test_batch_begin(self, batch, logs=None):

/usr/local/lib/python3.7/dist-packages/keras/callbacks.py in_
<__main__.call_batch_hook(self, mode, hook, batch, logs)>
      295         self._call_batch_begin_hook(mode, batch, logs)
      296     elif hook == 'end':
--> 297         self._call_batch_end_hook(mode, batch, logs)
      298     else:
      299         raise ValueError(

```

```

/usr/local/lib/python3.7/dist-packages/keras/callbacks.py in_
<__main__.call_batch_end_hook(self, mode, batch, logs)>
      316         self._batch_times.append(batch_time)
      317
--> 318     self._call_batch_hook_helper(hook_name, batch, logs)

```

```

319
320     if len(self._batch_times) >= self._num_batches_for_timing_check:
321
322         # Call batch hook helper
323         # This is the same code as in _call_batch_hook_helper()
324
325         for callback in self.callbacks:
326             hook = getattr(callback, hook_name)
327             hook(batch, logs)
328
329         if self._check_timing:
330
331             # Call on_train_batch_end hook
332             # This is the same code as in _call_on_train_batch_end()
333
334             def on_train_batch_end(self, batch, logs=None):
335                 self._batch_update_progbar(batch, logs)
336
337             def on_test_batch_end(self, batch, logs=None):
338
339                 # Call batch update progbar
340                 # This is the same code as in _batch_update_progbar()
341
342                 if self.verbose == 1:
343                     # Only block async when verbose = 1.
344                     logs = tf_utils.sync_to_numpy_or_python_type(logs)
345                     self.progbar.update(self.seen, list(logs.items()), finalize=False)
346
347
348             # Call sync to numpy or python type
349             # This is the same code as in _sync_to_numpy_or_python_type()
350
351             def sync_to_numpy_or_python_type(tensors):
352                 return t.item() if np.ndim(t) == 0 else t
353
354             return tf.nest.map_structure(_to_single_numpy_or_python_type, tensors)
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910

```

```

917         expand_composites=expand_composites)
918
/usr/local/lib/python3.7/dist-packages/keras/utils/tf_utils.py in
↳_to_single_numpy_or_python_type(t)
599     # Don't turn ragged or sparse tensors to NumPy.
600     if isinstance(t, tf.Tensor):
--> 601         t = t.numpy()
602     # Strings, ragged and sparse tensors don't have .item(). Return them
↳as-is.
603     if not isinstance(t, (np.ndarray, np.generic)):
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/ops.py in
↳numpy(self)
1157     """
1158     # TODO(slebedev): Consider avoiding a copy for non-CPU or remote
↳tensors.
-> 1159     maybe_arr = self._numpy()  # pylint: disable=protected-access
1160     return maybe_arr.copy() if isinstance(maybe_arr, np.ndarray) else
↳maybe_arr
1161
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/ops.py in
↳numpy(self)
1123     def _numpy(self):
1124         try:
-> 1125             return self._numpy_internal()
1126         except core._NotOkStatusException as e:  # pylint:u
↳disable=protected-access
1127             raise core._status_to_exception(e) from None  # pylint:u
↳disable=protected-access

```

KeyboardInterrupt:

```

[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-07
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
↳metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data =
↳([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight =
↳class_weight_dict,shuffle = True
               )

```

```

For learning Rate =  1e-07
Epoch 1/5
654/654 [=====] - 32s 37ms/step - loss: 2.3212 -
binary_accuracy: 0.2760 - val_loss: 0.2162 - val_binary_accuracy: 0.7115
Epoch 2/5
654/654 [=====] - 23s 34ms/step - loss: 0.4720 -
binary_accuracy: 0.3013 - val_loss: 0.2243 - val_binary_accuracy: 0.7135
Epoch 3/5
654/654 [=====] - 22s 34ms/step - loss: 0.2376 -
binary_accuracy: 0.6079 - val_loss: 0.2404 - val_binary_accuracy: 0.7135
Epoch 4/5
654/654 [=====] - 22s 34ms/step - loss: 0.2098 -
binary_accuracy: 0.7163 - val_loss: 0.2480 - val_binary_accuracy: 0.7135
Epoch 5/5
654/654 [=====] - 22s 34ms/step - loss: 0.2044 -
binary_accuracy: 0.7230 - val_loss: 0.2509 - val_binary_accuracy: 0.7135

```

```

[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-08
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
    ↪metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = ↪
    ↪([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight = ↪
    ↪class_weight_dict,shuffle = True
               )

```

```

For learning Rate =  1e-08
Epoch 1/5
654/654 [=====] - 29s 42ms/step - loss: 0.1911 -
binary_accuracy: 0.7238 - val_loss: 0.2165 - val_binary_accuracy: 0.7320
Epoch 2/5
654/654 [=====] - 25s 38ms/step - loss: 0.1907 -
binary_accuracy: 0.7241 - val_loss: 0.2162 - val_binary_accuracy: 0.7320
Epoch 3/5
654/654 [=====] - 25s 38ms/step - loss: 0.1910 -
binary_accuracy: 0.7238 - val_loss: 0.2154 - val_binary_accuracy: 0.7320
Epoch 4/5
654/654 [=====] - 25s 38ms/step - loss: 0.1914 -
binary_accuracy: 0.7238 - val_loss: 0.2154 - val_binary_accuracy: 0.7320
Epoch 5/5
654/654 [=====] - 25s 38ms/step - loss: 0.1908 -
binary_accuracy: 0.7241 - val_loss: 0.2160 - val_binary_accuracy: 0.7320

```

0.4 FINAL TRAINING

BEST LEARNING RATE IS 1e-04

```
[ ]: folder = '/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed'

[ ]: Batch_size = 32
     lr = 1e-04
     Epochs = 30

[ ]: with tf.device('/device:GPU:0'):
        rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
        model.compile(loss = constructive_loss, optimizer = rms,
                      metrics=['BinaryAccuracy'])

[ ]: callbacks = [
        EarlyStopping(patience=6, verbose=1),
        ReduceLROnPlateau(factor=0.1, patience=5, min_lr=0.000001, verbose=1),
        ModelCheckpoint(folder + '/Weights/resnet-bhsig260-{epoch:03d}.h5',
                      verbose=1, save_weights_only=True)
    ]

[ ]: results = model.fit(x = [S1_train, S2_train],y = Y_train,
                        validation_data = ([S1_validate,S2_validate],Y_validate),
                        epochs = Epochs,
                        callbacks = callbacks,
                        batch_size = Batch_size,
                        class_weight = class_weight_dict
                      )
```

```
Epoch 1/30
654/654 [=====] - ETA: 0s - loss: 0.2256 -
binary_accuracy: 0.7226
Epoch 1: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-001.h5
654/654 [=====] - 334s 484ms/step - loss: 0.2256 -
binary_accuracy: 0.7226 - val_loss: 0.2081 - val_binary_accuracy: 0.7216 - lr:
1.0000e-04
Epoch 2/30
654/654 [=====] - ETA: 0s - loss: 0.2051 -
binary_accuracy: 0.7237
Epoch 2: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-002.h5
654/654 [=====] - 329s 503ms/step - loss: 0.2051 -
binary_accuracy: 0.7237 - val_loss: 0.2014 - val_binary_accuracy: 0.7216 - lr:
1.0000e-04
Epoch 3/30
```

```
654/654 [=====] - ETA: 0s - loss: 0.2029 -
binary_accuracy: 0.7237
Epoch 3: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-003.h5
654/654 [=====] - 329s 503ms/step - loss: 0.2029 -
binary_accuracy: 0.7237 - val_loss: 0.2009 - val_binary_accuracy: 0.7216 - lr:
1.0000e-04
Epoch 4/30
654/654 [=====] - ETA: 0s - loss: 0.2028 -
binary_accuracy: 0.7237
Epoch 4: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-004.h5
654/654 [=====] - 309s 472ms/step - loss: 0.2028 -
binary_accuracy: 0.7237 - val_loss: 0.2009 - val_binary_accuracy: 0.7216 - lr:
1.0000e-04
Epoch 5/30
654/654 [=====] - ETA: 0s - loss: 0.2028 -
binary_accuracy: 0.7237
Epoch 5: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-005.h5
654/654 [=====] - 309s 473ms/step - loss: 0.2028 -
binary_accuracy: 0.7237 - val_loss: 0.2009 - val_binary_accuracy: 0.7216 - lr:
1.0000e-04
Epoch 6/30
654/654 [=====] - ETA: 0s - loss: 0.2029 -
binary_accuracy: 0.7237
Epoch 6: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-006.h5
654/654 [=====] - 310s 473ms/step - loss: 0.2029 -
binary_accuracy: 0.7237 - val_loss: 0.2009 - val_binary_accuracy: 0.7216 - lr:
1.0000e-04
Epoch 7/30
654/654 [=====] - ETA: 0s - loss: 0.2030 -
binary_accuracy: 0.7237
Epoch 7: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-007.h5
654/654 [=====] - 309s 472ms/step - loss: 0.2030 -
binary_accuracy: 0.7237 - val_loss: 0.2009 - val_binary_accuracy: 0.7216 - lr:
1.0000e-04
Epoch 8/30
654/654 [=====] - ETA: 0s - loss: 0.2029 -
binary_accuracy: 0.7237
Epoch 8: ReduceLROnPlateau reducing learning rate to 9.99999747378752e-06.
```

```

Epoch 8: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-008.h5
654/654 [=====] - 309s 472ms/step - loss: 0.2029 -
binary_accuracy: 0.7237 - val_loss: 0.2009 - val_binary_accuracy: 0.7216 - lr:
1.0000e-04
Epoch 9/30
654/654 [=====] - ETA: 0s - loss: 0.2030 -
binary_accuracy: 0.7237
Epoch 9: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-009.h5
654/654 [=====] - 309s 472ms/step - loss: 0.2030 -
binary_accuracy: 0.7237 - val_loss: 0.2009 - val_binary_accuracy: 0.7216 - lr:
1.0000e-05
Epoch 10/30
654/654 [=====] - ETA: 0s - loss: 0.2028 -
binary_accuracy: 0.7237
Epoch 10: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-010.h5
654/654 [=====] - 306s 468ms/step - loss: 0.2028 -
binary_accuracy: 0.7237 - val_loss: 0.2009 - val_binary_accuracy: 0.7216 - lr:
1.0000e-05
Epoch 11/30
654/654 [=====] - ETA: 0s - loss: 0.2031 -
binary_accuracy: 0.7237
Epoch 11: saving model to
/content/drive/MyDrive/ResNet(1e-04)Hindi220x155Inversed/Weights/resnet-
bhsig260-011.h5
654/654 [=====] - 306s 468ms/step - loss: 0.2031 -
binary_accuracy: 0.7237 - val_loss: 0.2009 - val_binary_accuracy: 0.7216 - lr:
1.0000e-05
Epoch 11: early stopping

```

```

[ ]: # Y_train.shape
[ ]: import pickle
[ ]: with open(folder + '/trainHistoryDict', 'wb') as file_pi:
        pickle.dump(results, file_pi)
# model.save(folder + 'BestModel.h5')

```

```

-----
NameError                                 Traceback (most recent call last)
<ipython-input-30-1b37eae3722b> in <module>
      1 with open(folder + '/trainHistoryDict', 'wb') as file_pi:

```

```
----> 2         pickle.dump(results, file_pi)
  3 # model.save(folder + 'BestModel.h5')
```

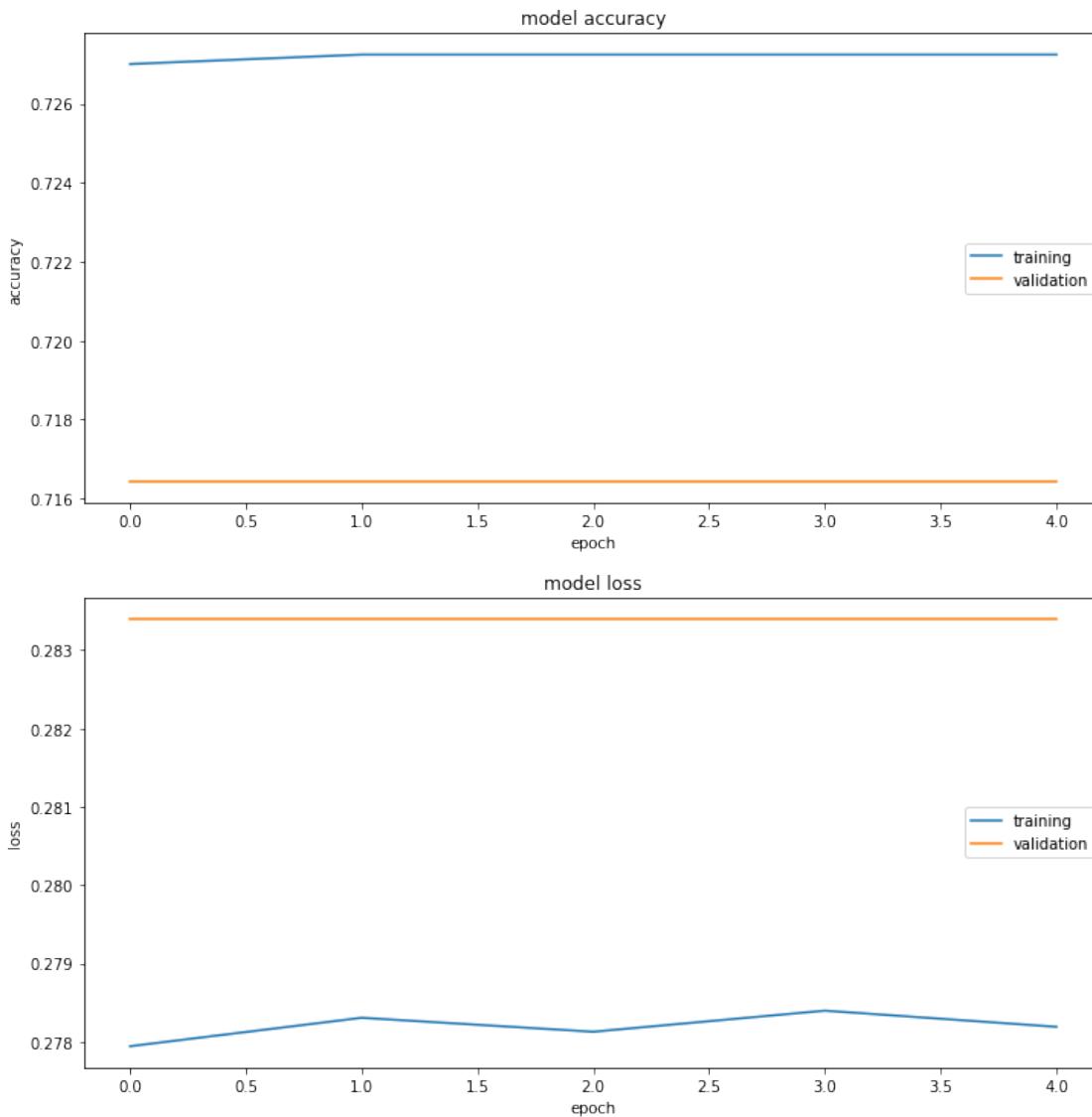
```
NameError: name 'results' is not defined
```

```
[ ]: # import pickle
# history = pickle.load(open(folder + '/trainHistoryDict', "rb"))

[ ]: results.history['val_loss'].index(min(results.history['val_loss'])) + 1

[ ]: def display_training_curves(training, validation, title, subplot):
    ax = plt.subplot(subplot)
    ax.plot(training)
    ax.plot(validation)
    ax.set_title('model ' + title)
    ax.set_ylabel(title)
    ax.set_xlabel('epoch')
    ax.legend(['training', 'validation'])

plt.subplots(figsize=(10,10))
plt.tight_layout()
display_training_curves(results.history['binary_accuracy'], results.
    ↪history['val_binary_accuracy'], 'accuracy', 211)
display_training_curves(results.history['loss'], results.history['val_loss'], ↪
    ↪'loss', 212)
plt.savefig(folder + "loss_accuracy.svg", dpi = 1200)
```



```
[ ]: y_pred_keras
```

```
[ ]: <tf.Tensor: shape=(4482,), dtype=int64, numpy=array([0, 0, 0, ..., 0, 0, 0])>
```

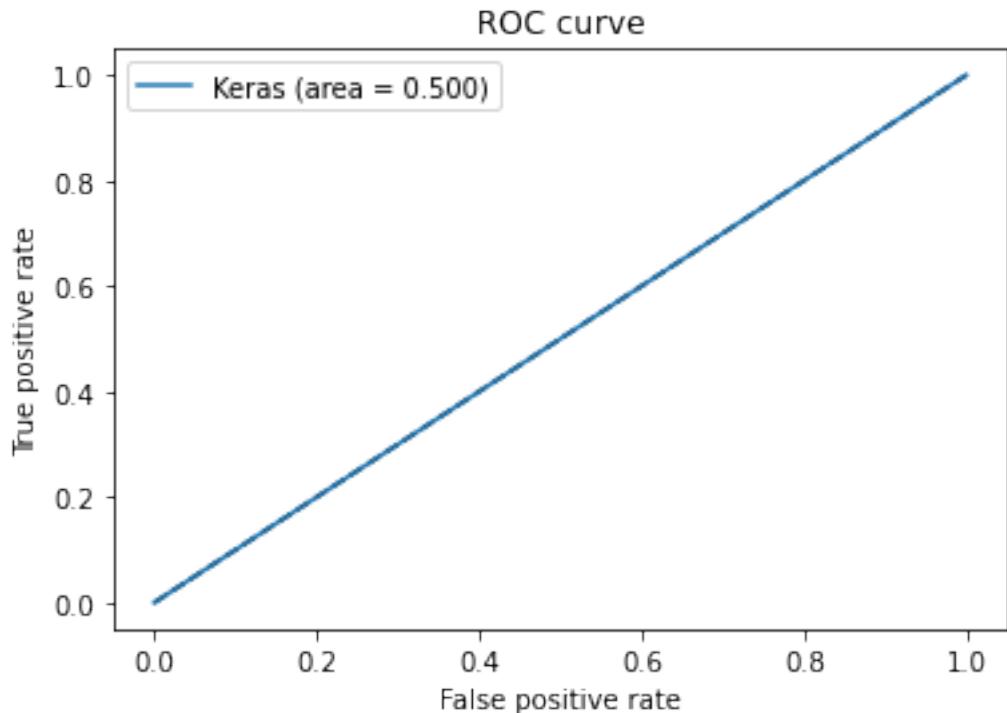
```
[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
y_pred_keras = model.predict([S1_test, S2_test])
fpr_keras, tpr_keras, thresholds_keras = roc_curve(Y_test, y_pred_keras)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
```

```

# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.svg", dpi = 1200)
plt.show()

```

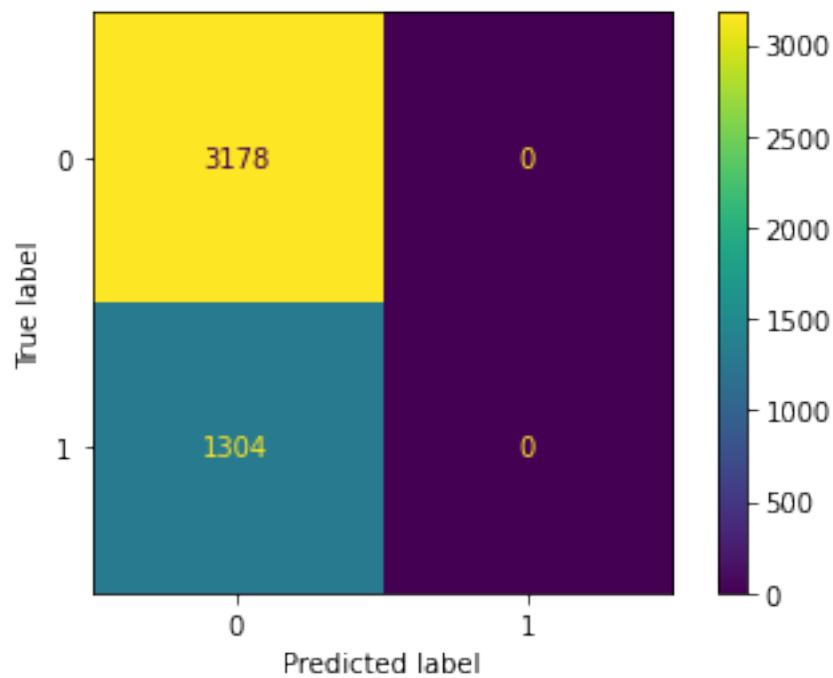
141/141 [=====] - 30s 211ms/step



```

[ ]: ConfusionMatrixDisplay(confusion_matrix(Y_test, tf.argmax(y_pred_keras, axis = 1))).plot()
plt.savefig(folder + "confusionmatrix.svg", dpi = 1200)
plt.show()

```



```
[ ]: y_pred_keras
```

```
[ ]: array([[0.00031623],  
           [0.00031623],  
           [0.00031623],  
           ...,  
           [0.00031623],  
           [0.00031623],  
           [0.00031623]], dtype=float32)
```

```
[ ]:
```

Model5_Resnet18Siamese_BhSig260HindiTraining

November 14, 2024

```
#Model 5 (ResNet18 + Siamese)
```

```
[ ]: from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
import os
from tensorflow.keras.initializers import RandomNormal, Constant
import tensorflow as tf
from tensorflow.keras.backend import max, mean, sqrt, square, sum

import seaborn as sns
from tensorflow.keras.optimizers import Adam,RMSprop
from keras.models import Model

from tensorflow import keras
from keras.layers import LeakyReLU, Softmax
from keras.layers import Conv2D, Activation, Input,Dropout,Lambda,Flatten, Dense
from keras.layers import Dense, Flatten, Reshape, Activation
from keras.layers import BatchNormalization ,ZeroPadding2D

from keras.layers import MaxPooling2D
from keras.layers import Concatenate
from keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
from keras.layers import Layer, InputSpec
from keras.regularizers import l2
from keras import backend as K
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from tensorflow import one_hot, reshape

from tensorflow.keras.layers import Activation, Add, AvgPool2D, ↴
    ↴BatchNormalization, Conv2D, Dense, Flatten, Input, MaxPool2D, ZeroPadding2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.initializers import random_uniform, glorot_uniform, ↴
    ↴constant
from tensorflow.keras.utils import plot_model
```

```

from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model

```

```

[ ]: import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))

```

Found GPU at: /device:GPU:0

```

[ ]: from google.colab import drive
drive.mount('/content/drive')

```

Mounted at /content/drive

```

[ ]: with tf.device('/device:GPU:0'):
    colab = True
    # database = 'bigsig260_224x224_siamese_preprocessed.h5'
    if colab:
        # from google.colab import drive
        # drive.mount('/content/gdrive')
        file = '/content/drive/MyDrive/bigsig260224x224x1_siamese_preprocessed.h5'

```

```

else:
    file = os.path.join(os.getcwd(), 'RoboticLab', database)
    print(file)
    with File(file, 'r') as hdf:
        S1 = np.array(hdf.get('S1'))
        S2 = np.array(hdf.get('S2'))
        Y = np.array(hdf.get('Y'))
    print(S1.shape)
    print(S2.shape)
    print(Y.shape)

```

```

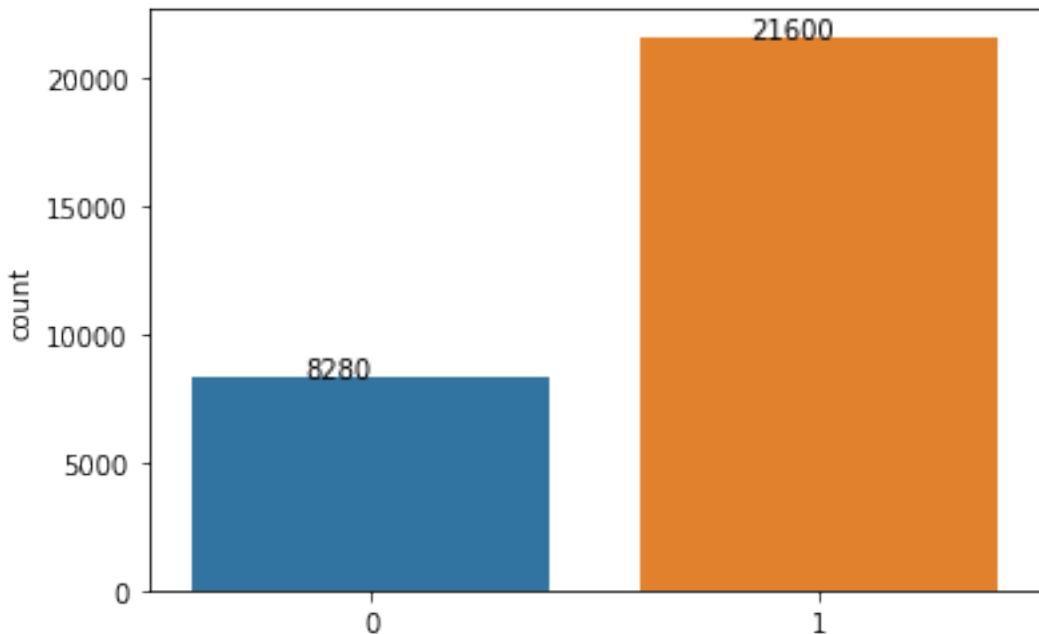
/content/drive/MyDrive/bigsig260224x224x1_siamese_preprocessed.h5
(29880, 224, 224, 1)
(29880, 224, 224, 1)
(29880, 1)

```

```

[ ]: ax = sns.countplot(x = Y.reshape(Y.shape[0]))
for p in ax.patches:
    ax.annotate('{:}.'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.
    ↵01))

```



```

[ ]: 21600/(21600 + 8280)

```

```

[ ]: 0.7228915662650602

```

```
[ ]: import sklearn
class_weights = sklearn.utils.class_weight.compute_class_weight(class_weight = "balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict
```

```
[ ]: {0: 1.8043478260869565, 1: 0.6916666666666667}
```

```
[ ]: Y = Y/1.0
```

```
[ ]: S1.dtype
```

```
[ ]: dtype('uint8')
```

```
[ ]: with tf.device('/device:GPU:0'):
    seed=randint(10)
    print('seed=' + str(seed))
    indices = permutation(Y.shape[0])
    m = int(0.70 * Y.shape[0])
    n = int(0.15 * Y.shape[0])
    training_id, validation_id, test_id = indices[:m], indices[m:m+n], indices[m+n:]
    S1_train, S1_test, S1_validate = S1[training_id], S1[test_id], S1[validation_id]
    S2_train, S2_test, S2_validate = S2[training_id], S2[test_id], S2[validation_id]
    Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]
    print(S1_train.shape)
    print(S2_train.shape)
    print(Y_train.shape)

del S1,S2,Y
```

```
seed=8
(20916, 224, 224, 1)
(20916, 224, 224, 1)
(20916, 1)
```

```
[ ]: fig = plt.figure(figsize = (16,16))
rows,cols = 4,4
i = 1
while(i in range(1,rows*cols)):
    random_idx = randint(0,len(S1_train))
    if(Y_train[random_idx] == 0):
        Label = "Genuine Pair"
    else:
        Label = "Forged Pair"
    img1 = S1_train[random_idx]
```

```

img2 = S2_train[random_idx]
fig.add_subplot(rows,cols,i)
plt.imshow(img1.squeeze(),cmap = "gray");
plt.axis(False);
i += 1
fig.add_subplot(rows,cols,i)
plt.imshow(img2.squeeze(),cmap = "gray");
plt.axis(False);
plt.text(0.5, 0.5, Label,
horizontalalignment='center',verticalalignment='center',fontsize=15,fontweight='bold',
color='red');
i += 1

```

Genuine Pair

ਪ੍ਰਾਣ ਅਨੁਸਾਰੀ.

ਪ੍ਰਾਣ ਅਨੁਸਾਰੀ.

Forged Pair

ਪੜ੍ਹਾ ਝੂ।

ਪੜ੍ਹਾ ਝੂ।

Forged Pair

ਕਿਉਣ ਮੀਸ਼ਾ

ਕਿਉਣ ਮੀਸ਼ਾ

Genuine Pair

ਸ਼ਿਆਮਕਾ ਪੈਟੌਂ

ਸ਼ਿਆਮਕਾ ਪੈਟੌਂ

Genuine Pair

ਰਨੀ ਸਾਡੇ ਕਾਲੇ ਗੁਰੂ

ਰਨੀ ਸਾਡੇ ਕਾਲੇ ਗੁਰੂ

Genuine Pair

ਗੁਰੂ ਪਾਂਡੀ

ਗੁਰੂ ਪਾਂਡੀ

Forged Pair

ਮੁਖੀ ਮੁਖਗਲ

ਆਕੂਤੀ ਅਗੁਰਾਲ

Forged Pair

ਗੁਲਾਮਾਨ

ਗੁਲਾਮ ਖਾਨ

```

[ ]: Y_train[0]

[ ]: array([0.])

[ ]: # #One hot Encoding
# Y_train = one_hot(Y_train, depth=2)
# Y_train = reshape(Y_train, (-1, 2))

# Y_test = one_hot(Y_test, depth=2)
# Y_test = reshape(Y_test, (-1, 2))

# Y_validate = one_hot(Y_validate, depth=2)
# Y_validate = reshape(Y_validate, (-1, 2))

```

```

[ ]: Y_train[0]

[ ]: array([0.])

[ ]: Y_train.shape

[ ]: (20916, 1)

[ ]:
    """
    ResNet-18
    Reference:
    [1] K. He et al. Deep Residual Learning for Image Recognition. CVPR, 2016
    [2] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers:
    Surpassing human-level performance on imagenet classification. In
    ICCV, 2015.
    """
    from keras.callbacks import EarlyStopping
    from keras.layers import Dense, Conv2D, MaxPool2D, Flatten,
        GlobalAveragePooling2D, BatchNormalization, Layer, Add
    from keras.models import Sequential
    from keras.models import Model
    import tensorflow as tf

    class ResnetBlock(Model):
        """
        A standard resnet block.
        """

        def __init__(self, channels: int, down_sample=False):
            """
            channels: same as number of convolution kernels
            """
            super().__init__()

```

```

    self.__channels = channels
    self.__down_sample = down_sample
    self.__strides = [2, 1] if down_sample else [1, 1]

    KERNEL_SIZE = (3, 3)
    # use He initialization, instead of Xavier (a.k.a 'glorot_uniform' in
    ↵Keras), as suggested in [2]
    INIT_SCHEME = "glorot_uniform"

    self.conv_1 = Conv2D(self.__channels, strides=self.__strides[0],
                        kernel_size=KERNEL_SIZE, padding="same", ↵
    ↵kernel_initializer=INIT_SCHEME)
    self.bn_1 = BatchNormalization()
    self.conv_2 = Conv2D(self.__channels, strides=self.__strides[1],
                        kernel_size=KERNEL_SIZE, padding="same", ↵
    ↵kernel_initializer=INIT_SCHEME)
    self.bn_2 = BatchNormalization()
    self.merge = Add()

    if self.__down_sample:
        # perform down sampling using stride of 2, according to [1].
        self.res_conv = Conv2D(
            self.__channels, strides=2, kernel_size=(1, 1), ↵
    ↵kernel_initializer=INIT_SCHEME, padding="same")
        self.res_bn = BatchNormalization()

    def call(self, inputs):
        res = inputs

        x = self.conv_1(inputs)
        x = self.bn_1(x)
        x = tf.nn.relu(x)
        x = self.conv_2(x)
        x = self.bn_2(x)

        if self.__down_sample:
            res = self.res_conv(res)
            res = self.res_bn(res)

        # if not perform down sample, then add a shortcut directly
        x = self.merge([x, res])
        out = tf.nn.relu(x)
        return out

    class ResNet18(Model):

```

```

def __init__(self, num_classes, **kwargs):
    """
        num_classes: number of classes in specific classification task.
    """
    super().__init__(**kwargs)
    self.conv_1 = Conv2D(64, (7, 7), ↴
    ↪strides=2,kernel_initializer="glorot_uniform")
    self.init_bn = BatchNormalization()
    self.pool_2 = MaxPool2D(pool_size=(3, 3), strides=2)
    self.res_1_1 = ResnetBlock(64)
    self.res_1_2 = ResnetBlock(64)
    self.res_2_1 = ResnetBlock(128, down_sample=True)
    self.res_2_2 = ResnetBlock(128)
    self.res_3_1 = ResnetBlock(256, down_sample=True)
    self.res_3_2 = ResnetBlock(256)
    self.res_4_1 = ResnetBlock(512, down_sample=True)
    self.res_4_2 = ResnetBlock(512)
    self.avg_pool = AvgPool2D()
    self.flat = Flatten()
    self.fc = Dense(num_classes, activation="sigmoid")

def call(self, inputs):
    out = self.conv_1(inputs)
    out = self.init_bn(out)
    out = tf.nn.relu(out)
    out = self.pool_2(out)
    for res_block in [self.res_1_1, self.res_1_2, self.res_2_1, self.
    ↪res_2_2, self.res_3_1, self.res_3_2, self.res_4_1, self.res_4_2]:
        out = res_block(out)
    out = self.avg_pool(out)
    out = self.flat(out)
    out = self.fc(out)
    return out

```

```
[ ]: input_shape=(224,224,1)
input_shape
```

```
[ ]: (224, 224, 1)
```

```
[ ]: def euclidean_distance(vects):
    '''Compute Euclidean Distance between two vectors'''
    x, y = vects
    return K.sqrt(K.maximum(K.sum(K.square(x - y), axis=1, keepdims=True), K.
    ↪epsilon()))
```

```
[ ]: def constructive_loss(y_true, y_pred):
    '''Contrastive loss from Hadsell-et-al.'06
    http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf
    '''
    margin = 1
    y_true = tf.cast(y_true, y_pred.dtype)
    squaredPreds = K.square(y_pred)
    squaredMargin = K.square(K.maximum(margin - y_pred, 0))
    loss = K.mean(y_true * squaredMargin + (1 - y_true) * squaredPreds)
    return loss
```

```
[ ]: with tf.device('/device:GPU:0'):
    inputShape = S1_train.shape[1:]
    f = ResNet18(num_classes = 2)
    f.build((None,224,224,1))
```

WARNING:tensorflow:AutoGraph could not transform <bound method ResnetBlock.call of <__main__.ResnetBlock object at 0x7f8da01e7730>> and will run it as-is.
Cause: mangled names are not yet supported
To silence this warning, decorate the function with
@tf.autograph.experimental.do_not_convert

WARNING: AutoGraph could not transform <bound method ResnetBlock.call of <__main__.ResnetBlock object at 0x7f8da01e7730>> and will run it as-is.
Cause: mangled names are not yet supported
To silence this warning, decorate the function with
@tf.autograph.experimental.do_not_convert

```
[ ]: def eucl_dist_output_shape(shapes):
    shape1, shape2 = shapes
    return (shape1[0], 1)
```

```
[ ]: # network definition
input_a = Input(shape=(input_shape))
input_b = Input(shape=(input_shape))

# because we re-use the same instance `base_network`,
# the weights of the network
# will be shared across the two branches
processed_a = f(input_a)
processed_b = f(input_b)

# Compute the Euclidean distance between the two vectors in the latent space
distance = Lambda(euclidean_distance, [
    output_shape=eucl_dist_output_shape)([processed_a, processed_b])

model = Model(inputs = [input_a, input_b], outputs = distance)
```

```
[ ]: pred = model([expand_dims(S1_train[0],axis = 0), expand_dims(S2_train[0],axis = 0)])
print(pred.shape)
pred

(1, 1)

[ ]: <tf.Tensor: shape=(1, 1), dtype=float32, numpy=array([[0.32704693]], dtype=float32)>
```

0.1 HYPER PARAMETER TUNING

0.1.1 LEARNING RATE

```
[ ]: batch_Size = 32

[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-03
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
    metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
              validation_data = ([S1_validate,S2_validate],Y_validate),
              epochs = 5,batch_size = batch_Size,class_weight = class_weight_dict
    )

[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-04
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
    metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
              validation_data =
    ([S1_validate,S2_validate],Y_validate),
              epochs = 5,batch_size = batch_Size,class_weight =
    class_weight_dict
    )
```

For learning Rate = 0.0001
Epoch 1/5
559/654 [=====>...] - ETA: 19s - loss: 0.2939 -
binary_accuracy: 0.6672

KeyboardInterrupt
<ipython-input-27-ad7150843c07> in <module>

Traceback (most recent call last)

```

    4     rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho =
→0.9)
    5     model.compile(loss = constructive_loss, optimizer = rms,✉
←metrics=['BinaryAccuracy'])
--> 6     model.fit(x = [S1_train, S2_train],y = Y_train,
    7                     validation_data =✉
←([S1_validate,S2_validate],Y_validate),
    8                     epochs = 5,batch_size = batch_Size,class_weight=
← class_weight_dict

/usr/local/lib/python3.8/dist-packages/keras/utils/traceback_utils.py in✉
←error_handler(*args, **kwargs)
    62     filtered_tb = None
    63     try:
--> 64         return fn(*args, **kwargs)
    65     except Exception as e: # pylint: disable=broad-except
    66         filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.8/dist-packages/keras/engine/training.py in fit(self, x,
→y, batch_size, epochs, verbose, callbacks, validation_split, validation_data,
→shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch,✉
→validation_steps, validation_batch_size, validation_freq, max_queue_size,✉
→workers, use_multiprocessing)
    1412             logs = tmp_logs # No error, now safe to assign to logs.
    1413             end_step = step + data_handler.step_increment
--> 1414             callbacks.on_train_batch_end(end_step, logs)
    1415             if self.stop_training:
    1416                 break

/usr/local/lib/python3.8/dist-packages/keras/callbacks.py in✉
←on_train_batch_end(self, batch, logs)
    436     """
    437     if self._should_call_train_batch_hooks:
--> 438         self._call_batch_hook(ModeKeys.TRAIN, 'end', batch, logs=logs)
    439
    440     def on_test_batch_begin(self, batch, logs=None):

/usr/local/lib/python3.8/dist-packages/keras/callbacks.py in✉
←_call_batch_hook(self, mode, hook, batch, logs)
    295         self._call_batch_begin_hook(mode, batch, logs)
    296     elif hook == 'end':
--> 297         self._call_batch_end_hook(mode, batch, logs)
    298     else:
    299         raise ValueError()

/usr/local/lib/python3.8/dist-packages/keras/callbacks.py in✉
←_call_batch_end_hook(self, mode, batch, logs)
    316         self._batch_times.append(batch_time)

```

```

 317
--> 318     self._call_batch_hook_helper(hook_name, batch, logs)
 319
 320     if len(self._batch_times) >= self._num_batches_for_timing_check:

```

/usr/local/lib/python3.8/dist-packages/keras/callbacks.py in `_call_batch_hook_helper(self, hook_name, batch, logs)`

```

 321         for callback in self.callbacks:
 322             hook = getattr(callback, hook_name)
--> 323             hook(batch, logs)
 324
 325     if self._check_timing:
```

/usr/local/lib/python3.8/dist-packages/keras/callbacks.py in `on_train_batch_end(self, batch, logs)`

```

 326
 327     def on_train_batch_end(self, batch, logs=None):
-> 328         self._batch_update_progbar(batch, logs)
 329
 330     def on_test_batch_end(self, batch, logs=None):
```

/usr/local/lib/python3.8/dist-packages/keras/callbacks.py in `_batch_update_progbar(self, batch, logs)`

```

 331
 332     if self.verbose == 1:
 333         # Only block async when verbose = 1.
-> 334         logs = tf_utils.sync_to_numpy_or_python_type(logs)
 335         self.progbar.update(self.seen, list(logs.items()), finalize=False)
 336
 337
```

/usr/local/lib/python3.8/dist-packages/keras/utils/tf_utils.py in `sync_to_numpy_or_python_type(tensor)`

```

 338
 339     return t.item() if np.ndim(t) == 0 else t
 340
--> 341     return tf.nest.map_structure(_to_single_numpy_or_python_type, tensors)
 342
 343
 344
```

/usr/local/lib/python3.8/dist-packages/tensorflow/python/util/nest.py in `map_structure(func, *structure, **kwargs)`

```

 345
 346     return pack_sequence_as(
--> 347         structure[0], [func(*x) for x in entries],
 348         expand_composites=expand_composites)
 349
 350
```

/usr/local/lib/python3.8/dist-packages/tensorflow/python/util/nest.py in `<listcomp>(.0)`

```

 351
 352     914
 353
 354     915     return pack_sequence_as(
--> 355         structure[0], [func(*x) for x in entries],
 356         expand_composites=expand_composites)
 357
 358
```

```

  915     return pack_sequence_as(
--> 916         structure[0], [func(*x) for x in entries],
  917         expand_composites=expand_composites)
  918

/usr/local/lib/python3.8/dist-packages/keras/utils/tf_utils.py in __
_to_single_numpy_or_python_type(t)
 599     # Don't turn ragged or sparse tensors to NumPy.
 600     if isinstance(t, tf.Tensor):
--> 601         t = t.numpy()
 602     # Strings, ragged and sparse tensors don't have .item(). Return them
__as-is.
 603     if not isinstance(t, (np.ndarray, np.generic)):

/usr/local/lib/python3.8/dist-packages/tensorflow/python/framework/ops.py in __
numpy(self)
 1157     """
 1158     # TODO(slebedev): Consider avoiding a copy for non-CPU or remote
__tensors.
-> 1159     maybe_arr = self._numpy()  # pylint: disable=protected-access
 1160     return maybe_arr.copy() if isinstance(maybe_arr, np.ndarray) else
__maybe_arr
 1161

/usr/local/lib/python3.8/dist-packages/tensorflow/python/framework/ops.py in __
numpy(self)
 1123     def __numpy__(self):
 1124         try:
--> 1125             return self._numpy_internal()
 1126         except core._NotOkStatusException as e:  # pylint: disable=
protected-access
 1127             raise core._status_to_exception(e) from None  # pylint: disable=
protected-access

```

KeyboardInterrupt:

```
[ ]: with tf.device('/device:GPU:0'):
    lr = 1e-05
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,
__metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
              validation_data = ([S1_validate,S2_validate],Y_validate),
```

```

    epochs = 5,batch_size = batch_Size,class_weight = □
    ↵class_weight_dict
)

```

[]: with tf.device('/device:GPU:0'):

```

    lr = 1e-06
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,□
    ↵metrics=['BinaryAccuracy' ])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = □
    ↵([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight = □
    ↵class_weight_dict
)

```

[]: with tf.device('/device:GPU:0'):

```

    lr = 1e-07
    print("For learning Rate = ",lr)
    rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
    model.compile(loss = constructive_loss, optimizer = rms,□
    ↵metrics=['BinaryAccuracy'])
    model.fit(x = [S1_train, S2_train],y = Y_train,
               validation_data = □
    ↵([S1_validate,S2_validate],Y_validate),
               epochs = 5,batch_size = batch_Size,class_weight = □
    ↵class_weight_dict
)

```

0.2 FINAL TRAINING

BEST LEARNING RATE IS 1e-04

[]: folder = '/content/drive/MyDrive/ResNet18(1e-04)Hindi224x224x1Inversed'

[]: Batch_size = 32
 lr = 1e-04
 Epochs = 30

[]: with tf.device('/device:GPU:0'):
 rms = RMSprop(learning_rate = lr,epsilon = 1e-08,momentum = 0.9,rho = 0.9)
 model.compile(loss = constructive_loss, optimizer = rms,□
 ↵metrics=['BinaryAccuracy'])

[]: callbacks = [
 EarlyStopping(patience=6, verbose=1),

```

    ReduceLROnPlateau(factor=0.1, patience=5, min_lr=0.000001, verbose=1),
    ModelCheckpoint(folder + '/Weights/resnet18-bhsig260-{epoch:03d}.h5',  

    ↴verbose=1, save_weights_only=True)
]

[ ]: with tf.device('/device:GPU:0'):
    results = model.fit(x = [S1_train, S2_train],y = Y_train,
        validation_data = ([S1_validate,S2_validate],Y_validate),
        epochs = Epochs,
        callbacks = callbacks,
        batch_size = Batch_size,
        class_weight = class_weight_dict
    )

```

```

Epoch 1/30
654/654 [=====] - ETA: 0s - loss: 0.2708 -
binary_accuracy: 0.6890
Epoch 1: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-001.h5
654/654 [=====] - 155s 227ms/step - loss: 0.2708 -
binary_accuracy: 0.6890 - val_loss: 0.3778 - val_binary_accuracy: 0.5694 - lr:
1.0000e-04
Epoch 2/30
654/654 [=====] - ETA: 0s - loss: 0.0354 -
binary_accuracy: 0.9552
Epoch 2: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-002.h5
654/654 [=====] - 143s 219ms/step - loss: 0.0354 -
binary_accuracy: 0.9552 - val_loss: 0.6919 - val_binary_accuracy: 0.2947 - lr:
1.0000e-04
Epoch 3/30
654/654 [=====] - ETA: 0s - loss: 0.0101 -
binary_accuracy: 0.9872
Epoch 3: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-003.h5
654/654 [=====] - 143s 218ms/step - loss: 0.0101 -
binary_accuracy: 0.9872 - val_loss: 0.4025 - val_binary_accuracy: 0.5533 - lr:
1.0000e-04
Epoch 4/30
654/654 [=====] - ETA: 0s - loss: 0.0064 -
binary_accuracy: 0.9925
Epoch 4: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-004.h5
654/654 [=====] - 143s 219ms/step - loss: 0.0064 -
binary_accuracy: 0.9925 - val_loss: 0.2329 - val_binary_accuracy: 0.7352 - lr:
1.0000e-04
Epoch 5/30
654/654 [=====] - ETA: 0s - loss: 0.0047 -

```

```
binary_accuracy: 0.9948
Epoch 5: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-005.h5
654/654 [=====] - 143s 219ms/step - loss: 0.0047 -
binary_accuracy: 0.9948 - val_loss: 0.0020 - val_binary_accuracy: 0.9980 - lr:
1.0000e-04
Epoch 6/30
654/654 [=====] - ETA: 0s - loss: 0.0040 -
binary_accuracy: 0.9956
Epoch 6: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-006.h5
654/654 [=====] - 142s 217ms/step - loss: 0.0040 -
binary_accuracy: 0.9956 - val_loss: 0.0253 - val_binary_accuracy: 0.9692 - lr:
1.0000e-04
Epoch 7/30
654/654 [=====] - ETA: 0s - loss: 0.0023 -
binary_accuracy: 0.9975
Epoch 7: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-007.h5
654/654 [=====] - 142s 218ms/step - loss: 0.0023 -
binary_accuracy: 0.9975 - val_loss: 0.1035 - val_binary_accuracy: 0.8846 - lr:
1.0000e-04
Epoch 8/30
654/654 [=====] - ETA: 0s - loss: 0.0020 -
binary_accuracy: 0.9979
Epoch 8: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-008.h5
654/654 [=====] - 142s 218ms/step - loss: 0.0020 -
binary_accuracy: 0.9979 - val_loss: 0.0054 - val_binary_accuracy: 0.9949 - lr:
1.0000e-04
Epoch 9/30
654/654 [=====] - ETA: 0s - loss: 0.0017 -
binary_accuracy: 0.9983
Epoch 9: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-009.h5
654/654 [=====] - 142s 218ms/step - loss: 0.0017 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-04
Epoch 10/30
654/654 [=====] - ETA: 0s - loss: 0.0017 -
binary_accuracy: 0.9983
Epoch 10: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-010.h5
654/654 [=====] - 142s 218ms/step - loss: 0.0017 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-04
Epoch 11/30
654/654 [=====] - ETA: 0s - loss: 0.0017 -
```

```
binary_accuracy: 0.9983
Epoch 11: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-011.h5
654/654 [=====] - 143s 219ms/step - loss: 0.0017 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-04
Epoch 12/30
654/654 [=====] - ETA: 0s - loss: 0.0016 -
binary_accuracy: 0.9983
Epoch 12: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-012.h5
654/654 [=====] - 142s 217ms/step - loss: 0.0016 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-04
Epoch 13/30
654/654 [=====] - ETA: 0s - loss: 0.0016 -
binary_accuracy: 0.9983
Epoch 13: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-013.h5
654/654 [=====] - 142s 217ms/step - loss: 0.0016 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-04
Epoch 14/30
654/654 [=====] - ETA: 0s - loss: 0.0016 -
binary_accuracy: 0.9983
Epoch 14: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-06.

Epoch 14: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-014.h5
654/654 [=====] - 142s 218ms/step - loss: 0.0016 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-04
Epoch 15/30
654/654 [=====] - ETA: 0s - loss: 0.0017 -
binary_accuracy: 0.9983
Epoch 15: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-015.h5
654/654 [=====] - 142s 218ms/step - loss: 0.0017 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-05
Epoch 16/30
654/654 [=====] - ETA: 0s - loss: 0.0017 -
binary_accuracy: 0.9983
Epoch 16: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-016.h5
654/654 [=====] - 142s 218ms/step - loss: 0.0017 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-05
```

```

Epoch 17/30
654/654 [=====] - ETA: 0s - loss: 0.0017 -
binary_accuracy: 0.9983
Epoch 17: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-017.h5
654/654 [=====] - 142s 217ms/step - loss: 0.0017 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-05
Epoch 18/30
654/654 [=====] - ETA: 0s - loss: 0.0017 -
binary_accuracy: 0.9983
Epoch 18: saving model to /content/drive/MyDrive/ResNet18(1e-
04)Hindi224x224x1Inversed/Weights/resnet18-bhsig260-018.h5
654/654 [=====] - 144s 220ms/step - loss: 0.0017 -
binary_accuracy: 0.9983 - val_loss: 0.0013 - val_binary_accuracy: 0.9987 - lr:
1.0000e-05
Epoch 18: early stopping

```

```

[ ]: # Y_train.shape
[ ]: import pickle
[ ]: # with open(folder + '/trainHistoryDict.h5', 'wb') as file_pi:
#       pickle.dump(results, file_pi)
# # model.save(folder + 'BestModel.h5')
[ ]: import pickle
results = pickle.load(open(folder + '/trainHistoryDict', "rb"))
[ ]: best_result = results.history['val_loss'].index(min(results.
    ↪history['val_loss'])) + 1
best_result

```

[]: 12

```

[ ]: def display_training_curves(training, validation, title, subplot):
    ax = plt.subplot(subplot)
    ax.plot(training)
    ax.plot(validation)
    ax.set_title('model ' + title)
    ax.set_ylabel(title)
    ax.set_xlabel('epoch')
    ax.legend(['training', 'validation'])

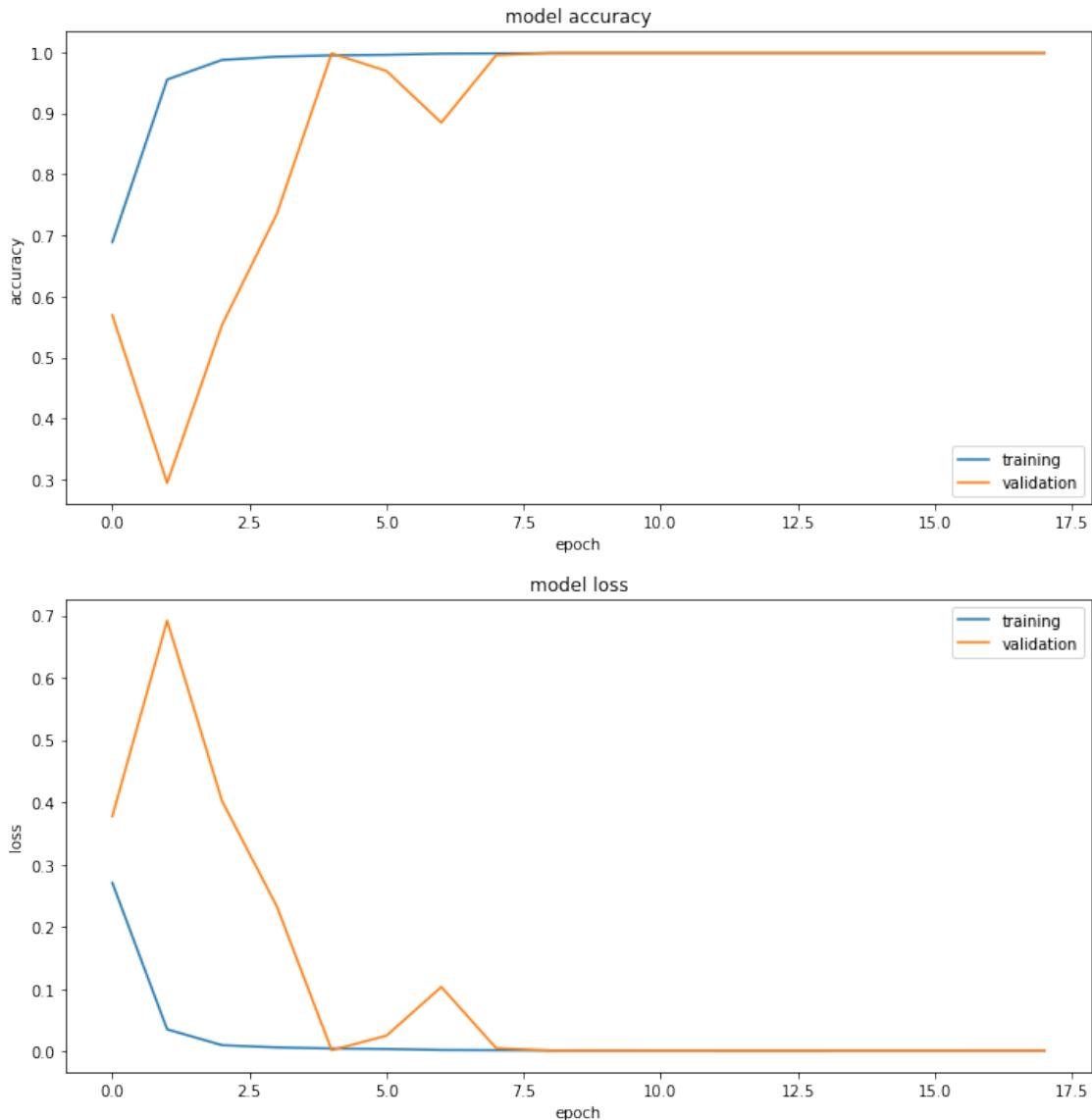
plt.subplots(figsize=(10,10))
plt.tight_layout()
display_training_curves(results.history['binary_accuracy'], results.
    ↪history['val_binary_accuracy'], 'accuracy', 211)

```

```

display_training_curves(results.history['loss'], results.history['val_loss'],
    ↪'loss', 212)
plt.savefig(folder + "loss_accuracy.svg", dpi = 1200)

```



```

[ ]: best_result = 12
if(best_result < 10):
    model.load_weights(folder + f'/Weights/resnet18-bhsig260-0{best_result}.h5')
else:
    model.load_weights(folder + f'/Weights/resnet18-bhsig260-0{best_result}.h5')

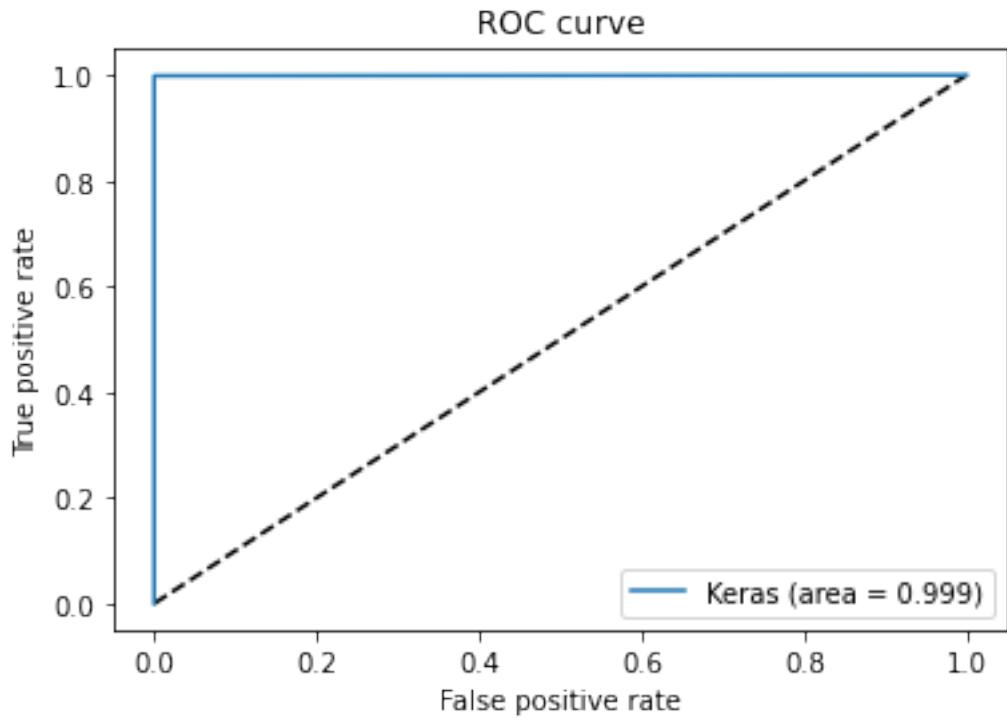
[ ]: # if we keep threshold as 0.99
threshold = 0.99

```

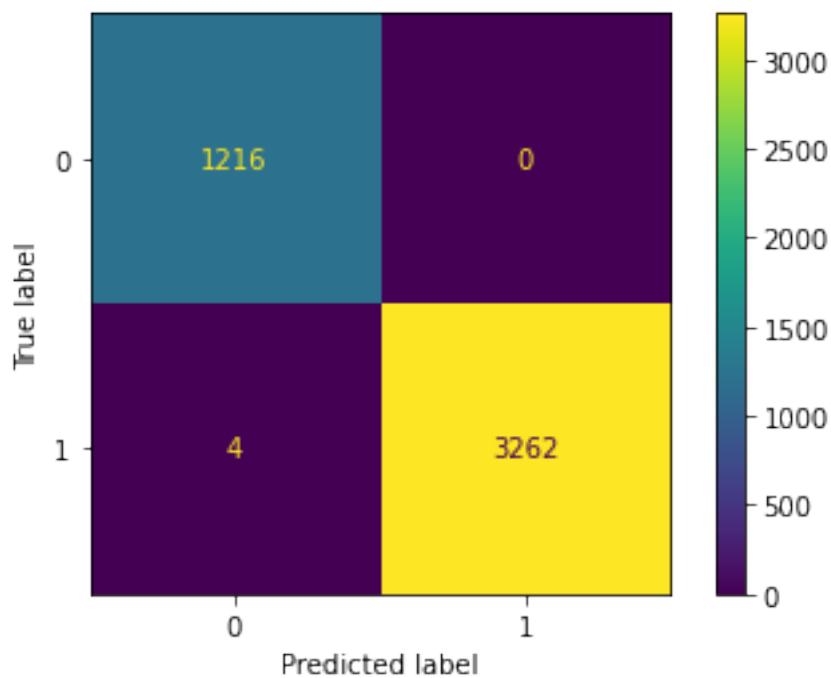
```
[ ]: y_pred_keras = model.predict([S1_test, S2_test])
141/141 [=====] - 17s 63ms/step

[ ]: y_pred_keras_new = np.zeros((len(y_pred_keras), 1), dtype = np.uint8)
for i in range(len(y_pred_keras)):
    label = Y_test[i]
    if y_pred_keras[i] >= threshold:
        predicted_value = 1 # both different
        if(label == predicted_value):
            y_pred_keras_new[i] = label
        else:
            y_pred_keras_new[i] = predicted_value
    else:
        predicted_value = 0
        if(label == predicted_value):
            y_pred_keras_new[i] = label
        else:
            y_pred_keras_new[i] = predicted_value

[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
fpr_keras, tpr_keras, thresholds_keras = roc_curve(Y_test, y_pred_keras_new)
auc_keras = auc(fpr_keras, tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f})'.format(auc_keras))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.svg", dpi = 1200)
plt.show()
```



```
[ ]: ConfusionMatrixDisplay(confusion_matrix(Y_test,y_pred_keras_new)).plot()
plt.savefig(folder + "confusionmatrix.svg",dpi = 1200)
plt.show()
```



Load the weights from the epoch which gave the best validation accuracy

```
[ ]: def predict_score(i = 0):
    '''Predict distance score and classify test images as Genuine or Forged'''
    test_gen = ([S1_test[i],S2_test[i]],Y_test[i])
    (img1, img2), label = test_gen

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 10))
    ax1.imshow(np.squeeze(img1), cmap='gray')
    ax2.imshow(np.squeeze(img2), cmap='gray')
    ax1.set_title('Genuine')
    if label == 0:
        ax2.set_title('Genuine')
    else:
        ax2.set_title('Forged')
    ax1.axis('off')
    ax2.axis('off')
    plt.show()
    result = model.predict([np.expand_dims(a = img1, axis = 0), np.expand_dims(a =
    ↪= img2, axis = 0)])
    diff = result[0][0]
    print("Difference Score = ", diff)
    if diff >= threshold:
        print("Its a Forged Signature")
    else:
        print("Its a Genuine Signature")
```

```
[ ]: def test_accuracy(threshold = 1):
    '''Predict distance score and classify test images as Genuine or Forged'''
    predicted_values = []
    for i in range(len(S1_test)):
        test_gen = ([S1_test[i],S2_test[i]],Y_test[i])
        (img1, img2), label = test_gen
        result = model.predict([np.expand_dims(a = img1, axis = 0), np.
        ↪expand_dims(a = img2, axis = 0)])
        diff = result[0][0]
        if diff >= threshold:
            predicted_value = 1 # both different
            if(label == predicted_value):
                predicted_values.append(True) # append True for correct value else
            ↪false
            else:
                predicted_values.append(False)
        elif(diff < threshold):
            predicted_value = 0 # both same so diff. between them is 0
```

```

    if(label == predicted_value):
        predicted_values.append(True) # append True for correct value else
        ↵false
    else:
        predicted_values.append(False)
correct_prediction = predicted_values.count(True)
return (correct_prediction/len(S1_test))*100,predicted_values

```

[]: test_acc,predicted_labels = test_accuracy(threshold = threshold)

[]: test_acc

[]: 99.91075412762159

[]: predicted_labels.count(True)/len(Y_test)*100

[]: 99.91075412762159

[]: predicted_labels.count(False)

[]: 4

0.2.1 Note: The first image is always Genuine. Score prediction and classification is done for the second image

[]: predict_score(0)

Genuine

Genuine



1/1 [=====] - 0s 25ms/step

Difference Score = 0.00031622776

Its a Genuine Signature

[]: predict_score(1)

Genuine

Forged



1/1 [=====] - 0s 22ms/step

Difference Score = 0.9999996

Its a Forged Signature

[]: predict_score(2)

Genuine

Forged



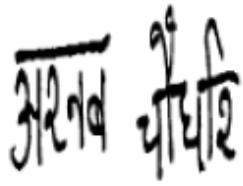
1/1 [=====] - 0s 23ms/step

Difference Score = 1.0
Its a Forged Signature

[]: predict_score(10)

Genuine

Forged

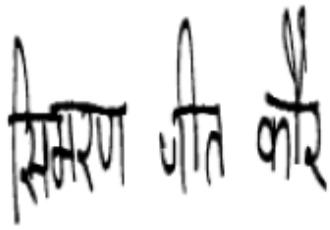


1/1 [=====] - 0s 23ms/step
Difference Score = 1.0
Its a Forged Signature

[]: predict_score(11)

Genuine

Forged



```
1/1 [=====] - 0s 24ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(13)
```

Genuine

Forged



```
1/1 [=====] - 0s 22ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(8)
```

Genuine

Forged

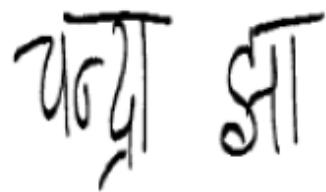


```
1/1 [=====] - 0s 24ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(19)
```

Genuine

Genuine



```
1/1 [=====] - 0s 22ms/step  
Difference Score = 0.00031622776  
Its a Genuine Signature
```

```
[ ]: predict_score(20)
```

Genuine

Forged



```
1/1 [=====] - 0s 23ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(12)
```

Genuine

Genuine



```
1/1 [=====] - 0s 23ms/step  
Difference Score = 0.00031622776  
Its a Genuine Signature
```

```
[ ]: predict_score(15)
```

Genuine

Forged



```
1/1 [=====] - 0s 24ms/step  
Difference Score = 0.99999774  
Its a Forged Signature
```

```
[ ]: predict_score(16)
```

Genuine

Forged



```
1/1 [=====] - 0s 28ms/step  
Difference Score = 0.9999939  
Its a Forged Signature
```

```
[ ]: predict_score(30)
```

Genuine

Forged

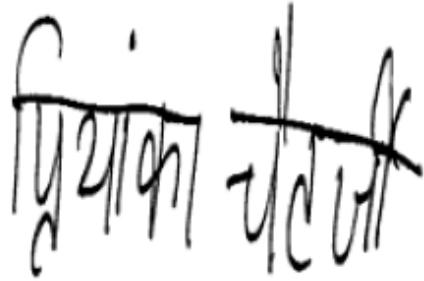
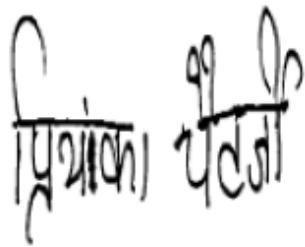


```
1/1 [=====] - 0s 74ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged

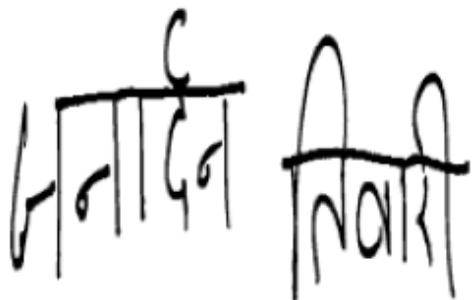


```
1/1 [=====] - 0s 40ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged

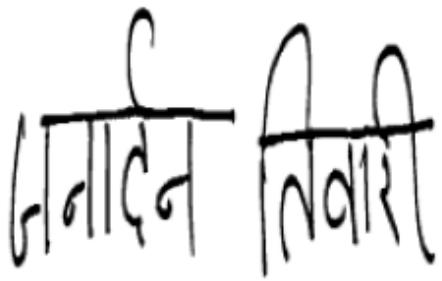


```
1/1 [=====] - 0s 27ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged

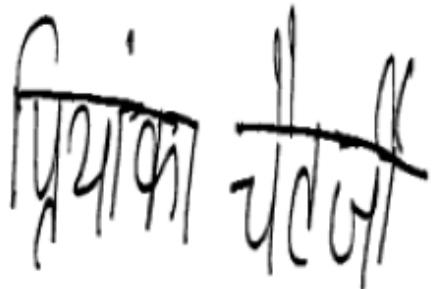


```
1/1 [=====] - 0s 29ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

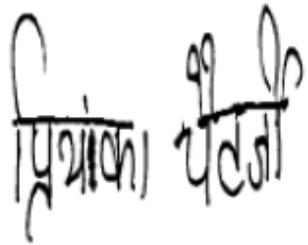
Forged



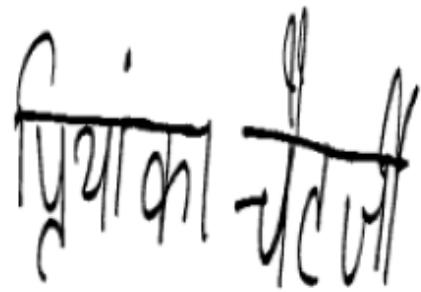
```
1/1 [=====] - 0s 27ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



```
1/1 [=====] - 0s 29ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



```
1/1 [=====] - 0s 58ms/step  
Difference Score = 0.99999815  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Genuine

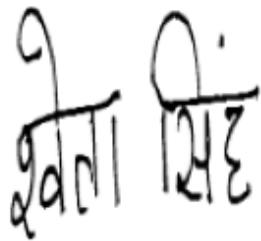


```
1/1 [=====] - 0s 69ms/step  
Difference Score = 0.00031622776  
Its a Genuine Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged

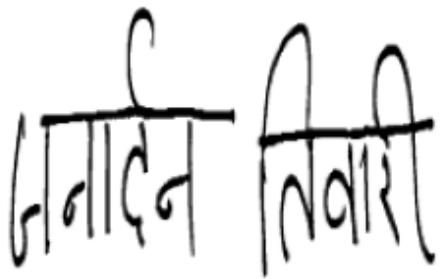


```
1/1 [=====] - 0s 109ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



```
1/1 [=====] - 0s 41ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged

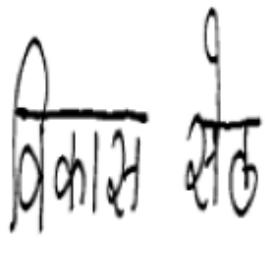


```
1/1 [=====] - 0s 27ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged

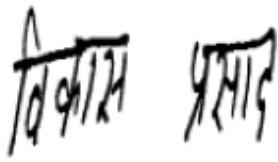


```
1/1 [=====] - 0s 27ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Genuine

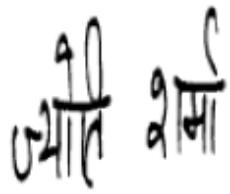


```
1/1 [=====] - 0s 48ms/step  
Difference Score = 0.00031622776  
Its a Genuine Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Genuine

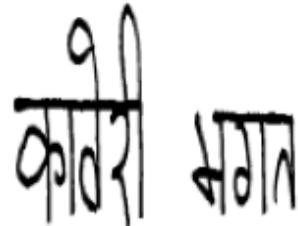
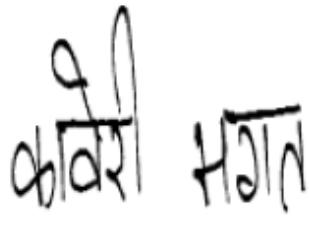


```
1/1 [=====] - 0s 27ms/step  
Difference Score = 0.00031622776  
Its a Genuine Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



```
1/1 [=====] - 0s 29ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Genuine

A handwritten signature in Devanagari script, reading "निलंदा धीर". The signature is written in a cursive style with some variations in stroke thickness.

A second handwritten signature in Devanagari script, reading "निलंदा धीर". This signature appears slightly more stylized than the first one.

```
1/1 [=====] - 0s 28ms/step  
Difference Score = 0.00031622776  
Its a Genuine Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Genuine

A handwritten signature in Devanagari script, reading "विकास सौर". The signature is written in a cursive style with some variations in stroke thickness.

A second handwritten signature in Devanagari script, reading "विकास सौर". This signature appears slightly more stylized than the first one.

```
1/1 [=====] - 0s 25ms/step  
Difference Score = 0.00031622776  
Its a Genuine Signature
```

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



आकृति अवाल



आकृति अवाल

```
1/1 [=====] - 0s 28ms/step  
Difference Score = 1.0  
Its a Forged Signature
```

```
#Model 6 (EfficientNet + Siamese)
```

```
from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
import os
from tensorflow.keras.initializers import RandomNormal, Constant
import tensorflow as tf
from tensorflow.keras.backend import max, mean, sqrt, square, sum

import seaborn as sns
from tensorflow.keras.optimizers import Adam, RMSprop
from keras.models import Model

from tensorflow import keras
from keras.layers import LeakyReLU, Softmax
from keras.layers import Conv2D, Activation,
Input, Dropout, Lambda, Flatten, Dense
from keras.layers import Dense, Flatten, Reshape, Activation
from keras.layers import BatchNormalization, ZeroPadding2D

from keras.layers import MaxPooling2D
from keras.layers import Concatenate
from keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
from keras.layers import Layer, InputSpec
from keras.regularizers import l2
from keras import backend as K
from keras.callbacks import EarlyStopping, ModelCheckpoint,
ReduceLROnPlateau
from tensorflow import one_hot, reshape

from tensorflow.keras.layers import Activation, Add, AvgPool2D,
BatchNormalization, Conv2D, Dense, Flatten, Input, MaxPool2D,
ZeroPadding2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.initializers import random_uniform,
glorot_uniform, constant
from tensorflow.keras.utils import plot_model
from keras.callbacks import EarlyStopping, ModelCheckpoint,
ReduceLROnPlateau

from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
```

```

import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model

import tensorflow as tf
print("Tensorflow version " + tf.__version__)

try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver() # TPU
    detection
    print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])
except ValueError:
    raise BaseException('ERROR: Not connected to a TPU runtime; please
    see the previous cell in this notebook for instructions!')

tf.config.experimental_connect_to_cluster(tpu)
tf.tpu.experimental.initialize_tpu_system(tpu)
tpu_strategy = tf.distribute.TPUStrategy(tpu)

Tensorflow version 2.12.0
Running on TPU  ['10.75.117.170:8470']

# import tensorflow as tf
# device_name = tf.test.gpu_device_name()
# if device_name != '/device:GPU:0':
#     raise SystemError('GPU device not found')
# print('Found GPU at: {}'.format(device_name))

from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

```

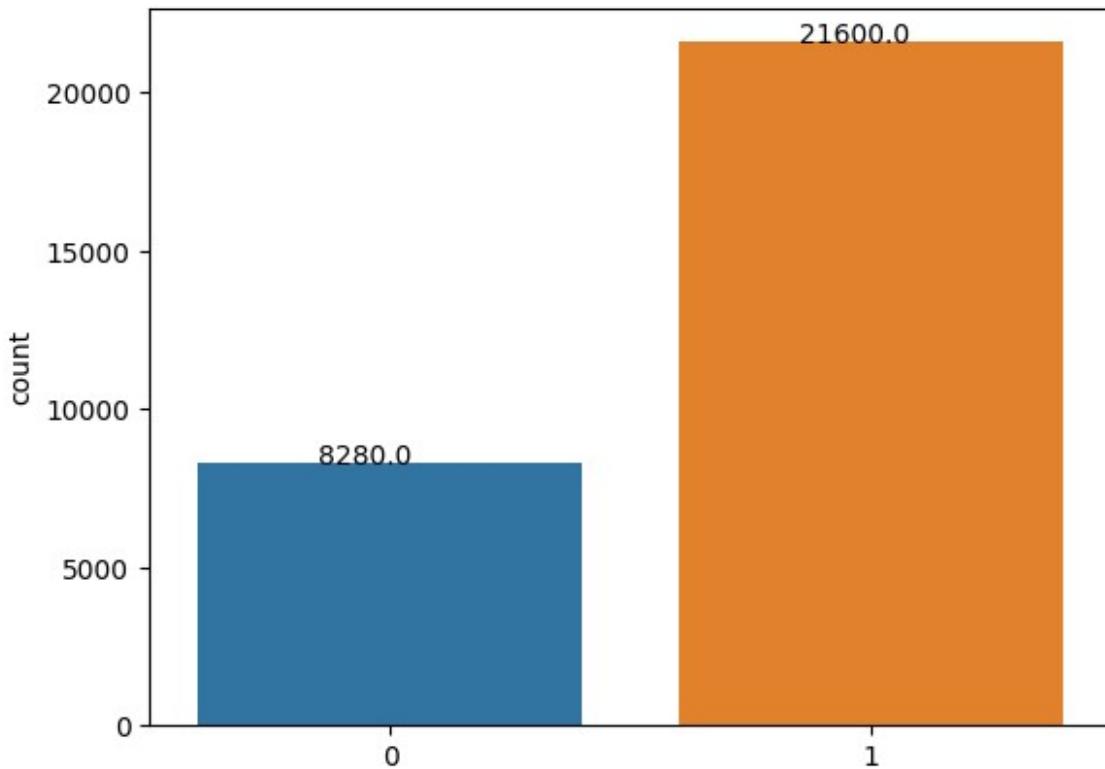
```

# with tpu_strategy.scope():
colab = True
# database = 'bigsig260_224x224_siamese_preprocessed.h5'
if colab:
    # from google.colab import drive
    # drive.mount('/content/gdrive')
    file =
'/content/drive/MyDrive/bigsig260bengali224x224x1_siamese_preprocessed
.h5'
print(file)
with File(file, 'r') as hdf:
    S1 = np.array(hdf.get('S1'))
    S2 = np.array(hdf.get('S2'))
    Y = np.array(hdf.get('Y'))
print(S1.shape)
print(S2.shape)
print(Y.shape)

/content/drive/MyDrive/
bigsig260bengali224x224x1_siamese_preprocessed.h5
(29880, 224, 224, 1)
(29880, 224, 224, 1)
(29880, 1)

ax = sns.countplot(x = Y.reshape(Y.shape[0]))
for p in ax.patches:
    ax.annotate('{:}'.format(p.get_height()), (p.get_x()+0.25,
p.get_height()+0.01))

```



$21600 / (21600 + 8280)$

0.7228915662650602

```
import sklearn
class_weights =
sklearn.utils.class_weight.compute_class_weight(class_weight =
"balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict

{0: 1.8043478260869565, 1: 0.6916666666666667}

Y = Y/1.0

S1.dtype

dtype('uint8')

seed=randint(10)
print('seed=' + str(seed))
indices = permutation(Y.shape[0])
m = int(0.70 * Y.shape[0])
n = int(0.15 * Y.shape[0])
training_id, validation_id, test_id = indices[:m], indices[m:m+n],
indices[m+n:]
S1_train, S1_test, S1_validate = S1[training_id], S1[test_id],
```

```

S1[validation_id]
S2_train, S2_test, S2_validate = S2[training_id], S2[test_id],
S2[validation_id]
Y_train, Y_test, Y_validate = Y[training_id], Y[test_id],
Y[validation_id]
print(S1_train.shape)
print(S2_train.shape)
print(Y_train.shape)

del S1,S2,Y

seed=7
(20916, 224, 224, 1)
(20916, 224, 224, 1)
(20916, 1)

fig = plt.figure(figsize = (16,16))
rows,cols = 4,4
i = 1
while(i < rows*cols):
    random_idx = randint(0,len(S1_train))
    if(Y_train[random_idx] == 0):
        Label = "Genuine Pair"
    else:
        Label = "Forged Pair"
    img1 = S1_train[random_idx]
    img2 = S2_train[random_idx]
    fig.add_subplot(rows,cols,i)
    plt.imshow(img1.squeeze(),cmap = "gray");
    plt.axis(False);
    i += 1
    fig.add_subplot(rows,cols,i)
    plt.imshow(img2.squeeze(),cmap = "gray");
    plt.axis(False);
    plt.text(0.5, 0.5, Label,
horizontalalignment='center',verticalalignment='center',fontsize=15,fontweight = 1000);
    i += 1

```

Forged Pair

ମୁଖ୍ୟା ପିଲ୍

ମୁଖ୍ୟା ପିଲ୍

Forged Pair

ଅନୁଷ୍ଠାନ ଚ

ଅନୁଷ୍ଠାନ ଚ

Genuine Pair

ଶରୀର କାହାର

ଶରୀର କାହାର

Forged Pair

କାହାର କିମ୍ବା

କାହାର କିମ୍ବା

Forged Pair

ଶରୀରକାହାରକିମ୍ବା

ଶରୀରକାହାରକିମ୍ବା

Genuine Pair

କାହାର କିମ୍ବା

କାହାର କିମ୍ବା

Forged Pair

ଶରୀର କାହାର କିମ୍ବା

ଶରୀର କାହାର କିମ୍ବା

Forged Pair

ଶରୀର କାହାର

```
Y_train[0]
```

```
array([1.])
```

```
# #One hot Encoding
```

```
# Y_train = one_hot(Y_train, depth=2)
```

```
# Y_train = reshape(Y_train, (-1, 2))
```

```
# Y_test = one_hot(Y_test, depth=2)
```

```
# Y_test = reshape(Y_test, (-1, 2))
```

```
# Y_validate = one_hot(Y_validate, depth=2)
```

```
# Y_validate = reshape(Y_validate, (-1, 2))
```

```

Y_train.shape
(20916, 1)

input_shape=(224,224,1)
input_shape

(224, 224, 1)

def euclidean_distance(vects):
    """Find the Euclidean distance between two vectors.

    Arguments:
        vects: List containing two tensors of same length.

    Returns:
        Tensor containing euclidean distance
        (as floating point value) between vectors.
    """
    x, y = vects
    sum_square = tf.math.reduce_sum(tf.math.square(x - y), axis=1,
keepdims=True)
    return tf.math.sqrt(tf.math.maximum(sum_square,
tf.keras.backend.epsilon()))

def constructive_loss(y_true, y_pred):
    '''Contrastive loss from Hadsell-et-al.'06
    http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf
    '''
    margin = 1
    square_pred = tf.math.square(y_pred)
    margin_square = tf.math.square(tf.math.maximum(margin - (y_pred),
0))
    return tf.math.reduce_mean(
        (1 - y_true) * square_pred + (y_true) * margin_square
    )

from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras import layers
NUM_CLASSES = 2
IMG_SIZE = 224
def build_model(num_classes):
    inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
    #x = img_augmentation(inputs)
    x = inputs
    model = EfficientNetB0(include_top=False, input_tensor=x,
weights="imagenet")

    # Freeze the pretrained weights
    model.trainable = False

```

```

# Rebuild top
x = layers.GlobalAveragePooling2D(name="avg_pool")(model.output)
x = layers.BatchNormalization()(x)

top_dropout_rate = 0.2
x = layers.Dropout(top_dropout_rate, name="top_dropout")(x)
outputs = layers.Dense(NUM_CLASSES, activation="relu",
name="pred")(x)

model = tf.keras.Model(inputs, outputs, name="EfficientNet")
return model
# inputs = layers.Input(shape = input_shape)
# x = inputs
# model = EfficientNetB0(include_top=False, input_tensor=x,
weights = None)

# # Freeze the pretrained weights
# model.trainable = False

# # Rebuild top
# x = layers.GlobalAveragePooling2D(name="avg_pool")(model.output)
# x = layers.BatchNormalization()(x)

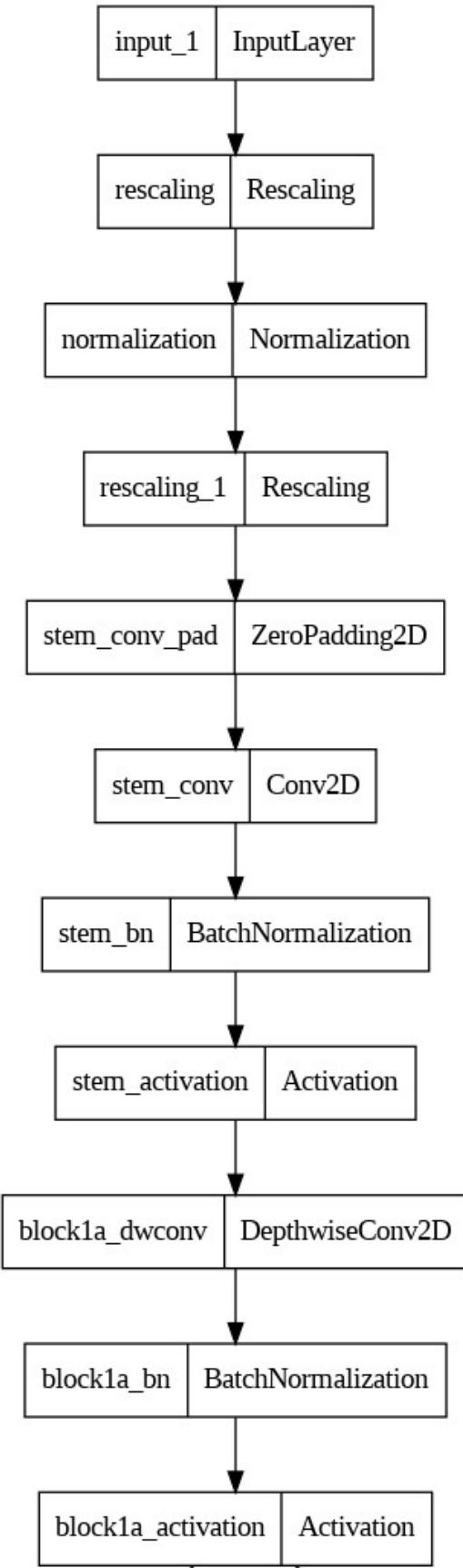
# top_dropout_rate = 0.5
# x = layers.Dropout(top_dropout_rate, name="top_dropout")(x)
# outputs = layers.Dense(num_classes, activation="sigmoid",
name="pred")(x)

# model = tf.keras.Model(inputs, outputs, name="EfficientNet")
# return model

plot_model(build_model(2))

Downloading data from https://storage.googleapis.com/keras-
applications/efficientnetb0_notop.h5
16705208/16705208 [=====] - 0s 0us/step

```



```

with tpu_strategy.scope():
# with tf.device('/device:GPU:0'):
    inputShape = S1_train.shape[1:]
    f = build_model(num_classes = 2)

def eucl_dist_output_shape(shapes):
    shape1, shape2 = shapes
    return (shape1[0], 1)

with tpu_strategy.scope():
    # network definition
    input_a = Input(shape=(input_shape))
    input_b = Input(shape=(input_shape))

    # because we re-use the same instance `base_network`,
    # the weights of the network
    # will be shared across the two branches
    processed_a = f(input_a)
    processed_b = f(input_b)

    # Compute the Euclidean distance between the two vectors in the
    latent space
    # https://keras.io/examples/vision/siamese_contrastive/
    merge_layer = layers.Lambda(euclidean_distance)([processed_a,
    processed_b])
    normal_layer = tf.keras.layers.BatchNormalization()(merge_layer)
    output_layer = layers.Dense(1, activation="sigmoid")(normal_layer)
    model = keras.Model(inputs=[input_a, input_b], outputs=output_layer)

pred = model([expand_dims(S1_train[0],axis = 0),
expand_dims(S2_train[0],axis = 0)])
print(pred.shape)
pred

(1, 1)

<tf.Tensor: shape=(1, 1), dtype=float32, numpy=array([[0.4778299]], dtype=float32)>

folder = '/content/drive/MyDrive/EfficientNetBengali224x224x1'

```

FINAL TRAINING

BEST LEARNING RATE IS 1e-04

```

Batch_size = 32
lr = 1e-04
Epochs = 60

```

```

with tpu_strategy.scope():
    adam = Adam(learning_rate = lr,epsilon = 1e-08)
    model.compile(loss = constructive_loss, optimizer = adam,
metrics=['BinaryAccuracy'])

tf.config.experimental_connect_to_cluster(tf.distribute.cluster_resolver.TPUClusterResolver())

#
tf.tpu.experimental.initialize_tpu_system(tf.distribute.cluster_resolver.TPUClusterResolver().master())

callbacks = [
    EarlyStopping(patience = 6, verbose = 1),
    ReduceLROnPlateau(factor=0.1, patience = 3, min_lr=0.000001,
verbose=1),
    ModelCheckpoint(folder + '/Weights/EffiNet-bhsig260-
{epoch:03d}.h5', verbose=1, save_weights_only=True)
]

with tpu_strategy.scope():
    results = model.fit(x = [S1_train, S2_train],
                        y = Y_train,
                        validation_data =
([S1_validate,S2_validate],Y_validate),
                        epochs = Epochs,
                        callbacks = callbacks,
                        batch_size = Batch_size,
                        class_weight = class_weight_dict,
                        steps_per_epoch = S1_train.shape[0]//Batch_size-1,
                        validation_steps =
S1_validate.shape[0]//Batch_size-1
                        # To fix Loss Nan on TPU
                        #
)
https://stackoverflow.com/questions/64079759/validation-loss-become-nan-while-training-on-tpu-but-perfectly-ok-on-gpu
)

```

```

Epoch 1/60
651/652 [=====>.] - ETA: 0s - loss: 0.2433 -
binary_accuracy: 0.5671
Epoch 1: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-001.h5
652/652 [=====] - 82s 69ms/step - loss:
0.2434 - binary_accuracy: 0.5671 - val_loss: 0.2225 -
val_binary_accuracy: 0.7030 - lr: 1.0000e-04
Epoch 2/60
651/652 [=====>.] - ETA: 0s - loss: 0.2342 -
binary_accuracy: 0.6351

```

```
Epoch 2: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-002.h5
652/652 [=====] - 38s 59ms/step - loss:
0.2342 - binary_accuracy: 0.6350 - val_loss: 0.2044 -
val_binary_accuracy: 0.7723 - lr: 1.0000e-04
Epoch 3/60
652/652 [=====] - ETA: 0s - loss: 0.2243 -
binary_accuracy: 0.6789
Epoch 3: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-003.h5
652/652 [=====] - 26s 40ms/step - loss:
0.2243 - binary_accuracy: 0.6789 - val_loss: 0.1842 -
val_binary_accuracy: 0.8051 - lr: 1.0000e-04
Epoch 4/60
652/652 [=====] - ETA: 0s - loss: 0.2165 -
binary_accuracy: 0.6958
Epoch 4: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-004.h5
652/652 [=====] - 26s 40ms/step - loss:
0.2165 - binary_accuracy: 0.6958 - val_loss: 0.1680 -
val_binary_accuracy: 0.8195 - lr: 1.0000e-04
Epoch 5/60
652/652 [=====] - ETA: 0s - loss: 0.2083 -
binary_accuracy: 0.7069
Epoch 5: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-005.h5
652/652 [=====] - 25s 39ms/step - loss:
0.2083 - binary_accuracy: 0.7069 - val_loss: 0.1523 -
val_binary_accuracy: 0.8402 - lr: 1.0000e-04
Epoch 6/60
652/652 [=====] - ETA: 0s - loss: 0.2018 -
binary_accuracy: 0.7143
Epoch 6: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-006.h5
652/652 [=====] - 25s 39ms/step - loss:
0.2018 - binary_accuracy: 0.7143 - val_loss: 0.1409 -
val_binary_accuracy: 0.8536 - lr: 1.0000e-04
Epoch 7/60
651/652 [=====>.] - ETA: 0s - loss: 0.1967 -
binary_accuracy: 0.7196
Epoch 7: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-007.h5
652/652 [=====] - 25s 39ms/step - loss:
```

```
0.1967 - binary_accuracy: 0.7196 - val_loss: 0.1291 -
val_binary_accuracy: 0.8689 - lr: 1.0000e-04
Epoch 8/60
652/652 [=====] - ETA: 0s - loss: 0.1929 -
binary_accuracy: 0.7220
Epoch 8: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-008.h5
652/652 [=====] - 26s 39ms/step - loss:
0.1929 - binary_accuracy: 0.7220 - val_loss: 0.1204 -
val_binary_accuracy: 0.8725 - lr: 1.0000e-04
Epoch 9/60
652/652 [=====] - ETA: 0s - loss: 0.1891 -
binary_accuracy: 0.7268
Epoch 9: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-009.h5
652/652 [=====] - 25s 38ms/step - loss:
0.1891 - binary_accuracy: 0.7268 - val_loss: 0.1164 -
val_binary_accuracy: 0.8739 - lr: 1.0000e-04
Epoch 10/60
651/652 [=====>.] - ETA: 0s - loss: 0.1875 -
binary_accuracy: 0.7292
Epoch 10: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-010.h5
652/652 [=====] - 25s 39ms/step - loss:
0.1874 - binary_accuracy: 0.7293 - val_loss: 0.1085 -
val_binary_accuracy: 0.8788 - lr: 1.0000e-04
Epoch 11/60
652/652 [=====] - ETA: 0s - loss: 0.1837 -
binary_accuracy: 0.7339
Epoch 11: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-011.h5
652/652 [=====] - 26s 40ms/step - loss:
0.1837 - binary_accuracy: 0.7339 - val_loss: 0.1044 -
val_binary_accuracy: 0.8811 - lr: 1.0000e-04
Epoch 12/60
652/652 [=====] - ETA: 0s - loss: 0.1820 -
binary_accuracy: 0.7377
Epoch 12: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-012.h5
652/652 [=====] - 25s 39ms/step - loss:
0.1820 - binary_accuracy: 0.7377 - val_loss: 0.0994 -
val_binary_accuracy: 0.8824 - lr: 1.0000e-04
Epoch 13/60
651/652 [=====>.] - ETA: 0s - loss: 0.1792 -
```

```
binary_accuracy: 0.7385
Epoch 13: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-013.h5
652/652 [=====] - 26s 39ms/step - loss:
0.1792 - binary_accuracy: 0.7386 - val_loss: 0.0964 -
val_binary_accuracy: 0.8849 - lr: 1.0000e-04
Epoch 14/60
651/652 [=====>.] - ETA: 0s - loss: 0.1794 -
binary_accuracy: 0.7381
Epoch 14: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-014.h5
652/652 [=====] - 25s 39ms/step - loss:
0.1794 - binary_accuracy: 0.7382 - val_loss: 0.0906 -
val_binary_accuracy: 0.8921 - lr: 1.0000e-04
Epoch 15/60
652/652 [=====] - ETA: 0s - loss: 0.1751 -
binary_accuracy: 0.7466
Epoch 15: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-015.h5
652/652 [=====] - 26s 40ms/step - loss:
0.1751 - binary_accuracy: 0.7466 - val_loss: 0.0892 -
val_binary_accuracy: 0.8898 - lr: 1.0000e-04
Epoch 16/60
651/652 [=====>.] - ETA: 0s - loss: 0.1766 -
binary_accuracy: 0.7434
Epoch 16: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-016.h5
652/652 [=====] - 26s 41ms/step - loss:
0.1765 - binary_accuracy: 0.7433 - val_loss: 0.0875 -
val_binary_accuracy: 0.8941 - lr: 1.0000e-04
Epoch 17/60
652/652 [=====] - ETA: 0s - loss: 0.1735 -
binary_accuracy: 0.7457
Epoch 17: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-017.h5
652/652 [=====] - 26s 40ms/step - loss:
0.1735 - binary_accuracy: 0.7457 - val_loss: 0.0847 -
val_binary_accuracy: 0.8948 - lr: 1.0000e-04
Epoch 18/60
652/652 [=====] - ETA: 0s - loss: 0.1744 -
binary_accuracy: 0.7485
Epoch 18: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-018.h5
```

```
652/652 [=====] - 30s 46ms/step - loss:  
0.1744 - binary_accuracy: 0.7485 - val_loss: 0.0852 -  
val_binary_accuracy: 0.8950 - lr: 1.0000e-04  
Epoch 19/60  
651/652 [=====.>.] - ETA: 0s - loss: 0.1751 -  
binary_accuracy: 0.7474  
Epoch 19: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-019.h5  
652/652 [=====] - 28s 43ms/step - loss:  
0.1751 - binary_accuracy: 0.7472 - val_loss: 0.0794 -  
val_binary_accuracy: 0.9049 - lr: 1.0000e-04  
Epoch 20/60  
651/652 [=====.>.] - ETA: 0s - loss: 0.1735 -  
binary_accuracy: 0.7426  
Epoch 20: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-020.h5  
652/652 [=====] - 25s 39ms/step - loss:  
0.1736 - binary_accuracy: 0.7427 - val_loss: 0.0785 -  
val_binary_accuracy: 0.9060 - lr: 1.0000e-04  
Epoch 21/60  
651/652 [=====.>.] - ETA: 0s - loss: 0.1740 -  
binary_accuracy: 0.7418  
Epoch 21: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-021.h5  
652/652 [=====] - 26s 40ms/step - loss:  
0.1740 - binary_accuracy: 0.7418 - val_loss: 0.0805 -  
val_binary_accuracy: 0.8986 - lr: 1.0000e-04  
Epoch 22/60  
652/652 [=====] - ETA: 0s - loss: 0.1700 -  
binary_accuracy: 0.7499  
Epoch 22: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-022.h5  
652/652 [=====] - 27s 41ms/step - loss:  
0.1700 - binary_accuracy: 0.7499 - val_loss: 0.0758 -  
val_binary_accuracy: 0.9143 - lr: 1.0000e-04  
Epoch 23/60  
651/652 [=====.>.] - ETA: 0s - loss: 0.1712 -  
binary_accuracy: 0.7530  
Epoch 23: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-023.h5  
652/652 [=====] - 26s 39ms/step - loss:  
0.1711 - binary_accuracy: 0.7532 - val_loss: 0.0746 -  
val_binary_accuracy: 0.9089 - lr: 1.0000e-04  
Epoch 24/60
```

```
651/652 [=====>.] - ETA: 0s - loss: 0.1700 -  
binary_accuracy: 0.7501  
Epoch 24: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-024.h5  
652/652 [=====] - 26s 40ms/step - loss:  
0.1700 - binary_accuracy: 0.7501 - val_loss: 0.0740 -  
val_binary_accuracy: 0.9101 - lr: 1.0000e-04  
Epoch 25/60  
652/652 [=====] - ETA: 0s - loss: 0.1713 -  
binary_accuracy: 0.7523  
Epoch 25: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-025.h5  
652/652 [=====] - 26s 39ms/step - loss:  
0.1713 - binary_accuracy: 0.7523 - val_loss: 0.0739 -  
val_binary_accuracy: 0.9105 - lr: 1.0000e-04  
Epoch 26/60  
651/652 [=====>.] - ETA: 0s - loss: 0.1699 -  
binary_accuracy: 0.7515  
Epoch 26: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-026.h5  
652/652 [=====] - 26s 40ms/step - loss:  
0.1700 - binary_accuracy: 0.7515 - val_loss: 0.0706 -  
val_binary_accuracy: 0.9166 - lr: 1.0000e-04  
Epoch 27/60  
652/652 [=====] - ETA: 0s - loss: 0.1701 -  
binary_accuracy: 0.7509  
Epoch 27: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-027.h5  
652/652 [=====] - 26s 40ms/step - loss:  
0.1701 - binary_accuracy: 0.7509 - val_loss: 0.0694 -  
val_binary_accuracy: 0.9209 - lr: 1.0000e-04  
Epoch 28/60  
652/652 [=====] - ETA: 0s - loss: 0.1710 -  
binary_accuracy: 0.7496  
Epoch 28: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-028.h5  
652/652 [=====] - 26s 40ms/step - loss:  
0.1710 - binary_accuracy: 0.7496 - val_loss: 0.0679 -  
val_binary_accuracy: 0.9236 - lr: 1.0000e-04  
Epoch 29/60  
652/652 [=====] - ETA: 0s - loss: 0.1680 -  
binary_accuracy: 0.7544  
Epoch 29: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
```

```
bhsig260-029.h5
652/652 [=====] - 25s 39ms/step - loss:
0.1680 - binary_accuracy: 0.7544 - val_loss: 0.0690 -
val_binary_accuracy: 0.9168 - lr: 1.0000e-04
Epoch 30/60
652/652 [=====] - ETA: 0s - loss: 0.1694 -
binary_accuracy: 0.7516
Epoch 30: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-030.h5
652/652 [=====] - 25s 39ms/step - loss:
0.1694 - binary_accuracy: 0.7516 - val_loss: 0.0688 -
val_binary_accuracy: 0.9161 - lr: 1.0000e-04
Epoch 31/60
651/652 [=====>.] - ETA: 0s - loss: 0.1654 -
binary_accuracy: 0.7580
Epoch 31: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-031.h5
652/652 [=====] - 25s 39ms/step - loss:
0.1655 - binary_accuracy: 0.7579 - val_loss: 0.0650 -
val_binary_accuracy: 0.9249 - lr: 1.0000e-04
Epoch 32/60
652/652 [=====] - ETA: 0s - loss: 0.1691 -
binary_accuracy: 0.7523
Epoch 32: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-032.h5
652/652 [=====] - 26s 40ms/step - loss:
0.1691 - binary_accuracy: 0.7523 - val_loss: 0.0658 -
val_binary_accuracy: 0.9258 - lr: 1.0000e-04
Epoch 33/60
651/652 [=====>.] - ETA: 0s - loss: 0.1676 -
binary_accuracy: 0.7564
Epoch 33: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-033.h5
652/652 [=====] - 26s 40ms/step - loss:
0.1676 - binary_accuracy: 0.7563 - val_loss: 0.0644 -
val_binary_accuracy: 0.9272 - lr: 1.0000e-04
Epoch 34/60
651/652 [=====>.] - ETA: 0s - loss: 0.1660 -
binary_accuracy: 0.7595
Epoch 34: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-034.h5
652/652 [=====] - 26s 40ms/step - loss:
0.1661 - binary_accuracy: 0.7593 - val_loss: 0.0604 -
val_binary_accuracy: 0.9321 - lr: 1.0000e-04
```

```
Epoch 35/60
652/652 [=====] - ETA: 0s - loss: 0.1655 -
binary_accuracy: 0.7584
Epoch 35: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-035.h5
652/652 [=====] - 26s 41ms/step - loss:
0.1655 - binary_accuracy: 0.7584 - val_loss: 0.0615 -
val_binary_accuracy: 0.9265 - lr: 1.0000e-04
Epoch 36/60
652/652 [=====] - ETA: 0s - loss: 0.1660 -
binary_accuracy: 0.7564
Epoch 36: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-036.h5
652/652 [=====] - 27s 42ms/step - loss:
0.1660 - binary_accuracy: 0.7564 - val_loss: 0.0611 -
val_binary_accuracy: 0.9290 - lr: 1.0000e-04
Epoch 37/60
651/652 [=====>.] - ETA: 0s - loss: 0.1648 -
binary_accuracy: 0.7601
Epoch 37: ReduceLROnPlateau reducing learning rate to
9.99999747378752e-06.

Epoch 37: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-037.h5
652/652 [=====] - 27s 41ms/step - loss:
0.1648 - binary_accuracy: 0.7601 - val_loss: 0.0616 -
val_binary_accuracy: 0.9287 - lr: 1.0000e-04
Epoch 38/60
651/652 [=====>.] - ETA: 0s - loss: 0.1662 -
binary_accuracy: 0.7569
Epoch 38: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-038.h5
652/652 [=====] - 26s 40ms/step - loss:
0.1663 - binary_accuracy: 0.7570 - val_loss: 0.0611 -
val_binary_accuracy: 0.9281 - lr: 1.0000e-05
Epoch 39/60
651/652 [=====>.] - ETA: 0s - loss: 0.1665 -
binary_accuracy: 0.7549
Epoch 39: saving model to
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-
bhsig260-039.h5
652/652 [=====] - 26s 40ms/step - loss:
0.1665 - binary_accuracy: 0.7550 - val_loss: 0.0606 -
val_binary_accuracy: 0.9294 - lr: 1.0000e-05
Epoch 40/60
```

```
651/652 [=====.>.] - ETA: 0s - loss: 0.1641 -  
binary_accuracy: 0.7602  
Epoch 40: ReduceLROnPlateau reducing learning rate to 1e-06.  
  
Epoch 40: saving model to  
/content/drive/MyDrive/EfficientNetBengali224x224x1/Weights/EffiNet-  
bhsig260-040.h5  
652/652 [=====] - 26s 40ms/step - loss:  
0.1640 - binary_accuracy: 0.7603 - val_loss: 0.0609 -  
val_binary_accuracy: 0.9287 - lr: 1.0000e-05  
Epoch 40: early stopping
```

Evaluation

```
import pickle  
  
# with open(folder + '/trainHistoryDict.h5', 'wb') as file_pi:  
#     pickle.dump(results.history, file_pi)  
# # model.save(folder + 'BestModel.h5')  
  
history_loaded = pickle.load(open(folder + '/trainHistoryDict.h5',  
"rb"))  
# history_loaded  
  
-----  
----  
EOFError                                         Traceback (most recent call  
last)  
<ipython-input-28-0f838720ffda> in <cell line: 1>()  
----> 1 history_loaded = pickle.load(open(folder +  
'/trainHistoryDict.h5', "rb"))  
      2 # history_loaded  
  
EOFError: Ran out of input  
  
# history_loaded = results  
  
best_result =  
history_loaded['val_loss'].index(min(history_loaded['val_loss'])) + 1  
best_result  
  
def display_training_curves(training, validation, title, subplot):  
    ax = plt.subplot(subplot)  
    ax.plot(training)  
    ax.plot(validation)  
    ax.set_title('model ' + title)  
    ax.set_ylabel(title)  
    ax.set_xlabel('epoch')  
    ax.legend(['training', 'validation'])
```

```

plt.subplots(figsize=(10,10))
plt.tight_layout()
display_training_curves(history_loaded['binary_accuracy'],
history_loaded['val_binary_accuracy'], 'accuracy', 211)
display_training_curves(history_loaded['loss'],
history_loaded['val_loss'], 'loss', 212)
plt.savefig(folder + "loss_accuracy.svg",dpi = 1200)

if(best_result < 10):
    model.load_weights(folder + f'/Weights/EffiNet-bhsig260-
00{best_result}.h5')
else:
    model.load_weights(folder + f'/Weights/EffiNet-bhsig260-
0{best_result}.h5')

y_pred_keras = model.predict([S1_test, S2_test])

y_pred_keras
-----
-----
NameError                               Traceback (most recent call
last)
<ipython-input-30-7488c800ccb8> in <cell line: 1>()
----> 1 y_pred_keras

NameError: name 'y_pred_keras' is not defined

np.min(y_pred_keras),np.mean(y_pred_keras),np.max(y_pred_keras)

# # if we keep threshold as fixed to the mean value
# threshold = 0.755

def pred_threshold(y_pred_keras,threshold):
    y_pred_keras_new = np.zeros((len(y_pred_keras), 1), dtype =
np.uint8)
    for i in range(len(y_pred_keras)):
        label = Y_test[i]
        if y_pred_keras[i] >= threshold:
            predicted_value = 1 # both different
            if(label == predicted_value):
                y_pred_keras_new[i] = label
            else:
                y_pred_keras_new[i] = predicted_value
        else:
            predicted_value = 0
            if(label == predicted_value):
                y_pred_keras_new[i] = label
            else:

```

```

        y_pred_keras_new[i] = predicted_value
    return y_pred_keras_new

from sklearn.metrics import auc
from sklearn.metrics import roc_curve
threshold_areas = {}
for i in np.arange(0.5,0.8,0.001):
    y_pred_keras_new = pred_threshold(y_pred_keras,i)
    fpr_keras, tpr_keras, thresholds_keras = roc_curve(Y_test,
y_pred_keras_new)
    auc_keras = auc(fpr_keras, tpr_keras)
    threshold_areas[i] = auc_keras
    # print(i,auc_keras)
print(threshold_areas)
max_value = list(threshold_areas.values())
max_key= list(threshold_areas.keys())
threshold = max_key[max_value.index(max(max_value))]
y_pred_keras_new = pred_threshold(y_pred_keras,threshold)
fpr_keras, tpr_keras, thresholds_keras = roc_curve(Y_test,
y_pred_keras_new)
auc_keras = auc(fpr_keras, tpr_keras)
print("False Positive Rate : ",fpr_keras,"True Positive
Rate :",tpr_keras)
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f},threshold =
{:.3f})'.format(auc_keras,threshold))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.svg",dpi = 1200)
plt.show()

{0.5: 0.9116369552409458, 0.501: 0.9118782346521059, 0.502:
0.9118782346521059, 0.503: 0.9117230033574768, 0.504:
0.9115677720628479, 0.505: 0.9127573041802151, 0.506:
0.9129985835913751, 0.507: 0.9136363737083242, 0.508:
0.9137224218248553, 0.509: 0.9139637012360153, 0.51:
0.9161875341761206, 0.511: 0.9160323028814916, 0.512:
0.9168253242930697, 0.513: 0.9176183457046478, 0.514:
0.9176183457046478, 0.515: 0.9176183457046478, 0.516:
0.9176183457046478, 0.517: 0.9180148564104369, 0.518:
0.918411367116226, 0.519: 0.9196008992335931, 0.52:
0.9207904313509603, 0.521: 0.9207904313509603, 0.522:
0.9201695061724444, 0.523: 0.9198590435831864, 0.524:
0.9204968337001355, 0.525: 0.9204968337001355, 0.526:
0.9204276505220375, 0.527: 0.9216171826394046, 0.528:
0.9220136933451937, 0.529: 0.9222549727563538, 0.53:

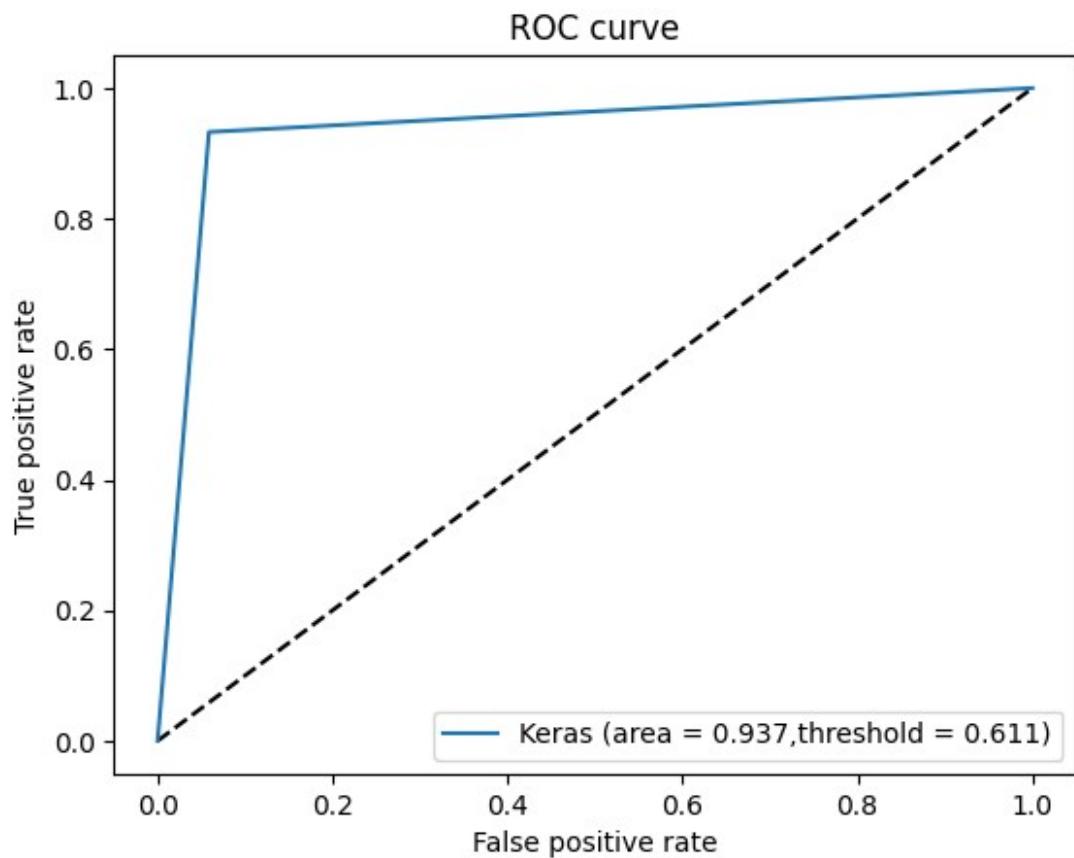
```

0.9219445101670959, 0.531: 0.9219445101670959, 0.532:
0.921789278872467, 0.533: 0.922427068989416, 0.534: 0.922271837694787,
0.535: 0.9229096278117361, 0.536: 0.9243404393402633, 0.537:
0.9247369500460523, 0.538: 0.9247369500460523, 0.539:
0.9255299714576305, 0.54: 0.9255299714576305, 0.541:
0.9259264821634194, 0.542: 0.9271160142807866, 0.543:
0.9269607829861578, 0.544: 0.9272020623973178, 0.545:
0.9270468311026887, 0.546: 0.9274433418084779, 0.547:
0.9274433418084779, 0.548: 0.9274433418084779, 0.549:
0.9274433418084779, 0.55: 0.9276846212196378, 0.551:
0.9275293899250089, 0.552: 0.9275293899250089, 0.553:
0.927218927335751, 0.554: 0.928011948747329, 0.555:
0.9278567174526999, 0.556: 0.9281840449803912, 0.557:
0.9281840449803912, 0.558: 0.9281840449803912, 0.559:
0.9284253243915511, 0.56: 0.9284253243915511, 0.561:
0.9281148618022933, 0.562: 0.9291491626250312, 0.5630000000000001:
0.9292352107415625, 0.5640000000000001: 0.9290799794469334,
0.5650000000000001: 0.9297177695638826, 0.5660000000000001:
0.9311485810924097, 0.5670000000000001: 0.9311485810924097,
0.5680000000000001: 0.9315450917981988, 0.5690000000000001:
0.9315450917981988, 0.5700000000000001: 0.9312346292089408,
0.5710000000000001: 0.932665440737468, 0.5720000000000001:
0.9325102094428391, 0.5730000000000001: 0.9323549781482101,
0.5740000000000001: 0.931889284264323, 0.5750000000000001:
0.9317340529696941, 0.5760000000000001: 0.9321305636754832,
0.5770000000000001: 0.9321305636754832, 0.5780000000000001:
0.9322166117920142, 0.5790000000000001: 0.9320613804973853,
0.5800000000000001: 0.9319061492027563, 0.5810000000000001:
0.9319921973192873, 0.5820000000000001: 0.9318369660246584,
0.5830000000000001: 0.9320782454358184, 0.5840000000000001:
0.9320782454358184, 0.5850000000000001: 0.9324747561416075,
0.5860000000000001: 0.9321642935523495, 0.5870000000000001:
0.9321642935523495, 0.5880000000000001: 0.9325608042581385,
0.5890000000000001: 0.9329573149639275, 0.5900000000000001:
0.9328020836692985, 0.5910000000000001: 0.9335951050808766,
0.5920000000000001: 0.9334398737862476, 0.5930000000000001:
0.9338363844920367, 0.5940000000000001: 0.9339224326085677,
0.5950000000000001: 0.9339224326085677, 0.5960000000000001:
0.9341637120197279, 0.5970000000000001: 0.9340945288416299,
0.5980000000000001: 0.9341805769581609, 0.5990000000000001:
0.9340253456635319, 0.6000000000000001: 0.9340253456635319,
0.6010000000000001: 0.9338701143689029, 0.6020000000000001:
0.934663135780481, 0.6030000000000001: 0.934663135780481,
0.6040000000000001: 0.934507904485852, 0.6050000000000001:
0.9356974366032192, 0.6060000000000001: 0.9355422053085902,
0.6070000000000001: 0.9355422053085902, 0.6080000000000001:
0.9353869740139612, 0.6090000000000001: 0.9357834847197503,
0.6100000000000001: 0.9365765061313284, 0.6110000000000001:
0.9369730168371174, 0.6120000000000001: 0.9369730168371174,

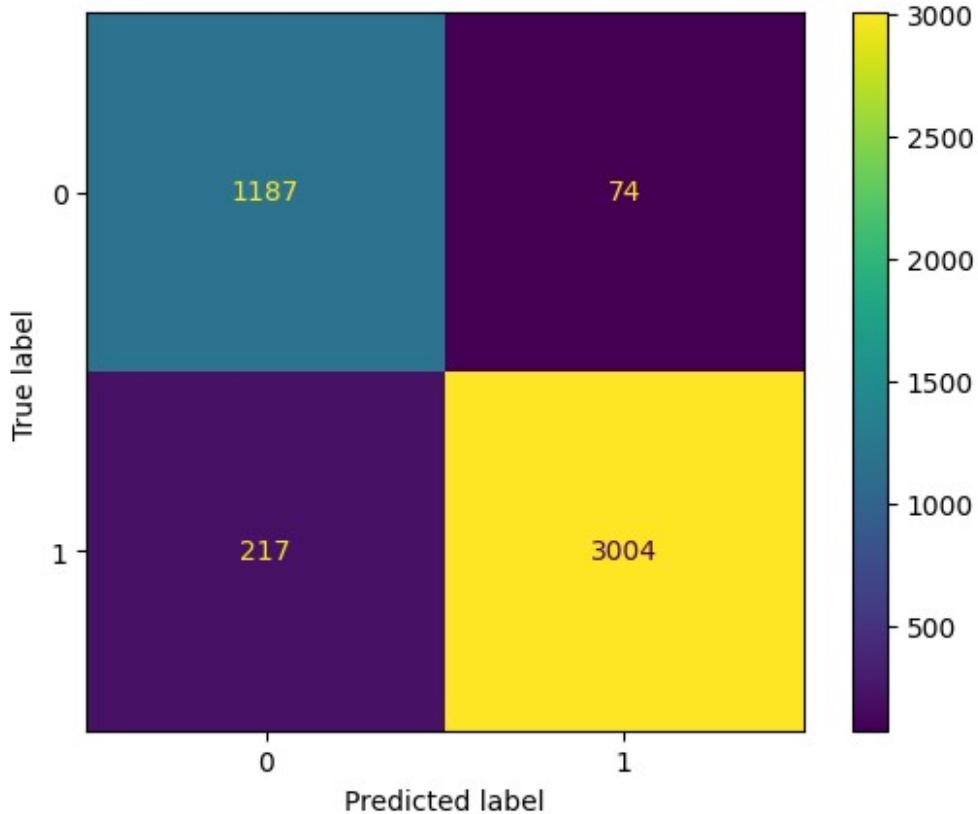
0.6130000000000001: 0.9368177855424885, 0.6140000000000001:
0.9368177855424885, 0.6150000000000001: 0.9357311664800855,
0.6160000000000001: 0.9357311664800855, 0.6170000000000001:
0.9365241878916635, 0.6180000000000001: 0.9365241878916635,
0.6190000000000001: 0.9363689565970346, 0.6200000000000001:
0.9364550047135657, 0.6210000000000001: 0.9362997734189366,
0.6220000000000001: 0.9361445421243076, 0.6230000000000001:
0.9361445421243076, 0.6240000000000001: 0.9361445421243076,
0.6250000000000001: 0.9361445421243076, 0.6260000000000001:
0.9361445421243076, 0.6270000000000001: 0.9359893108296786,
0.6280000000000001: 0.9358340795350496, 0.6290000000000001:
0.9355236169457917, 0.6300000000000001: 0.9355236169457917,
0.6310000000000001: 0.9355236169457917, 0.6320000000000001:
0.9353683856511625, 0.6330000000000001: 0.9331951475263567,
0.6340000000000001: 0.9334364269375167, 0.6350000000000001:
0.9331259643482587, 0.6360000000000001: 0.9329707330536298,
0.6370000000000001: 0.9333672437594188, 0.6380000000000001:
0.9333672437594188, 0.6390000000000001: 0.9327463185809028,
0.6400000000000001: 0.9325910872862738, 0.6410000000000001:
0.9316596995184999, 0.6420000000000001: 0.9319009789296598,
0.6430000000000001: 0.9315905163404019, 0.6440000000000001:
0.9315905163404019, 0.6450000000000001: 0.9314352850457729,
0.6460000000000001: 0.933021327868929, 0.6470000000000001:
0.933021327868929, 0.6480000000000001: 0.9331073759854601,
0.6490000000000001: 0.9339864455135694, 0.6500000000000001:
0.9343829562193584, 0.6510000000000001: 0.9343829562193584,
0.6520000000000001: 0.9340724936301005, 0.6530000000000001:
0.9340724936301005, 0.6540000000000001: 0.9340724936301005,
0.6550000000000001: 0.9344690043358895, 0.6560000000000001:
0.9346411005689517, 0.6570000000000001: 0.9346411005689517,
0.6580000000000001: 0.9343306379796937, 0.6590000000000001:
0.9347271486854828, 0.6600000000000001: 0.9349684280966428,
0.6610000000000001: 0.9348131968020138, 0.6620000000000001:
0.9348131968020138, 0.6630000000000001: 0.9346579655073848,
0.6640000000000001: 0.9345027342127559, 0.6650000000000001:
0.9345027342127559, 0.6660000000000001: 0.9337265777396108,
0.6670000000000001: 0.9334161151503527, 0.6680000000000001:
0.9334161151503527, 0.6690000000000002: 0.9334161151503527,
0.6700000000000002: 0.9335021632668838, 0.6710000000000002:
0.933743442678044, 0.6720000000000002: 0.934139953383833,
0.6730000000000002: 0.9336742594999461, 0.6740000000000002:
0.9336742594999461, 0.6750000000000002: 0.933519028205317,
0.6760000000000002: 0.933208565616059, 0.6770000000000002:
0.932432409142914, 0.6780000000000002: 0.9319667152590271,
0.6790000000000002: 0.9310353274912531, 0.6800000000000002:
0.9308800961966242, 0.6810000000000002: 0.9308800961966242,
0.6820000000000002: 0.930724864901995, 0.6830000000000002:
0.930569633607366, 0.6840000000000002: 0.9308109130185261,
0.6850000000000002: 0.9312074237243152, 0.6860000000000002:

0.9310521924296861, 0.6870000000000002: 0.9314487031354751,
0.6880000000000002: 0.9316899825466354, 0.6890000000000002:
0.9316899825466354, 0.6900000000000002: 0.9317760306631663,
0.6910000000000002: 0.9316207993685373, 0.6920000000000002:
0.9316207993685373, 0.6930000000000002: 0.9316207993685373,
0.6940000000000002: 0.9318620787796974, 0.6950000000000002:
0.9309306910119235, 0.6960000000000002: 0.9304649971280364,
0.6970000000000002: 0.9304649971280364, 0.6980000000000002:
0.9303097658334074, 0.6990000000000002: 0.9303097658334074,
0.7000000000000002: 0.9303097658334074, 0.7010000000000002:
0.9303097658334074, 0.7020000000000002: 0.9305510452445674,
0.7030000000000002: 0.9302405826553095, 0.7040000000000002:
0.9307231414776296, 0.7050000000000002: 0.9313609315945787,
0.7060000000000002: 0.9302743125321756, 0.7070000000000002:
0.9296533873536598, 0.7080000000000002: 0.9294981560590307,
0.7090000000000002: 0.9288772308805148, 0.7100000000000002:
0.9286528164077879, 0.7110000000000002: 0.9281871225239008,
0.7120000000000002: 0.9280318912292718, 0.7130000000000002:
0.9274109660507558, 0.7140000000000002: 0.9274109660507558,
0.7150000000000002: 0.92547900733711, 0.7160000000000002:
0.92547900733711, 0.7170000000000002: 0.92547900733711,
0.7180000000000002: 0.92572028674827, 0.7190000000000002:
0.92572028674827, 0.7200000000000002: 0.92572028674827,
0.7210000000000002: 0.9254098241590121, 0.7220000000000002:
0.9258063348648011, 0.7230000000000002: 0.9256511035701721,
0.7240000000000002: 0.9254958722755431, 0.7250000000000002:
0.925185409686285, 0.7260000000000002: 0.9247197158023981,
0.7270000000000002: 0.9247197158023981, 0.7280000000000002:
0.9246505326243002, 0.7290000000000002: 0.9250470433300891,
0.7300000000000002: 0.9254435540358782, 0.7310000000000002:
0.9258400647416672, 0.7320000000000002: 0.9258400647416672,
0.7330000000000002: 0.9258400647416672, 0.7340000000000002:
0.9260813441528274, 0.7350000000000002: 0.9257708815635695,
0.7360000000000002: 0.9261673922693585, 0.7370000000000002:
0.9255464670908423, 0.7380000000000002: 0.9252360045015845,
0.7390000000000002: 0.9252360045015845, 0.7400000000000002:
0.9249255419123266, 0.7410000000000002: 0.9253220526181155,
0.7420000000000002: 0.9253220526181155, 0.7430000000000002:
0.9247011274395995, 0.7440000000000002: 0.9250976381453885,
0.7450000000000002: 0.9254941488511775, 0.7460000000000002:
0.9251836862619195, 0.7470000000000002: 0.9251836862619195,
0.7480000000000002: 0.9255801969677085, 0.7490000000000002:
0.9246488091999346, 0.7500000000000002: 0.9227860336643866,
0.7510000000000002: 0.9220098771912416, 0.7520000000000002:
0.9216302314238857, 0.7530000000000002: 0.921475001292568,
0.7540000000000002: 0.9208540749507409, 0.7550000000000002:
0.9210953543619008, 0.7560000000000002: 0.9210953543619008,
0.7570000000000002: 0.9206296604780139, 0.7580000000000002:
0.9204744291833848, 0.7590000000000002: 0.9200087352994978,

0.7600000000000002: 0.9196982727102399, 0.7610000000000002:
0.9192325788263528, 0.7620000000000002: 0.9186116536478369,
0.7630000000000002: 0.9181459597639499, 0.7640000000000002:
0.9175250345854339, 0.7650000000000002: 0.916438415523031,
0.7660000000000002: 0.916438415523031, 0.7670000000000002:
0.9161279529337729, 0.7680000000000002: 0.915507027755257,
0.7690000000000002: 0.915507027755257, 0.7700000000000002:
0.914575639987483, 0.7710000000000002: 0.9141099461035961,
0.7720000000000002: 0.914506456809385, 0.7730000000000002:
0.914351225514756, 0.7740000000000002: 0.914437273631287,
0.7750000000000002: 0.9141268110420292, 0.7760000000000002:
0.9130401919796262, 0.7770000000000002: 0.9125744980957391,
0.7780000000000002: 0.9125744980957391, 0.7790000000000002:
0.9125744980957391, 0.7800000000000002: 0.9116431103279652,
0.7810000000000002: 0.9104012599709332, 0.7820000000000003:
0.9104012599709332, 0.7830000000000003: 0.9088489470246433,
0.7840000000000003: 0.9086937157300142, 0.7850000000000003:
0.9085384844353851, 0.7860000000000003: 0.9072966340783531,
0.7870000000000003: 0.9069861714890952, 0.7880000000000003:
0.9062100150159503, 0.7890000000000003: 0.9054338585428052,
0.7900000000000003: 0.9054338585428052, 0.7910000000000003:
0.9058303692485944, 0.7920000000000003: 0.9050542127754494,
0.7930000000000003: 0.9028809746506434, 0.7940000000000003:
0.9028809746506434, 0.7950000000000003: 0.9019495868828695,
0.7960000000000003: 0.9011734304097245, 0.7970000000000003:
0.9010181991150954, 0.7980000000000003: 0.9007077365258374,
0.7990000000000003: 0.9005525052312083, 0.8000000000000003:
0.9002420426419504}



```
ConfusionMatrixDisplay(confusion_matrix(Y_test,y_pred_keras_new)).plot()
plt.savefig(folder + "confusionmatrix.svg",dpi = 1200)
plt.show()
```



Load the weights from the epoch which gave the best validation accuracy

```
i = 0
test_gen = ([S1_test[i],S2_test[i]],Y_test[i])
(img1, img2), label = test_gen
result = model([np.expand_dims(a = img1, axis = 0), np.expand_dims(a = img2, axis = 0)])
result.numpy()[0][0]

0.59371305

def predict_score(i = 0):
    '''Predict distance score and classify test images as Genuine or Forged'''
    test_gen = ([S1_test[i],S2_test[i]],Y_test[i])
    (img1, img2), label = test_gen

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 10))
    ax1.imshow(np.squeeze(img1), cmap='gray')
    ax2.imshow(np.squeeze(img2), cmap='gray')
    ax1.set_title('Genuine')
    if label == 0:
        ax2.set_title('Genuine')
    else:
```

```

        ax2.set_title('Forged')
ax1.axis('off')
ax2.axis('off')
plt.show()
result = model([np.expand_dims(a = img1, axis = 0),
np.expand_dims(a = img2, axis = 0)])
diff = result.numpy()[0][0]
print("Difference Score = ", diff)
if diff >= threshold:
    print("Its a Forged Signature")
else:
    print("Its a Genuine Signature")

def test_accuracy():
    '''Predict distance score and classify test images as Genuine or
Forged'''
    predicted_labels = []
    # y_pred_keras_new
    for i in range(len(S1_test)):
        test_gen = ([S1_test[i], S2_test[i]], Y_test[i])
        (img1, img2), label = test_gen
        if(label == y_pred_keras_new[i]):
            predicted_labels.append(True) # append True for correct value
    else false
    else:
        predicted_labels.append(False)
    correct_prediction = predicted_labels.count(True)
    return (correct_prediction/len(S1_test))*100, predicted_labels

with tpu_strategy.scope():
    test_acc, predicted_labels = test_accuracy()

test_acc
93.50736278447121
predicted_labels.count(True)/len(predicted_labels)*100
93.50736278447121
predicted_labels.count(False)
291

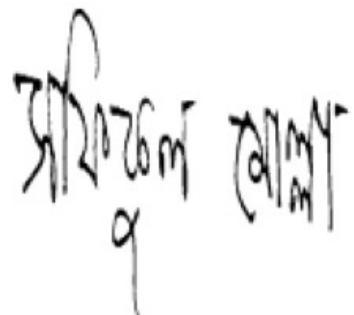
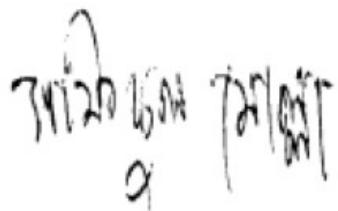
```

Note: The first image is always Genuine. Score prediction and classification is done for the second image

```
predict_score(0)
```

Genuine

Forged

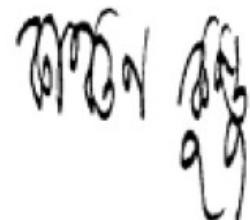


```
Difference Score = 0.59371305  
Its a Genuine Signature
```

```
predict_score(1)
```

Genuine

Forged

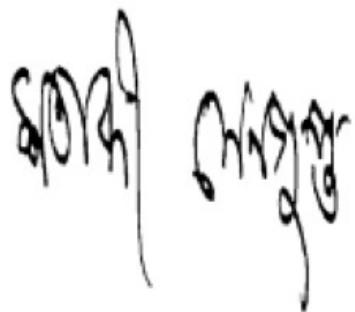


```
Difference Score = 0.72356164  
Its a Forged Signature
```

```
predict_score(2)
```

Genuine

Forged



Difference Score = 0.9459486
Its a Forged Signature

`predict_score(10)`

Genuine

Forged



Difference Score = 0.9498749
Its a Forged Signature

`predict_score(11)`

Genuine

Forged

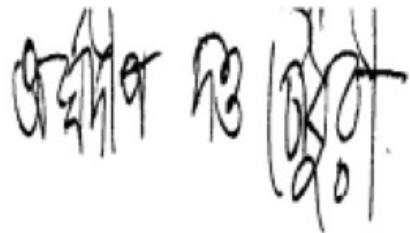


Difference Score = 0.90118676
Its a Forged Signature

`predict_score(13)`

Genuine

Forged



Difference Score = 0.9917966
Its a Forged Signature

`predict_score(8)`

Genuine



Forged



Difference Score = 0.8372152
Its a Forged Signature

`predict_score(19)`

Genuine



Forged



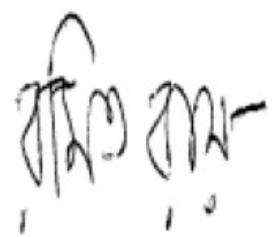
Difference Score = 0.99398065
Its a Forged Signature

`predict_score(20)`

Genuine



Genuine



Difference Score = 0.25984165
Its a Genuine Signature

`predict_score(12)`

Genuine



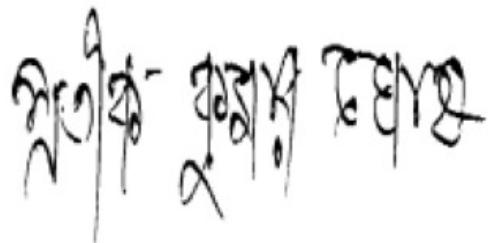
Forged



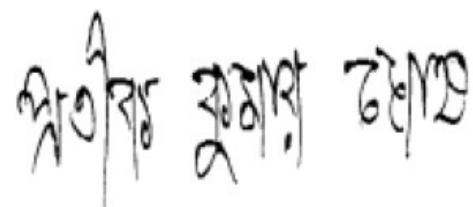
Difference Score = 0.9190464
Its a Forged Signature

`predict_score(15)`

Genuine

A handwritten signature in black ink on white paper. The characters are fluid and somewhat stylized, appearing to be in a cursive script.

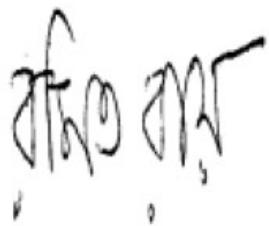
Genuine

A second handwritten signature in black ink on white paper, matching the style of the first one.

Difference Score = 0.23682731
Its a Genuine Signature

`predict_score(16)`

Genuine

A handwritten signature in black ink on white paper, appearing to be in a cursive script.

Genuine

A second handwritten signature in black ink on white paper, matching the style of the first one.

Difference Score = 0.26209497
Its a Genuine Signature

`predict_score(30)`

Genuine



Genuine



```
Difference Score =  0.7071669  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



```
Difference Score =  0.87343353  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



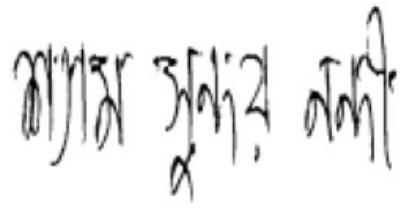
Forged



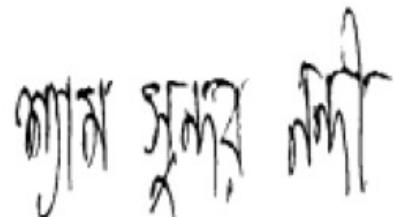
```
Difference Score =  0.8028871  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



Genuine



```
Difference Score =  0.35500818  
Its a Genuine Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine

Zooprifer mif

Forged

Zooprifer mif

```
Difference Score = 0.99743545  
Its a Forged Signature
```

```
predict_score(randint(0, len(S1_test)))
```

Genuine

Maz

Forged

Maz

```
Difference Score = 0.85190797  
Its a Forged Signature
```

```
predict_score(randint(0, len(S1_test)))
```

Genuine



Forged



```
Difference Score = 0.7647957  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



```
Difference Score = 0.77674943  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine

Forged

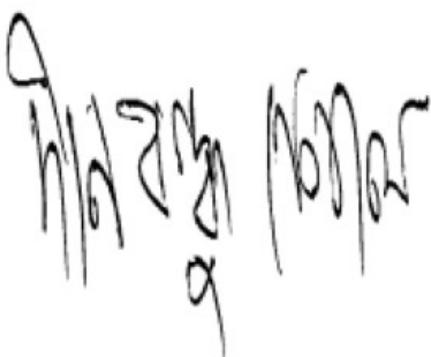


```
Difference Score =  0.91892135  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine

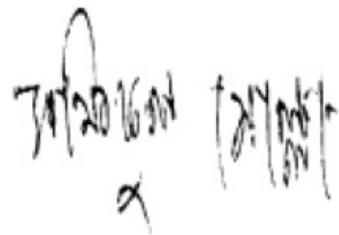
Genuine



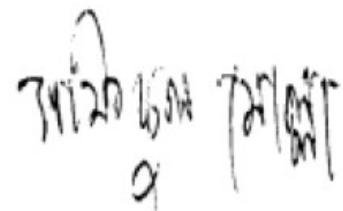
```
Difference Score =  0.2419497  
Its a Genuine Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



Genuine



```
Difference Score =  0.5604566
Its a Genuine Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



```
Difference Score =  0.91316396
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine

31/12/2016 (25/10)

Forged

31/12/2016 25/10

```
Difference Score =  0.26262912  
Its a Genuine Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine

31/12/2016

Forged

31/12/2016

```
Difference Score =  0.96977454  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



```
Difference Score =  0.9770989  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



```
Difference Score =  0.8736247  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



```
Difference Score =  0.9490729  
Its a Forged Signature
```

```
predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



```
Difference Score =  0.76796  
Its a Forged Signature
```

Model6_EfficientNetB0Siamese_BhSig260HindiTraining

November 14, 2024

#Model 6 (EfficientNet + Siamese)

```
[ ]: from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
import os
from tensorflow.keras.initializers import RandomNormal, Constant
import tensorflow as tf
from tensorflow.keras.backend import max, mean, sqrt, square, sum

import seaborn as sns
from tensorflow.keras.optimizers import Adam,RMSprop
from keras.models import Model

from tensorflow import keras
from keras.layers import LeakyReLU, Softmax
from keras.layers import Conv2D, Activation, Input,Dropout,Lambda,Flatten, Dense
from keras.layers import Dense, Flatten, Reshape, Activation
from keras.layers import BatchNormalization ,ZeroPadding2D

from keras.layers import MaxPooling2D
from keras.layers import Concatenate
from keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
from keras.layers import Layer, InputSpec
from keras.regularizers import l2
from keras import backend as K
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from tensorflow import one_hot, reshape

from tensorflow.keras.layers import Activation, Add, AvgPool2D, ↴
    ↴BatchNormalization, Conv2D, Dense, Flatten, Input, MaxPool2D, ZeroPadding2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.initializers import random_uniform, glorot_uniform, ↴
    ↴constant
from tensorflow.keras.utils import plot_model
```

```

from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

from h5py import File
import matplotlib.pyplot as plt
import numpy as np
from numpy.random import permutation, randint, rand
from numpy import rint
import os
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf

from numpy import expand_dims
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow import one_hot, reshape
from keras.layers import BatchNormalization
import cv2

# from keras.models import Sequential
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras import Sequential
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model

```

```

[ ]: import tensorflow as tf
print("Tensorflow version " + tf.__version__)

try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver() # TPU detection
    print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])
except ValueError:
    raise BaseException('ERROR: Not connected to a TPU runtime; please see the previous cell in this notebook for instructions!')

tf.config.experimental_connect_to_cluster(tpu)
tf.tpu.experimental.initialize_tpu_system(tpu)
tpu_strategy = tf.distribute.TPUStrategy(tpu)

```

Tensorflow version 2.12.0
 Running on TPU ['10.9.10.74:8470']

```
[ ]: # import tensorflow as tf
# device_name = tf.test.gpu_device_name()
```

```
# if device_name != '/device:GPU:0':  
#     raise SystemError('GPU device not found')  
# print('Found GPU at: {}'.format(device_name))
```

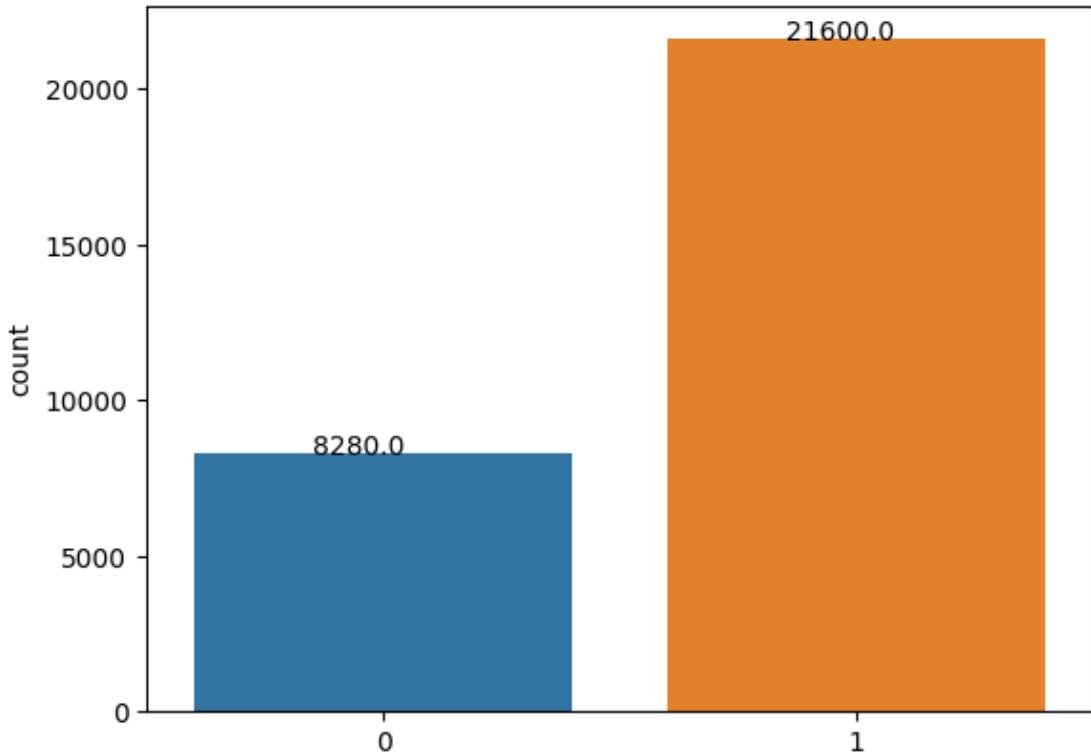
```
[ ]: from google.colab import drive  
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[ ]: # with tpu_strategy.scope():  
colab = True  
# database = 'bigsig260_224x224_siamese_preprocessed.h5'  
if colab:  
    # from google.colab import drive  
    # drive.mount('/content/gdrive')  
    file = '/content/drive/MyDrive/bigsig260224x224x1_siamese_preprocessed.h5'  
print(file)  
with File(file, 'r') as hdf:  
    S1 = np.array(hdf.get('S1'))  
    S2 = np.array(hdf.get('S2'))  
    Y = np.array(hdf.get('Y'))  
print(S1.shape)  
print(S2.shape)  
print(Y.shape)
```

```
/content/drive/MyDrive/bigsig260224x224x1_siamese_preprocessed.h5  
(29880, 224, 224, 1)  
(29880, 224, 224, 1)  
(29880, 1)
```

```
[ ]: ax = sns.countplot(x = Y.reshape(Y.shape[0]))  
for p in ax.patches:  
    ax.annotate('{:}{}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.  
        ↪01))
```



```
[ ]: 21600/(21600 + 8280)
```

```
[ ]: 0.7228915662650602
```

```
[ ]: import sklearn
class_weights = sklearn.utils.class_weight.compute_class_weight(class_weight = "balanced", classes = np.unique(Y), y = Y.reshape(Y.shape[0]))
class_weight_dict = dict(enumerate(class_weights))
class_weight_dict
```

```
[ ]: {0: 1.8043478260869565, 1: 0.6916666666666667}
```

```
[ ]: Y = Y/1.0
```

```
[ ]: S1.dtype
```

```
[ ]: dtype('uint8')
```

```
[ ]: seed=randint(10)
print('seed='+str(seed))
indices = permutation(Y.shape[0])
m = int(0.70 * Y.shape[0])
```

```

n = int(0.15 * Y.shape[0])
training_id, validation_id, test_id = indices[:m], indices[m: m + n], indices[m+n:]
S1_train, S1_test, S1_validate = S1[training_id], S1[test_id], S1[validation_id]
S2_train, S2_test, S2_validate = S2[training_id], S2[test_id], S2[validation_id]
Y_train, Y_test, Y_validate = Y[training_id], Y[test_id], Y[validation_id]
print(S1_train.shape)
print(S2_train.shape)
print(Y_train.shape)

del S1,S2,Y

```

```

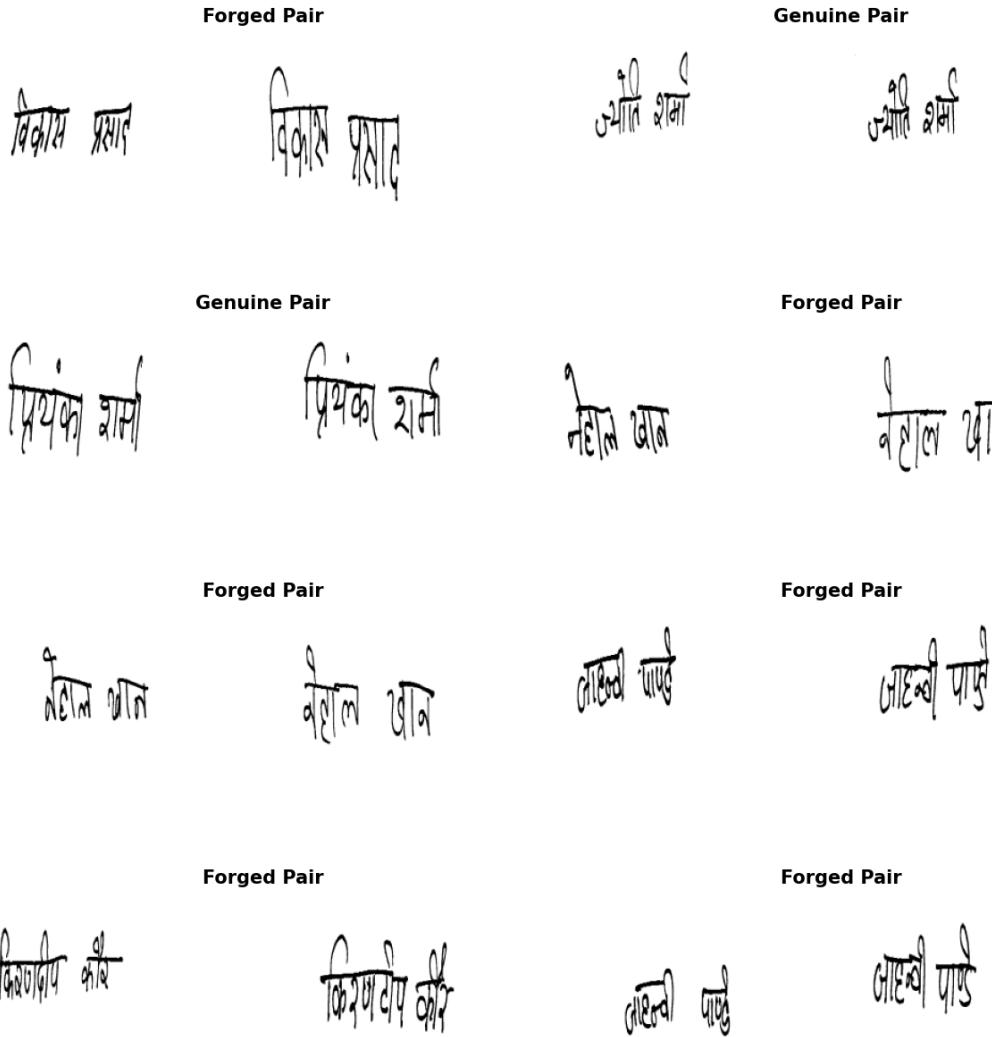
seed=8
(20916, 224, 224, 1)
(20916, 224, 224, 1)
(20916, 1)

```

```

[ ]: fig = plt.figure(figsize = (16,16))
rows,cols = 4,4
i = 1
while(i in range(1,rows*cols)):
    random_idx = randint(0,len(S1_train))
    if(Y_train[random_idx] == 0):
        Label = "Genuine Pair"
    else:
        Label = "Forged Pair"
    img1 = S1_train[random_idx]
    img2 = S2_train[random_idx]
    fig.add_subplot(rows,cols,i)
    plt.imshow(img1.squeeze(),cmap = "gray");
    plt.axis(False);
    i += 1
    fig.add_subplot(rows,cols,i)
    plt.imshow(img2.squeeze(),cmap = "gray");
    plt.axis(False);
    plt.text(0.5, 0.5, Label,
    horizontalalignment='center',verticalalignment='center',fontsize=15,fontweight=
    1000);
    i += 1

```



```
[ ]: Y_train[0]
```

```
[ ]: array([1.])
```

```
[ ]: # #One hot Encoding
# Y_train = one_hot(Y_train, depth=2)
# Y_train = reshape(Y_train, (-1, 2))

# Y_test = one_hot(Y_test, depth=2)
# Y_test = reshape(Y_test, (-1, 2))

# Y_validate = one_hot(Y_validate, depth=2)
# Y_validate = reshape(Y_validate, (-1, 2))
```

```
[ ]: Y_train.shape
[ ]: (20916, 1)
[ ]: input_shape=(224,224,1)
      input_shape
[ ]: (224, 224, 1)

[ ]: def euclidean_distance(vects):
    """Find the Euclidean distance between two vectors.

    Arguments:
        vects: List containing two tensors of same length.

    Returns:
        Tensor containing euclidean distance
        (as floating point value) between vectors.
    """
    x, y = vects
    sum_square = tf.math.reduce_sum(tf.math.square(x - y), axis=1, keepdims=True)
    return tf.math.sqrt(tf.math.maximum(sum_square, tf.keras.backend.epsilon()))

[ ]: def constructive_loss(y_true, y_pred):
    '''Contrastive loss from Hadsell-et-al.'06
    http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf
    '''
    margin = 1
    square_pred = tf.math.square(y_pred)
    margin_square = tf.math.square(tf.math.maximum(margin - (y_pred), 0))
    return tf.math.reduce_mean(
        (1 - y_true) * square_pred + (y_true) * margin_square
    )

[ ]: from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras import layers
NUM_CLASSES = 2
IMG_SIZE = 224
def build_model(num_classes):
    inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
    #x = img_augmentation(inputs)
    x = inputs
    model = EfficientNetB0(include_top=False, input_tensor=x, weights="imagenet")
```

```

# Freeze the pretrained weights
model.trainable = False

# Rebuild top
x = layers.GlobalAveragePooling2D(name="avg_pool")(model.output)
x = layers.BatchNormalization()(x)

top_dropout_rate = 0.2
x = layers.Dropout(top_dropout_rate, name="top_dropout")(x)
outputs = layers.Dense(NUM_CLASSES, activation="relu", name="pred")(x)

model = tf.keras.Model(inputs, outputs, name="EfficientNet")
return model

# inputs = layers.Input(shape = input_shape)
# x = inputs
# model = EfficientNetB0(include_top=False, input_tensor=x, weights = None)

# # Freeze the pretrained weights
# model.trainable = False

# # Rebuild top
# x = layers.GlobalAveragePooling2D(name="avg_pool")(model.output)
# x = layers.BatchNormalization()(x)

# top_dropout_rate = 0.5
# x = layers.Dropout(top_dropout_rate, name="top_dropout")(x)
# outputs = layers.Dense(num_classes, activation="sigmoid", name="pred")(x)

# model = tf.keras.Model(inputs, outputs, name="EfficientNet")
# return model

```

```
[ ]: plot_model(build_model(2))
```

```
[ ]: with tpu_strategy.scope():
# with tf.device('/device:GPU:0'):
    inputShape = S1_train.shape[1:]
    f = build_model(num_classes = 2)
```

Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5
16705208/16705208 [=====] - 0s 0us/step

```
[ ]: def eucl_dist_output_shape(shapes):
    shape1, shape2 = shapes
    return (shape1[0], 1)
```

```
[ ]: with tpu_strategy.scope():
    # network definition
    input_a = Input(shape=(input_shape))
    input_b = Input(shape=(input_shape))

    # because we re-use the same instance `base_network`,
    # the weights of the network
    # will be shared across the two branches
    processed_a = f(input_a)
    processed_b = f(input_b)

    # Compute the Euclidean distance between the two vectors in the latent space
    # https://keras.io/examples/vision/siamese_contrastive/
    merge_layer = layers.Lambda(euclidean_distance)([processed_a, processed_b])
    normal_layer = tf.keras.layers.BatchNormalization()(merge_layer)
    output_layer = layers.Dense(1, activation="sigmoid")(normal_layer)
    model = keras.Model(inputs=[input_a, input_b], outputs=output_layer)
```

```
[ ]: pred = model([expand_dims(S1_train[0],axis = 0), expand_dims(S2_train[0],axis = 0)])
print(pred.shape)
pred
```

(1, 1)

```
[ ]: <tf.Tensor: shape=(1, 1), dtype=float32, numpy=array([[0.5238475]], dtype=float32)>
```

```
[ ]: folder = '/content/drive/MyDrive/EfficientNetHindi224x224x1Part2'
```

0.1 FINAL TRAINING

BEST LEARNING RATE IS 1e-04

```
[ ]: Batch_size = 32
lr = 1e-04
Epochs = 60
```

```
[ ]: with tpu_strategy.scope():
    adam = Adam(learning_rate = lr,epsilon = 1e-08)
    model.compile(loss = constructive_loss, optimizer = adam,metrics=['BinaryAccuracy'])
```

```
[ ]: tf.config.experimental_connect_to_cluster(tf.distribute.cluster_resolver.TPUClusterResolver())
```

```
[ ]: # tf.tpu.experimental.initialize_tpu_system(tf.distribute.cluster_resolver.TPUClusterResolver().master())
```

```
-----
AssertionError                                     Traceback (most recent call last)
<ipython-input-33-91e980e391ce> in <cell line: 1>()
----> 1   tf.tpu.experimental.initialize_tpu_system(tf.distribute.cluster_resolver.TPUClusterResolver().master())
         ↳TPUClusterResolver().master()

/usr/local/lib/python3.10/dist-packages/tensorflow/python/tpu/tpu_strategy_util.py in initialize_tpu_system(cluster_resolver)
    78
    79     cluster_resolver = TPUClusterResolver("")
---> 80     assert isinstance(cluster_resolver, TPUClusterResolver)
    81
    82     tpu_name = compat.as_text(cluster_resolver._tpu)  # pylint: disable=protected-access

AssertionError:
```

```
[ ]: callbacks = [
    EarlyStopping(patience = 6, verbose = 1),
    ReduceLROnPlateau(factor=0.1, patience = 3, min_lr=0.000001, verbose=1),
    ModelCheckpoint(folder + '/Weights/EffiNet-bhsig260-{epoch:03d}.h5', verbose=1, save_weights_only=True)
]
```

```
[ ]: with tpu_strategy.scope():
    results = model.fit(x = [S1_train, S2_train],
                         y = Y_train,
                         validation_data = ([S1_validate,S2_validate],Y_validate),
                         epochs = Epochs,
                         callbacks = callbacks,
                         batch_size = Batch_size,
                         class_weight = class_weight_dict,
                         steps_per_epoch = S1_train.shape[0]//Batch_size-1,
                         validation_steps = S1_validate.shape[0]//Batch_size-1
                         # To fix Loss Nan on TPU
                         # https://stackoverflow.com/questions/64079759/
                         ↳validation-loss-become-nan-while-training-on-tpu-but-perfectly-ok-on-gpu
                         )
```

```
Epoch 1/60
651/652 [=====]>.] - ETA: 0s - loss: 0.3061 -
binary_accuracy: 0.5325
Epoch 1: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-001.h5
652/652 [=====] - 88s 70ms/step - loss: 0.3061 -
```

```
binary_accuracy: 0.5324 - val_loss: 0.2428 - val_binary_accuracy: 0.6120 - lr: 1.0000e-04
Epoch 2/60
651/652 [=====>.] - ETA: 0s - loss: 0.2542 -
binary_accuracy: 0.5911
Epoch 2: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-002.h5
652/652 [=====] - 38s 59ms/step - loss: 0.2542 -
binary_accuracy: 0.5911 - val_loss: 0.1793 - val_binary_accuracy: 0.7014 - lr: 1.0000e-04
Epoch 3/60
651/652 [=====>.] - ETA: 0s - loss: 0.2339 -
binary_accuracy: 0.6142
Epoch 3: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-003.h5
652/652 [=====] - 26s 40ms/step - loss: 0.2339 -
binary_accuracy: 0.6143 - val_loss: 0.1489 - val_binary_accuracy: 0.7774 - lr: 1.0000e-04
Epoch 4/60
652/652 [=====] - ETA: 0s - loss: 0.2147 -
binary_accuracy: 0.6481
Epoch 4: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-004.h5
652/652 [=====] - 26s 40ms/step - loss: 0.2147 -
binary_accuracy: 0.6481 - val_loss: 0.1322 - val_binary_accuracy: 0.8103 - lr: 1.0000e-04
Epoch 5/60
652/652 [=====] - ETA: 0s - loss: 0.2064 -
binary_accuracy: 0.6687
Epoch 5: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-005.h5
652/652 [=====] - 26s 40ms/step - loss: 0.2064 -
binary_accuracy: 0.6687 - val_loss: 0.1235 - val_binary_accuracy: 0.8330 - lr: 1.0000e-04
Epoch 6/60
651/652 [=====>.] - ETA: 0s - loss: 0.1987 -
binary_accuracy: 0.6894
Epoch 6: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-006.h5
652/652 [=====] - 25s 39ms/step - loss: 0.1988 -
binary_accuracy: 0.6892 - val_loss: 0.1206 - val_binary_accuracy: 0.8491 - lr: 1.0000e-04
Epoch 7/60
```

```
652/652 [=====] - ETA: 0s - loss: 0.1927 -
binary_accuracy: 0.7095
Epoch 7: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-007.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1927 -
binary_accuracy: 0.7095 - val_loss: 0.1172 - val_binary_accuracy: 0.8588 - lr:
1.0000e-04
Epoch 8/60
652/652 [=====] - ETA: 0s - loss: 0.1903 -
binary_accuracy: 0.7179
Epoch 8: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-008.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1903 -
binary_accuracy: 0.7179 - val_loss: 0.1132 - val_binary_accuracy: 0.8656 - lr:
1.0000e-04
Epoch 9/60
652/652 [=====] - ETA: 0s - loss: 0.1860 -
binary_accuracy: 0.7295
Epoch 9: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-009.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1860 -
binary_accuracy: 0.7295 - val_loss: 0.1127 - val_binary_accuracy: 0.8768 - lr:
1.0000e-04
Epoch 10/60
651/652 [=====>.] - ETA: 0s - loss: 0.1861 -
binary_accuracy: 0.7297
Epoch 10: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-010.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1860 -
binary_accuracy: 0.7300 - val_loss: 0.1061 - val_binary_accuracy: 0.8847 - lr:
1.0000e-04
Epoch 11/60
651/652 [=====>.] - ETA: 0s - loss: 0.1861 -
binary_accuracy: 0.7301
Epoch 11: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-011.h5
652/652 [=====] - 26s 41ms/step - loss: 0.1861 -
binary_accuracy: 0.7302 - val_loss: 0.1062 - val_binary_accuracy: 0.8822 - lr:
1.0000e-04
Epoch 12/60
652/652 [=====] - ETA: 0s - loss: 0.1848 -
binary_accuracy: 0.7277
Epoch 12: saving model to
```

```
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-012.h5
652/652 [=====] - 30s 46ms/step - loss: 0.1848 -
binary_accuracy: 0.7277 - val_loss: 0.1058 - val_binary_accuracy: 0.8883 - lr:
1.0000e-04
Epoch 13/60
651/652 [=====>.] - ETA: 0s - loss: 0.1832 -
binary_accuracy: 0.7283
Epoch 13: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-013.h5
652/652 [=====] - 25s 39ms/step - loss: 0.1833 -
binary_accuracy: 0.7282 - val_loss: 0.1031 - val_binary_accuracy: 0.8923 - lr:
1.0000e-04
Epoch 14/60
651/652 [=====>.] - ETA: 0s - loss: 0.1816 -
binary_accuracy: 0.7309
Epoch 14: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-014.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1817 -
binary_accuracy: 0.7308 - val_loss: 0.1002 - val_binary_accuracy: 0.8941 - lr:
1.0000e-04
Epoch 15/60
652/652 [=====] - ETA: 0s - loss: 0.1818 -
binary_accuracy: 0.7374
Epoch 15: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-015.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1818 -
binary_accuracy: 0.7374 - val_loss: 0.0987 - val_binary_accuracy: 0.8948 - lr:
1.0000e-04
Epoch 16/60
652/652 [=====] - ETA: 0s - loss: 0.1800 -
binary_accuracy: 0.7389
Epoch 16: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-016.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1800 -
binary_accuracy: 0.7389 - val_loss: 0.0963 - val_binary_accuracy: 0.9000 - lr:
1.0000e-04
Epoch 17/60
651/652 [=====>.] - ETA: 0s - loss: 0.1776 -
binary_accuracy: 0.7427
Epoch 17: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-017.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1776 -
```

```
binary_accuracy: 0.7429 - val_loss: 0.0953 - val_binary_accuracy: 0.9011 - lr: 1.0000e-04
Epoch 18/60
652/652 [=====] - ETA: 0s - loss: 0.1784 -
binary_accuracy: 0.7372
Epoch 18: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-018.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1784 -
binary_accuracy: 0.7372 - val_loss: 0.0931 - val_binary_accuracy: 0.9047 - lr: 1.0000e-04
Epoch 19/60
651/652 [=====>.] - ETA: 0s - loss: 0.1745 -
binary_accuracy: 0.7454
Epoch 19: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-019.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1745 -
binary_accuracy: 0.7454 - val_loss: 0.0906 - val_binary_accuracy: 0.9045 - lr: 1.0000e-04
Epoch 20/60
651/652 [=====>.] - ETA: 0s - loss: 0.1750 -
binary_accuracy: 0.7451
Epoch 20: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-020.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1750 -
binary_accuracy: 0.7451 - val_loss: 0.0906 - val_binary_accuracy: 0.9049 - lr: 1.0000e-04
Epoch 21/60
651/652 [=====>.] - ETA: 0s - loss: 0.1731 -
binary_accuracy: 0.7495
Epoch 21: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-021.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1731 -
binary_accuracy: 0.7495 - val_loss: 0.0893 - val_binary_accuracy: 0.9022 - lr: 1.0000e-04
Epoch 22/60
652/652 [=====] - ETA: 0s - loss: 0.1726 -
binary_accuracy: 0.7455
Epoch 22: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-022.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1726 -
binary_accuracy: 0.7455 - val_loss: 0.0846 - val_binary_accuracy: 0.9074 - lr: 1.0000e-04
Epoch 23/60
```

```
652/652 [=====] - ETA: 0s - loss: 0.1727 -
binary_accuracy: 0.7491
Epoch 23: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-023.h5
652/652 [=====] - 25s 38ms/step - loss: 0.1727 -
binary_accuracy: 0.7491 - val_loss: 0.0856 - val_binary_accuracy: 0.9103 - lr:
1.0000e-04
Epoch 24/60
651/652 [=====>.] - ETA: 0s - loss: 0.1724 -
binary_accuracy: 0.7458
Epoch 24: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-024.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1724 -
binary_accuracy: 0.7456 - val_loss: 0.0854 - val_binary_accuracy: 0.9103 - lr:
1.0000e-04
Epoch 25/60
651/652 [=====>.] - ETA: 0s - loss: 0.1724 -
binary_accuracy: 0.7449
Epoch 25: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-025.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1725 -
binary_accuracy: 0.7448 - val_loss: 0.0829 - val_binary_accuracy: 0.9134 - lr:
1.0000e-04
Epoch 26/60
651/652 [=====>.] - ETA: 0s - loss: 0.1699 -
binary_accuracy: 0.7520
Epoch 26: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-026.h5
652/652 [=====] - 25s 39ms/step - loss: 0.1699 -
binary_accuracy: 0.7522 - val_loss: 0.0816 - val_binary_accuracy: 0.9110 - lr:
1.0000e-04
Epoch 27/60
652/652 [=====] - ETA: 0s - loss: 0.1687 -
binary_accuracy: 0.7527
Epoch 27: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-027.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1687 -
binary_accuracy: 0.7527 - val_loss: 0.0810 - val_binary_accuracy: 0.9132 - lr:
1.0000e-04
Epoch 28/60
651/652 [=====>.] - ETA: 0s - loss: 0.1672 -
binary_accuracy: 0.7541
Epoch 28: saving model to
```

```
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-028.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1671 -
binary_accuracy: 0.7543 - val_loss: 0.0786 - val_binary_accuracy: 0.9123 - lr:
1.0000e-04
Epoch 29/60
652/652 [=====] - ETA: 0s - loss: 0.1682 -
binary_accuracy: 0.7528
Epoch 29: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-029.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1682 -
binary_accuracy: 0.7528 - val_loss: 0.0783 - val_binary_accuracy: 0.9195 - lr:
1.0000e-04
Epoch 30/60
651/652 [=====>.] - ETA: 0s - loss: 0.1670 -
binary_accuracy: 0.7551
Epoch 30: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-030.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1670 -
binary_accuracy: 0.7552 - val_loss: 0.0766 - val_binary_accuracy: 0.9195 - lr:
1.0000e-04
Epoch 31/60
652/652 [=====] - ETA: 0s - loss: 0.1675 -
binary_accuracy: 0.7531
Epoch 31: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-031.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1675 -
binary_accuracy: 0.7531 - val_loss: 0.0770 - val_binary_accuracy: 0.9164 - lr:
1.0000e-04
Epoch 32/60
652/652 [=====] - ETA: 0s - loss: 0.1664 -
binary_accuracy: 0.7566
Epoch 32: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-032.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1664 -
binary_accuracy: 0.7566 - val_loss: 0.0767 - val_binary_accuracy: 0.9168 - lr:
1.0000e-04
Epoch 33/60
651/652 [=====>.] - ETA: 0s - loss: 0.1675 -
binary_accuracy: 0.7514
Epoch 33: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-033.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1675 -
```

```
binary_accuracy: 0.7513 - val_loss: 0.0751 - val_binary_accuracy: 0.9240 - lr: 1.0000e-04
Epoch 34/60
651/652 [=====>.] - ETA: 0s - loss: 0.1638 -
binary_accuracy: 0.7550
Epoch 34: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-034.h5
652/652 [=====] - 27s 41ms/step - loss: 0.1638 -
binary_accuracy: 0.7550 - val_loss: 0.0747 - val_binary_accuracy: 0.9197 - lr: 1.0000e-04
Epoch 35/60
651/652 [=====>.] - ETA: 0s - loss: 0.1644 -
binary_accuracy: 0.7592
Epoch 35: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-035.h5
652/652 [=====] - 29s 45ms/step - loss: 0.1645 -
binary_accuracy: 0.7590 - val_loss: 0.0741 - val_binary_accuracy: 0.9186 - lr: 1.0000e-04
Epoch 36/60
651/652 [=====>.] - ETA: 0s - loss: 0.1643 -
binary_accuracy: 0.7585
Epoch 36: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-036.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1644 -
binary_accuracy: 0.7585 - val_loss: 0.0747 - val_binary_accuracy: 0.9213 - lr: 1.0000e-04
Epoch 37/60
651/652 [=====>.] - ETA: 0s - loss: 0.1651 -
binary_accuracy: 0.7552
Epoch 37: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-037.h5
652/652 [=====] - 25s 39ms/step - loss: 0.1653 -
binary_accuracy: 0.7549 - val_loss: 0.0742 - val_binary_accuracy: 0.9263 - lr: 1.0000e-04
Epoch 38/60
651/652 [=====>.] - ETA: 0s - loss: 0.1657 -
binary_accuracy: 0.7577
Epoch 38: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-bhsig260-038.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1656 -
binary_accuracy: 0.7577 - val_loss: 0.0708 - val_binary_accuracy: 0.9256 - lr: 1.0000e-04
Epoch 39/60
```

```
652/652 [=====] - ETA: 0s - loss: 0.1634 -
binary_accuracy: 0.7556
Epoch 39: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-039.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1634 -
binary_accuracy: 0.7556 - val_loss: 0.0725 - val_binary_accuracy: 0.9254 - lr:
1.0000e-04
Epoch 40/60
651/652 [=====>.] - ETA: 0s - loss: 0.1640 -
binary_accuracy: 0.7571
Epoch 40: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-040.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1640 -
binary_accuracy: 0.7572 - val_loss: 0.0711 - val_binary_accuracy: 0.9229 - lr:
1.0000e-04
Epoch 41/60
651/652 [=====>.] - ETA: 0s - loss: 0.1653 -
binary_accuracy: 0.7560
Epoch 41: ReduceLROnPlateau reducing learning rate to 9.99999747378752e-06.

Epoch 41: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-041.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1654 -
binary_accuracy: 0.7558 - val_loss: 0.0711 - val_binary_accuracy: 0.9260 - lr:
1.0000e-04
Epoch 42/60
652/652 [=====] - ETA: 0s - loss: 0.1622 -
binary_accuracy: 0.7568
Epoch 42: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-042.h5
652/652 [=====] - 26s 41ms/step - loss: 0.1622 -
binary_accuracy: 0.7568 - val_loss: 0.0713 - val_binary_accuracy: 0.9296 - lr:
1.0000e-05
Epoch 43/60
652/652 [=====] - ETA: 0s - loss: 0.1649 -
binary_accuracy: 0.7556
Epoch 43: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-043.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1649 -
binary_accuracy: 0.7556 - val_loss: 0.0702 - val_binary_accuracy: 0.9254 - lr:
1.0000e-05
Epoch 44/60
651/652 [=====>.] - ETA: 0s - loss: 0.1644 -
```

```
binary_accuracy: 0.7566
Epoch 44: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-044.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1643 -
binary_accuracy: 0.7566 - val_loss: 0.0697 - val_binary_accuracy: 0.9269 - lr:
1.0000e-05
Epoch 45/60
652/652 [=====] - ETA: 0s - loss: 0.1630 -
binary_accuracy: 0.7602
Epoch 45: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-045.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1630 -
binary_accuracy: 0.7602 - val_loss: 0.0701 - val_binary_accuracy: 0.9287 - lr:
1.0000e-05
Epoch 46/60
652/652 [=====] - ETA: 0s - loss: 0.1640 -
binary_accuracy: 0.7545
Epoch 46: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-046.h5
652/652 [=====] - 27s 42ms/step - loss: 0.1640 -
binary_accuracy: 0.7545 - val_loss: 0.0695 - val_binary_accuracy: 0.9276 - lr:
1.0000e-05
Epoch 47/60
652/652 [=====] - ETA: 0s - loss: 0.1657 -
binary_accuracy: 0.7540
Epoch 47: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-047.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1657 -
binary_accuracy: 0.7540 - val_loss: 0.0697 - val_binary_accuracy: 0.9272 - lr:
1.0000e-05
Epoch 48/60
651/652 [=====>.] - ETA: 0s - loss: 0.1643 -
binary_accuracy: 0.7536
Epoch 48: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-048.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1643 -
binary_accuracy: 0.7536 - val_loss: 0.0710 - val_binary_accuracy: 0.9283 - lr:
1.0000e-05
Epoch 49/60
651/652 [=====>.] - ETA: 0s - loss: 0.1625 -
binary_accuracy: 0.7605
Epoch 49: ReduceLROnPlateau reducing learning rate to 1e-06.
```

```

Epoch 49: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-049.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1625 -
binary_accuracy: 0.7605 - val_loss: 0.0708 - val_binary_accuracy: 0.9254 - lr:
1.0000e-05
Epoch 50/60
651/652 [=====>.] - ETA: 0s - loss: 0.1623 -
binary_accuracy: 0.7578
Epoch 50: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-050.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1623 -
binary_accuracy: 0.7578 - val_loss: 0.0711 - val_binary_accuracy: 0.9263 - lr:
1.0000e-06
Epoch 51/60
652/652 [=====] - ETA: 0s - loss: 0.1632 -
binary_accuracy: 0.7591
Epoch 51: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-051.h5
652/652 [=====] - 26s 39ms/step - loss: 0.1632 -
binary_accuracy: 0.7591 - val_loss: 0.0703 - val_binary_accuracy: 0.9267 - lr:
1.0000e-06
Epoch 52/60
652/652 [=====] - ETA: 0s - loss: 0.1630 -
binary_accuracy: 0.7602
Epoch 52: saving model to
/content/drive/MyDrive/EfficientNetHindi224x224x1Part2/Weights/EffiNet-
bhsig260-052.h5
652/652 [=====] - 26s 40ms/step - loss: 0.1630 -
binary_accuracy: 0.7602 - val_loss: 0.0700 - val_binary_accuracy: 0.9233 - lr:
1.0000e-06
Epoch 52: early stopping

```

1 Evaluation

```

[ ]: # Y_train.shape
[ ]: import pickle
[ ]: # with open(folder + '/trainHistoryDict.h5', 'wb') as file_pi:
#         pickle.dump(results.history, file_pi)
# # model.save(folder + 'BestModel.h5')
[ ]: history_loaded = pickle.load(open(folder + '/trainHistoryDict.h5', "rb"))
# history_loaded

```

```
[ ]: # history_loaded = results

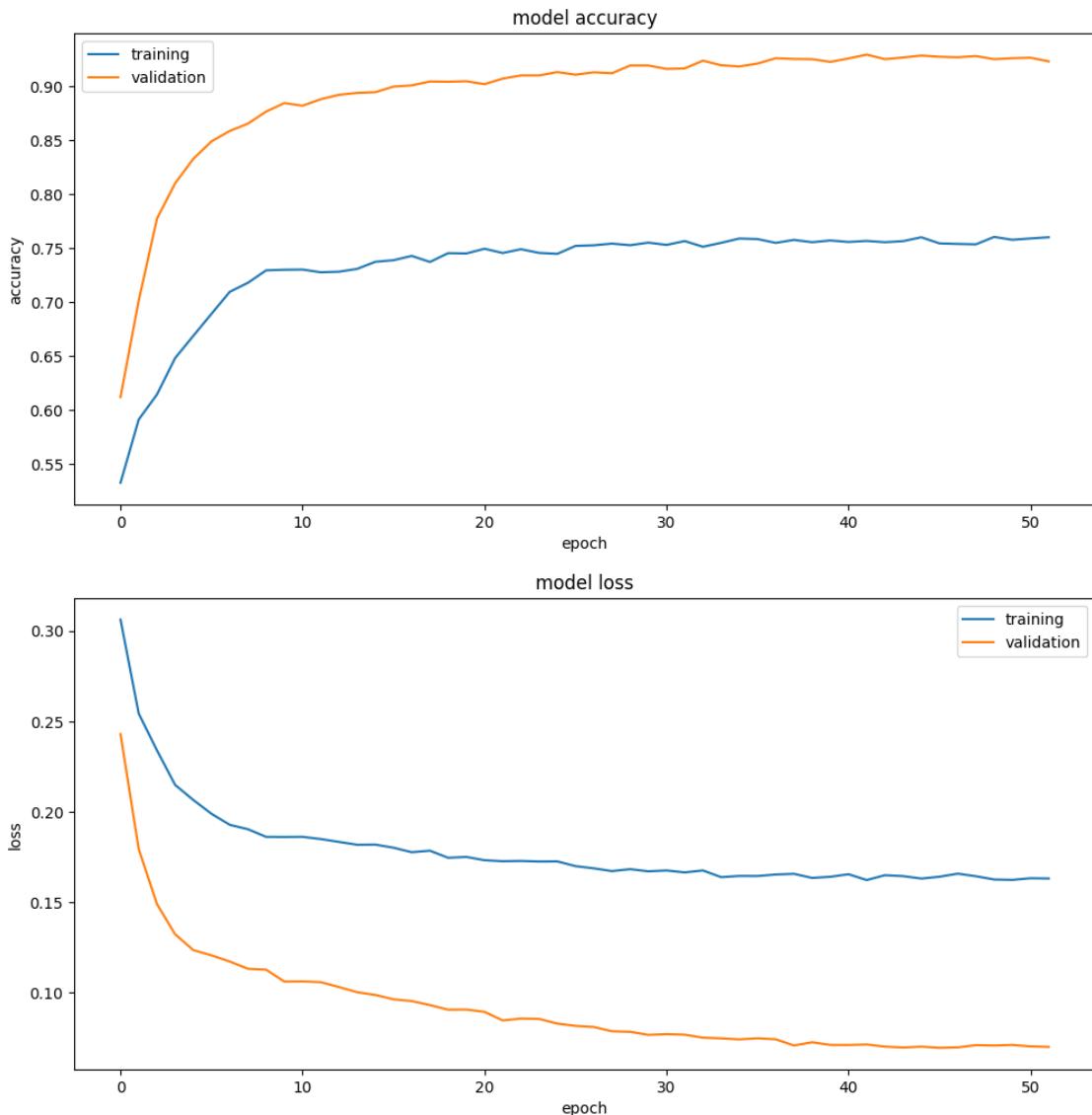
[ ]: best_result = history_loaded['val_loss'].index(min(history_loaded['val_loss'])) + 1
      best_result

[ ]: 46

[ ]: def display_training_curves(training, validation, title, subplot):
    ax = plt.subplot(subplot)
    ax.plot(training)
    ax.plot(validation)
    ax.set_title('model ' + title)
    ax.set_ylabel(title)
    ax.set_xlabel('epoch')
    ax.legend(['training', 'validation'])

    plt.subplots(figsize=(10,10))
    plt.tight_layout()
    display_training_curves(history_loaded['binary_accuracy'], history_loaded['val_binary_accuracy'], 'accuracy', 211)
    display_training_curves(history_loaded['loss'], history_loaded['val_loss'], 'loss', 212)
    plt.savefig(folder + "loss_accuracy.svg", dpi = 1200)

<ipython-input-31-2856f8786416>:2: MatplotlibDeprecationWarning: Auto-removal of
overlapping axes is deprecated since 3.6 and will be removed two minor releases
later; explicitly call ax.remove() as needed.
    ax = plt.subplot(subplot)
```



```
[ ]: if(best_result < 10):
    model.load_weights(folder + f'/Weights/EffiNet-bhsig260-00{best_result}.h5')
else:
    model.load_weights(folder + f'/Weights/EffiNet-bhsig260-0{best_result}.h5')

[ ]: y_pred_keras = model.predict([S1_test, S2_test])
141/141 [=====] - 24s 134ms/step

[ ]: y_pred_keras
```

```
[ ]: array([[0.55407536],
           [0.99932563],
           [0.9224025 ],
           ...,
           [0.32901496],
           [0.9587754 ],
           [0.9988337 ]], dtype=float32)

[ ]: np.min(y_pred_keras),np.mean(y_pred_keras),np.max(y_pred_keras)

[ ]: (0.32901496, 0.7550815, 0.9999997)

[ ]: # # if we keep threshold as fixed to the mean value
# threshold = 0.755

[ ]: def pred_threshold(y_pred_keras,threshold):
    y_pred_keras_new = np.zeros((len(y_pred_keras), 1), dtype = np.uint8)
    for i in range(len(y_pred_keras)):
        label = Y_test[i]
        if y_pred_keras[i] >= threshold:
            predicted_value = 1 # both different
            if(label == predicted_value):
                y_pred_keras_new[i] = label
            else:
                y_pred_keras_new[i] = predicted_value
        else:
            predicted_value = 0
            if(label == predicted_value):
                y_pred_keras_new[i] = label
            else:
                y_pred_keras_new[i] = predicted_value
    return y_pred_keras_new

[ ]: from sklearn.metrics import auc
from sklearn.metrics import roc_curve
threshold_areas = {}
for i in np.arange(0.5,0.8,0.001):
    y_pred_keras_new = pred_threshold(y_pred_keras,i)
    fpr_keras, tpr_keras, thresholds_keras = roc_curve(Y_test, y_pred_keras_new)
    auc_keras = auc(fpr_keras, tpr_keras)
    threshold_areas[i] = auc_keras
    # print(i,auc_keras)
print(threshold_areas)
max_value = list(threshold_areas.values())
max_key= list(threshold_areas.keys())
threshold = max_key[max_value.index(max(max_value))]
y_pred_keras_new = pred_threshold(y_pred_keras,threshold)
```

```

fpr_keras, tpr_keras, thresholds_keras = roc_curve(Y_test, y_pred_keras_new)
auc_keras = auc(fpr_keras, tpr_keras)
print("False Positive Rate : ",fpr_keras[1],"\nTrue Positive Rate :
      ↵",tpr_keras[1])
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='Keras (area = {:.3f},threshold = {:.3f})'.
         format(auc_keras,threshold))
# plt.plot(fpr_rf, tpr_rf, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.savefig(folder + "roc.svg",dpi = 1200)
plt.show()

```

{0.5: 0.9156229912033581, 0.501: 0.9157345258104277, 0.502: 0.9161508455439831, 0.503: 0.9159984529807402, 0.504: 0.9159984529807402, 0.505: 0.9159984529807402, 0.506: 0.9159984529807402, 0.507: 0.9158460604174973, 0.508: 0.9173589467884758, 0.509: 0.9174704813955454, 0.51: 0.9174704813955454, 0.511: 0.9174704813955454, 0.512: 0.9174704813955454, 0.513: 0.9174704813955454, 0.514: 0.9174704813955454, 0.515: 0.9173180888323025, 0.516: 0.9173180888323025, 0.517: 0.9198160072336348, 0.518: 0.9196636146703918, 0.519: 0.9195112221071489, 0.52: 0.9190540444174201, 0.521: 0.9190540444174201, 0.522: 0.9190540444174201, 0.523: 0.9190540444174201, 0.524: 0.9185968667276913, 0.525: 0.9182512236450321, 0.526: 0.9182512236450321, 0.527: 0.9182512236450321, 0.528: 0.9182512236450321, 0.529: 0.9182512236450321, 0.53: 0.9191953977192124, 0.531: 0.9191953977192124, 0.532: 0.9191953977192124, 0.533: 0.9190430051559696, 0.534: 0.9190430051559696, 0.535: 0.9190430051559696, 0.536: 0.9190430051559696, 0.537: 0.9190021471997962, 0.538: 0.9190021471997962, 0.539: 0.9188497546365533, 0.54: 0.9188497546365533, 0.541: 0.9192660743701085, 0.542: 0.919682394103664, 0.543: 0.9205150335707748, 0.544: 0.9213476730378853, 0.545: 0.9219870619855799, 0.546: 0.9214890263396778, 0.547: 0.9219053460732332, 0.548: 0.9217529535099903, 0.549: 0.9216005609467475, 0.55: 0.9214481683835045, 0.551: 0.9208385981305328, 0.552: 0.9211025253008452, 0.553: 0.9211025253008452, 0.554: 0.9211025253008452, 0.555: 0.9223514845015114, 0.556: 0.9223514845015114, 0.557: 0.9227678042350668, 0.558: 0.9227678042350668, 0.559: 0.9227678042350668, 0.56: 0.9227678042350668, 0.561: 0.9227678042350668, 0.562: 0.9231841239686222, 0.5630000000000001: 0.9231841239686222, 0.5640000000000001: 0.9230317314053792, 0.5650000000000001: 0.9230317314053792, 0.5660000000000001: 0.9246970103396008, 0.5670000000000001: 0.9246970103396008, 0.5680000000000001: 0.9245446177763578, 0.5690000000000001: 0.9245446177763578, 0.5700000000000001: 0.9245446177763578, 0.5710000000000001: 0.9243513672569416, 0.5720000000000001: 0.9241989746936985, 0.5730000000000001: 0.9241989746936985, 0.5740000000000001: 0.9241989746936985, 0.5750000000000001: 0.9241989746936985, 0.5760000000000001: 0.9241989746936985, 0.5770000000000001: 0.924310509300768, 0.5780000000000001: 0.924310509300768, 0.5790000000000001: 0.924310509300768, 0.5800000000000001: 0.924310509300768, 0.5810000000000001: 0.9241581167375252, 0.5820000000000001: 0.9241581167375252,

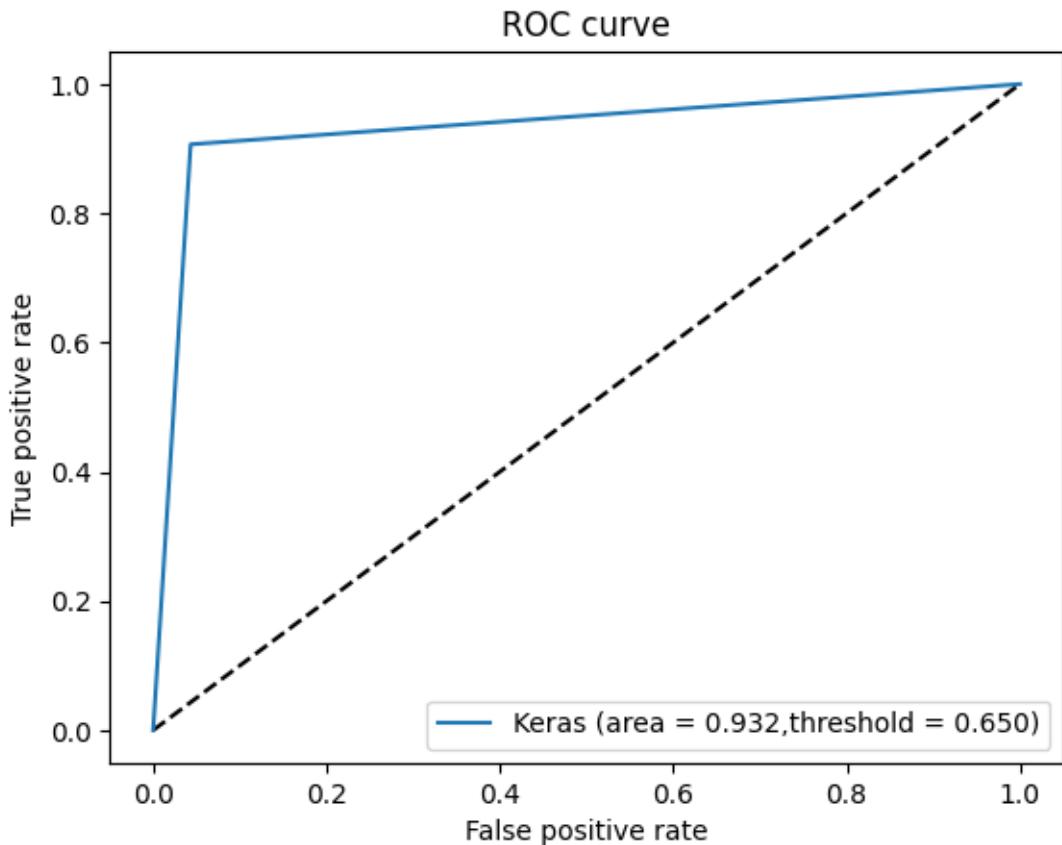
0.5830000000000001: 0.9247975056852196, 0.5840000000000001: 0.9247975056852196,
 0.5850000000000001: 0.9247975056852196, 0.5860000000000001: 0.9247975056852196,
 0.5870000000000001: 0.9258940723226429, 0.5880000000000001: 0.9258940723226429,
 0.5890000000000001: 0.9258940723226429, 0.5900000000000001: 0.9258940723226429,
 0.5910000000000001: 0.9265743192265107, 0.5920000000000001: 0.9265743192265107,
 0.5930000000000001: 0.9265743192265107, 0.5940000000000001: 0.9264219266632678,
 0.5950000000000001: 0.9268382463968231, 0.5960000000000001: 0.9268382463968231,
 0.5970000000000001: 0.9259238910173656, 0.5980000000000001: 0.9257714984541227,
 0.5990000000000001: 0.9257714984541227, 0.6000000000000001: 0.9256191058908798,
 0.6010000000000001: 0.9256191058908798, 0.6020000000000001: 0.9256191058908798,
 0.6030000000000001: 0.9256191058908798, 0.6040000000000001: 0.9256191058908798,
 0.6050000000000001: 0.9256191058908798, 0.6060000000000001: 0.9256191058908798,
 0.6070000000000001: 0.9258830330611924, 0.6080000000000001: 0.9258830330611924,
 0.6090000000000001: 0.9257306404979494, 0.6100000000000001: 0.9257306404979494,
 0.6110000000000001: 0.9254258553714636, 0.6120000000000001: 0.9258421751050189,
 0.6130000000000001: 0.9254965320223598, 0.6140000000000001: 0.9254556740661863,
 0.6150000000000001: 0.9253032815029435, 0.6160000000000001: 0.9253032815029435,
 0.6170000000000001: 0.9249984963764577, 0.6180000000000001: 0.9249984963764577,
 0.6190000000000001: 0.9248461038132147, 0.6200000000000001: 0.9248461038132147,
 0.6210000000000001: 0.9245413186867288, 0.6220000000000001: 0.9245413186867288,
 0.6230000000000001: 0.9245413186867288, 0.6240000000000001: 0.9245413186867288,
 0.6250000000000001: 0.9249576384202842, 0.6260000000000001: 0.9245004607305555,
 0.6270000000000001: 0.9249167804641109, 0.6280000000000001: 0.9249167804641109,
 0.6290000000000001: 0.9249167804641109, 0.6300000000000001: 0.9247235299446946,
 0.6310000000000001: 0.9247235299446946, 0.6320000000000001: 0.9266527360492285,
 0.6330000000000001: 0.9265003434859856, 0.6340000000000001: 0.926764270656298,
 0.6350000000000001: 0.9271805903898535, 0.6360000000000001: 0.9270281978266106,
 0.6370000000000001: 0.9270281978266106, 0.6380000000000001: 0.9270281978266106,
 0.6390000000000001: 0.9270281978266106, 0.6400000000000001: 0.9270281978266106,
 0.6410000000000001: 0.9282771570272766, 0.6420000000000001: 0.9282771570272766,
 0.6430000000000001: 0.928693476760832, 0.6440000000000001: 0.9295261162279427,
 0.6450000000000001: 0.9299424359614981, 0.6460000000000001: 0.9299424359614981,
 0.6470000000000001: 0.9302063631318105, 0.6480000000000001: 0.9302063631318105,
 0.6490000000000001: 0.9313029297692338, 0.6500000000000001: 0.9317192495027892,
 0.6510000000000001: 0.9317192495027892, 0.6520000000000001: 0.9317192495027892,
 0.6530000000000001: 0.9317192495027892, 0.6540000000000001: 0.9315668569395462,
 0.6550000000000001: 0.9308048941233317, 0.6560000000000001: 0.9308048941233317,
 0.6570000000000001: 0.9300429313071171, 0.6580000000000001: 0.9300429313071171,
 0.6590000000000001: 0.9300429313071171, 0.6600000000000001: 0.9300429313071171,
 0.6610000000000001: 0.9300429313071171, 0.6620000000000001: 0.9298905387438742,
 0.6630000000000001: 0.9298905387438742, 0.6640000000000001: 0.9301544659141866,
 0.6650000000000001: 0.9301544659141866, 0.6660000000000001: 0.9301544659141866,
 0.6670000000000001: 0.9301544659141866, 0.6680000000000001: 0.9301544659141866,
 0.6690000000000002: 0.930570785647742, 0.6700000000000002: 0.9299612153947704,
 0.6710000000000002: 0.9298088228315274, 0.6720000000000002: 0.9296564302682845,
 0.6730000000000002: 0.9296564302682845, 0.6740000000000002: 0.9296564302682845,
 0.6750000000000002: 0.9296564302682845, 0.6760000000000002: 0.9295040377050415,
 0.6770000000000002: 0.9295040377050415, 0.6780000000000002: 0.9295040377050415,

0.6790000000000002: 0.9291992525785558, 0.6800000000000002: 0.9290468600153129,
 0.6810000000000002: 0.9287420748888271, 0.6820000000000002: 0.9285896823255843,
 0.6830000000000002: 0.9285896823255843, 0.6840000000000002: 0.9276344689899533,
 0.6850000000000002: 0.9270248987369817, 0.6860000000000002: 0.9270248987369817,
 0.6870000000000002: 0.9278575382040923, 0.6880000000000002: 0.9278575382040923,
 0.6890000000000002: 0.9278575382040923, 0.6900000000000002: 0.9278575382040923,
 0.6910000000000002: 0.9275527530776064, 0.6920000000000002: 0.9274003605143635,
 0.6930000000000002: 0.9274003605143635, 0.6940000000000002: 0.9278166802479189,
 0.6950000000000002: 0.9278166802479189, 0.6960000000000002: 0.9278166802479189,
 0.6970000000000002: 0.9267499323052185, 0.6980000000000002: 0.9267499323052185,
 0.6990000000000002: 0.9267499323052185, 0.7000000000000002: 0.9265975397419758,
 0.7010000000000002: 0.9265975397419758, 0.7020000000000002: 0.9265975397419758,
 0.7030000000000002: 0.9268614669122881, 0.7040000000000002: 0.9262518966593164,
 0.7050000000000002: 0.9262518966593164, 0.7060000000000002: 0.9262518966593164,
 0.7070000000000002: 0.9262518966593164, 0.7080000000000002: 0.9262518966593164,
 0.7090000000000002: 0.9262518966593164, 0.7100000000000002: 0.9256423264063447,
 0.7110000000000002: 0.9256423264063447, 0.7120000000000002: 0.9253375412798591,
 0.7130000000000002: 0.9245347205074711, 0.7140000000000002: 0.9239251502544994,
 0.7150000000000002: 0.9239251502544994, 0.7160000000000002: 0.9239251502544994,
 0.7170000000000002: 0.9239251502544994, 0.7180000000000002: 0.9239251502544994,
 0.7190000000000002: 0.9239251502544994, 0.7200000000000002: 0.923163187438285,
 0.7210000000000002: 0.9235795071718402, 0.7220000000000002: 0.9239958269053956,
 0.7230000000000002: 0.924259754075708, 0.7240000000000002: 0.9241073615124651,
 0.7250000000000002: 0.9241073615124651, 0.7260000000000002: 0.9241073615124651,
 0.7270000000000002: 0.9239549689492221, 0.7280000000000002: 0.9239549689492221,
 0.7290000000000002: 0.9230406135697646, 0.7300000000000002: 0.9230406135697646,
 0.7310000000000002: 0.92345693330332, 0.7320000000000002: 0.9231521481768343,
 0.7330000000000002: 0.9229997556135914, 0.7340000000000002: 0.9229997556135914,
 0.7350000000000002: 0.9229997556135914, 0.7360000000000002: 0.9228473630503484,
 0.7370000000000002: 0.9226949704871055, 0.7380000000000002: 0.9225425779238626,
 0.7390000000000002: 0.9216282225444051, 0.7400000000000002: 0.9204090820384618,
 0.7410000000000002: 0.9204090820384618, 0.7420000000000002: 0.9204090820384618,
 0.7430000000000002: 0.920104296911976, 0.7440000000000002: 0.9202158315190455,
 0.7450000000000002: 0.9196062612660738, 0.7460000000000002: 0.9186919058866164,
 0.7470000000000002: 0.9183871207601306, 0.7480000000000002: 0.9183871207601306,
 0.7490000000000002: 0.9177775505071589, 0.7500000000000002: 0.9171679802541873,
 0.7510000000000002: 0.9175842999877426, 0.7520000000000002: 0.9163651594817993,
 0.7530000000000002: 0.9163651594817993, 0.7540000000000002: 0.9163651594817993,
 0.7550000000000002: 0.9162127669185564, 0.7560000000000002: 0.9164766940888688,
 0.7570000000000002: 0.9168521558662509, 0.7580000000000002: 0.9168521558662509,
 0.7590000000000002: 0.916547370739765, 0.7600000000000002: 0.9169636904733204,
 0.7610000000000002: 0.91589694253062, 0.7620000000000002: 0.91589694253062,
 0.7630000000000002: 0.91589694253062, 0.7640000000000002: 0.9158560845744467,
 0.7650000000000002: 0.9156628340550302, 0.7660000000000002: 0.9154287255794407,
 0.7670000000000002: 0.9154287255794407, 0.7680000000000002: 0.9149715478897119,
 0.7690000000000002: 0.9142095850734974, 0.7700000000000002: 0.9142095850734974,
 0.7710000000000002: 0.9144735122438098, 0.7720000000000002: 0.9144735122438098,
 0.7730000000000002: 0.9144735122438098, 0.7740000000000002: 0.9140163345540812,

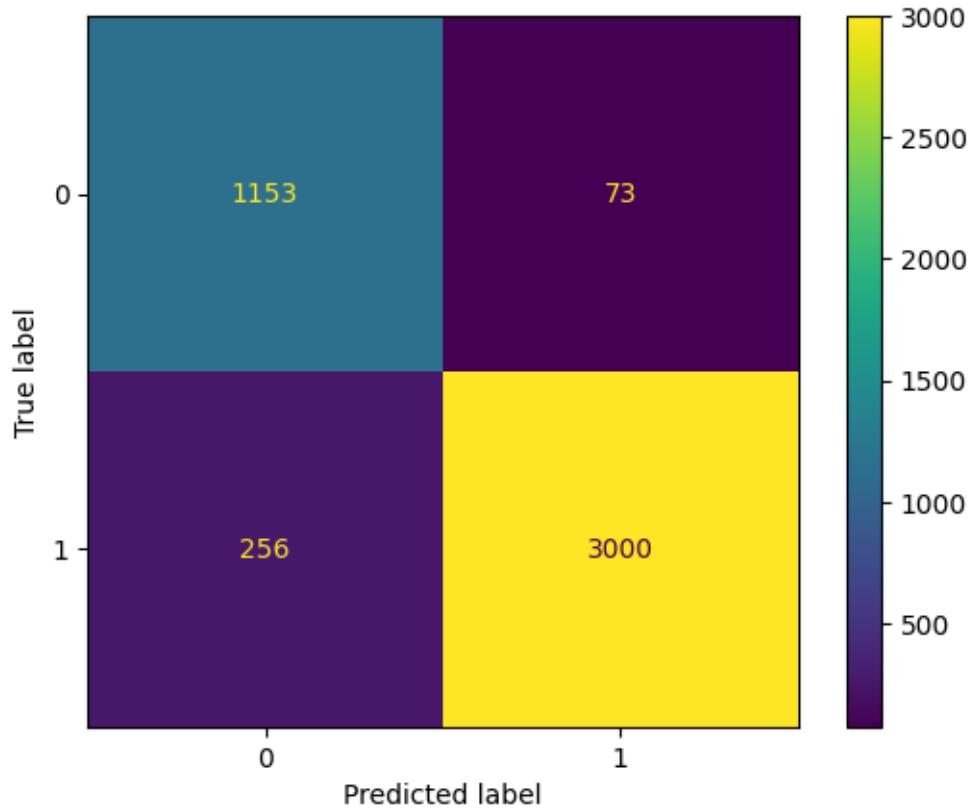
```

0.7750000000000002: 0.9141278691611506, 0.7760000000000002: 0.914391796331463,
0.7770000000000002: 0.9137822260784915, 0.7780000000000002: 0.9136298335152484,
0.7790000000000002: 0.9125630855725481, 0.7800000000000002: 0.9125630855725481,
0.7810000000000002: 0.9122583004460623, 0.7820000000000003: 0.9126746201796176,
0.7830000000000003: 0.9126746201796176, 0.7840000000000003: 0.9126746201796176,
0.7850000000000003: 0.9117602648001601, 0.7860000000000003: 0.9116078722369173,
0.7870000000000003: 0.9113030871104314, 0.7880000000000003: 0.9113030871104314,
0.7890000000000003: 0.9113030871104314, 0.7900000000000003: 0.9113030871104314,
0.7910000000000003: 0.9103887317309739, 0.7920000000000003: 0.9112213711980847,
0.7930000000000003: 0.9110689786348418, 0.7940000000000003: 0.9110689786348418,
0.7950000000000003: 0.9110689786348418, 0.7960000000000003: 0.9106118009451131,
0.7970000000000003: 0.9103070158186273, 0.7980000000000003: 0.9098498381288984,
0.7990000000000003: 0.9098498381288984, 0.8000000000000003: 0.9096974455656555}
False Positive Rate : 0.04329725228975854
True Positive Rate : 0.9067357512953368

```



```
[ ]: ConfusionMatrixDisplay(confusion_matrix(Y_test,y_pred_keras_new)).plot()
plt.savefig(folder + "confusionmatrix.svg",dpi = 1200)
plt.show()
```



Load the weights from the epoch which gave the best validation accuracy

```
[ ]: i = 0
test_gen = ([S1_test[i], S2_test[i]], Y_test[i])
(img1, img2), label = test_gen
result = model([np.expand_dims(a = img1, axis = 0), np.expand_dims(a = img2, axis = 0)])
```

```
[ ]: result.numpy()[0][0]
```

```
[ ]: 0.99851847
```

```
[ ]: def predict_score(i = 0):
    '''Predict distance score and classify test images as Genuine or Forged'''
    test_gen = ([S1_test[i], S2_test[i]], Y_test[i])
    (img1, img2), label = test_gen

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 10))
    ax1.imshow(np.squeeze(img1), cmap='gray')
    ax2.imshow(np.squeeze(img2), cmap='gray')
    ax1.set_title('Genuine')
```

```

if label == 0:
    ax2.set_title('Genuine')
else:
    ax2.set_title('Forged')
ax1.axis('off')
ax2.axis('off')
plt.show()
result = model([np.expand_dims(a = img1, axis = 0), np.expand_dims(a = img2, axis = 0)])
diff = result.numpy()[0][0]
print("Difference Score = ", diff)
if diff >= threshold:
    print("Its a Forged Signature")
else:
    print("Its a Genuine Signature")

```

```

[ ]: def test_accuracy():
    '''Predict distance score and classify test images as Genuine or Forged'''
    predicted_labels = []
    # y_pred_keras_new
    for i in range(len(S1_test)):
        test_gen = ([S1_test[i], S2_test[i]], Y_test[i])
        (img1, img2), label = test_gen
        if(label == y_pred_keras_new[i]):
            predicted_labels.append(True) # append True for correct value else False
        else:
            predicted_labels.append(False)
    correct_prediction = predicted_labels.count(True)
    return (correct_prediction/len(S1_test))*100, predicted_labels

```

```

[ ]: with tpu_strategy.scope():
    test_acc, predicted_labels = test_accuracy()

```

```

[ ]: test_acc

```

```

[ ]: 92.6595269968764

```

```

[ ]: predicted_labels.count(True)/len(predicted_labels)*100

```

```

[ ]: 92.6595269968764

```

```

[ ]: predicted_labels.count(False)

```

```

[ ]: 329

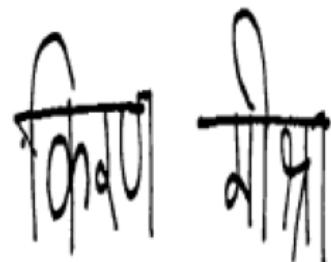
```

1.0.1 Note: The first image is always Genuine. Score prediction and classification is done for the second image

```
[ ]: predict_score(0)
```

Genuine

Forged

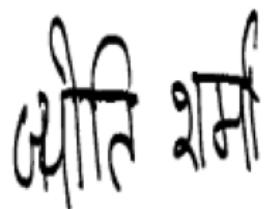
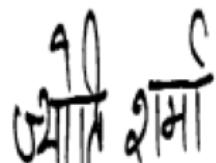


Difference Score = 0.99851847
Its a Forged Signature

```
[ ]: predict_score(1)
```

Genuine

Forged

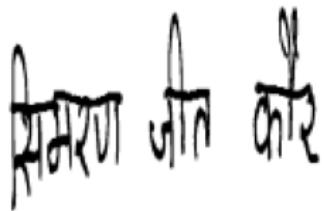


Difference Score = 0.9616341

Its a Forged Signature

```
[ ]: predict_score(2)
```

Genuine



Forged

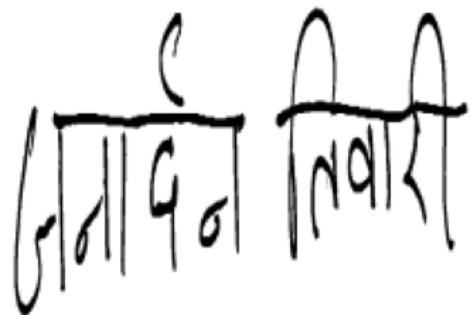


Difference Score = 0.9941176

Its a Forged Signature

```
[ ]: predict_score(10)
```

Genuine



Forged



Difference Score = 0.970948

Its a Forged Signature

```
[ ]: predict_score(11)
```

Genuine

Forged

आकृति अवाल

आकृति अवाल

Difference Score = 0.99911666

Its a Forged Signature

```
[ ]: predict_score(13)
```

Genuine

Forged

सिमरण जीव कीर

सिमरण जीव कीर

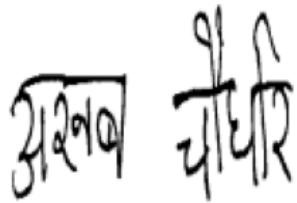
Difference Score = 0.8319302

Its a Forged Signature

```
[ ]: predict_score(8)
```

Genuine

Forged



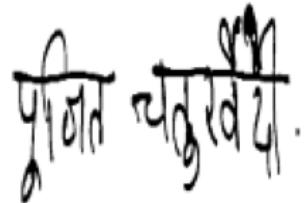
Difference Score = 0.96362895

Its a Forged Signature

```
[ ]: predict_score(19)
```

Genuine

Forged

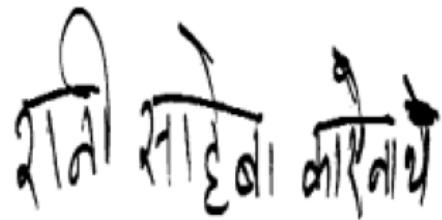


Difference Score = 0.78907496

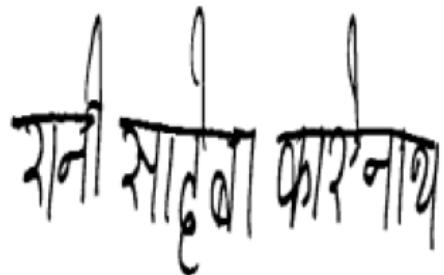
Its a Forged Signature

```
[ ]: predict_score(20)
```

Genuine



Forged

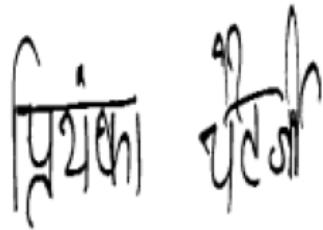


Difference Score = 0.99644685

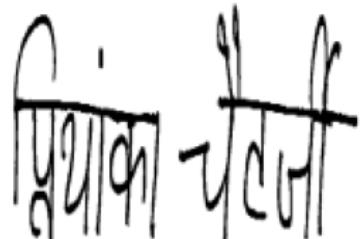
Its a Forged Signature

```
[ ]: predict_score(12)
```

Genuine



Forged



Difference Score = 0.9951913

Its a Forged Signature

```
[ ]: predict_score(15)
```

Genuine

A handwritten signature in Devanagari script, likely Marathi, reading "राजकीया" (Rajkiya).

Forged

A forged handwritten signature in Devanagari script, likely Marathi, reading "राजकीया" (Rajkiya).

Difference Score = 0.9999387

Its a Forged Signature

```
[ ]: predict_score(16)
```

Genuine

A handwritten signature in Devanagari script, likely Marathi, reading "प्रतिष्ठा" (Pratista).

Forged

A forged handwritten signature in Devanagari script, likely Marathi, reading "प्रतिष्ठा" (Pratista).

Difference Score = 0.88569486

Its a Forged Signature

```
[ ]: predict_score(30)
```

Genuine

Forged



Difference Score = 0.87014043

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged

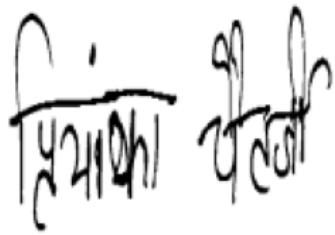


Difference Score = 0.9855374

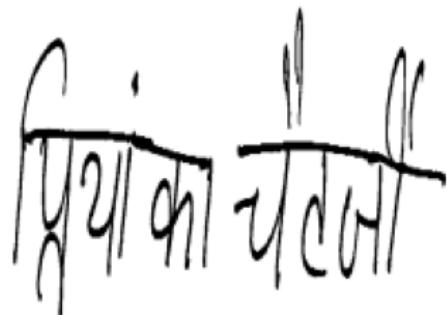
Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



Difference Score = 0.98797905

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine



Genuine



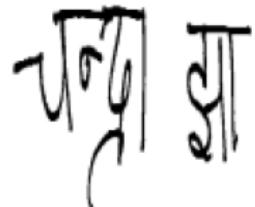
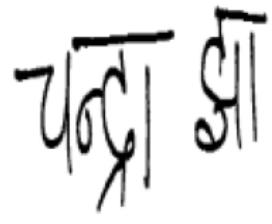
Difference Score = 0.32901025

Its a Genuine Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



Difference Score = 0.43026444

Its a Genuine Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Genuine



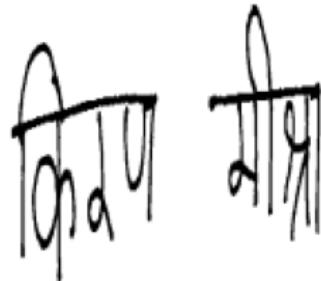
Difference Score = 0.63433075

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



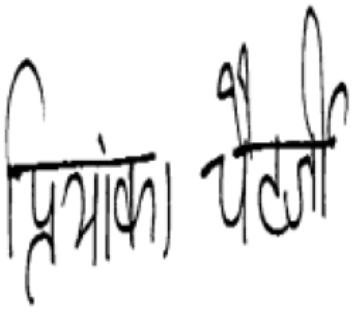
Difference Score = 0.98689026

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



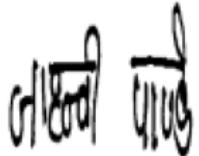
Difference Score = 0.9978987

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



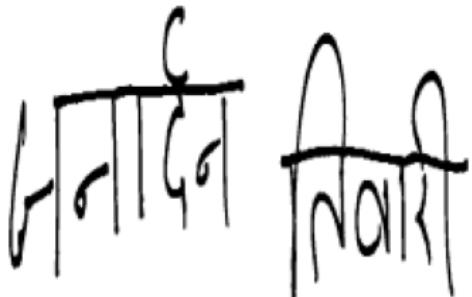
Difference Score = 0.94553477

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



Difference Score = 0.95464015

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



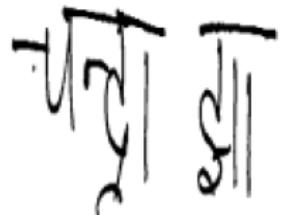
Difference Score = 0.67217237

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged

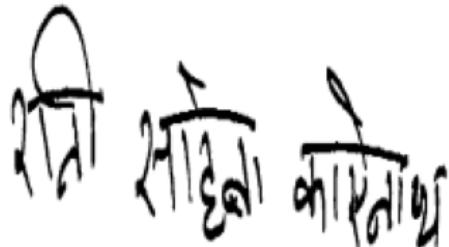


Difference Score = 0.99994135

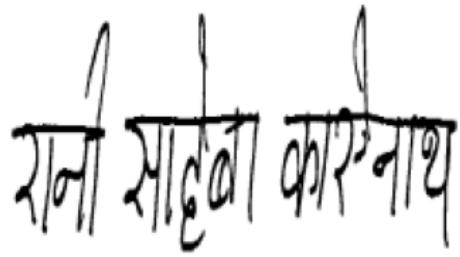
Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine



Forged

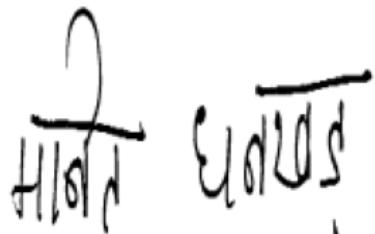


Difference Score = 0.9992669

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine



Forged



Difference Score = 0.9085497

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



Difference Score = 0.90416783

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



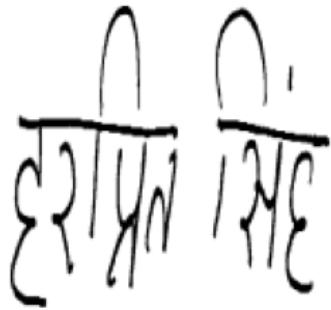
Difference Score = 0.9744856

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Forged



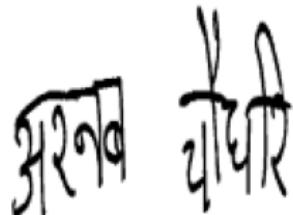
Difference Score = 0.99144185

Its a Forged Signature

```
[ ]: predict_score(randint(0,len(S1_test)))
```

Genuine

Genuine



Difference Score = 0.32901025

Its a Genuine Signature

[]: