In [1]:

```
"""This program trains a binary classification model on image data (human face images
saves the trained model, and classifies new/test images using the model. Also called b
It utilizes the Keras library and follows the typical structure of a Convolutional Neu
The trained model predicts/classifies the test/new images as: 0 - No Eye glasses & 1 -

This program will be hosted online via Streamlit community cloud.
Streamlit web app will be created that allows users to upload an image, classify it us
display the result."""
```

Out[1]:

```
'This program trains a binary classification model on image data (human face images w
ith and without glasses), \nsaves the trained model, and classifies new/test images u
sing the model. Also called binary image classification model.\nIt utilizes the Keras
library and follows the typical structure of a Convolutional Neural Network (CNN) mod
el. \nThe trained model predicts/classifies the test/new images as: 0 - No Eye glasse
s & 1 - Eye glasses present.\n\nThis program will be hosted online via Streamlit comm
unity cloud. \nStreamlit web app will be created that allows users to upload an imag
e, classify it using pre-trained models, and\ndisplay the result.'
```

In [1]:

```
# Importing all necessary libraries
import h5py
import os
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# Setting up training and validation directories
train_data_dir = 'train'
validation_data_dir = 'test'

# Counting the number of images in training and validation directories
train_count = len(os.listdir('train/none')) + len(os.listdir('train/present'))
test_count = len(os.listdir('test/none')) + len(os.listdir('test/present'))

# Setting up image counts for training and validation
nb_train_samples = train_count
nb_validation_samples = test_count

# Setting up training parameters
epochs = 100    # specific for Model 2
batch_size = 100

# Setting image dimensions
img_width, img_height = 224, 224
```

In [3]:

```
# Checking the image data format
if K.image_data_format() == 'channels_first':
        input_shape = (3, img_width, img_height)
else:
        input_shape = (img_width, img_height, 3)
```

In [4]:
```
# Creating a Sequential model
model = Sequential()

# Adding Convolutional and Pooling layers
model.add(Conv2D(32, (2, 2), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening and adding Dense layers
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

In [5]:
```
# Compiling the model
model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

In [6]:
```
# Setting up data augmentation for training
train_datagen = ImageDataGenerator(
        rescale=1. / 255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

# Setting up data normalization for testing
test_datagen = ImageDataGenerator(rescale=1. / 255)

# Creating data generators for training and validation
train_generator = train_datagen.flow_from_directory(train_data_dir,target_size=(img_wi
batch_size=batch_size, class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='binary')

# Training the model
history=model.fit_generator(
    train_generator,
        steps_per_epoch=nb_train_samples // batch_size,
        epochs=epochs,
        validation_data=validation_generator,
        validation_steps=nb_validation_samples // batch_size)
```

```
Found 4418 images belonging to 2 classes.
Found 2371 images belonging to 2 classes.
Epoch 1/100
44/44 [==============================] - 171s 4s/step - loss: 0.5695 - accuracy: 0.74
87 - val_loss: 0.2595 - val_accuracy: 0.9013
Epoch 2/100
44/44 [==============================] - 128s 3s/step - loss: 0.2752 - accuracy: 0.89
65 - val_loss: 0.1315 - val_accuracy: 0.9574
Epoch 3/100
44/44 [==============================] - 121s 3s/step - loss: 0.2107 - accuracy: 0.92
68 - val_loss: 0.1480 - val_accuracy: 0.9530
Epoch 4/100
44/44 [==============================] - 121s 3s/step - loss: 0.1819 - accuracy: 0.93
49 - val_loss: 0.1014 - val_accuracy: 0.9643
Epoch 5/100
44/44 [==============================] - 118s 3s/step - loss: 0.1592 - accuracy: 0.94
84 - val_loss: 0.0983 - val_accuracy: 0.9696
Epoch 6/100
44/44 [==============================] - 124s 3s/step - loss: 0.1463 - accuracy: 0.94
77 - val_loss: 0.1127 - val_accuracy: 0.9661
Epoch 7/100
44/44 [==============================] - 122s 3s/step - loss: 0.1441 - accuracy: 0.94
97 - val_loss: 0.1936 - val_accuracy: 0.9357
Epoch 8/100
44/44 [==============================] - 128s 3s/step - loss: 0.1418 - accuracy: 0.94
97 - val_loss: 0.0989 - val_accuracy: 0.9691
Epoch 9/100
44/44 [==============================] - 115s 3s/step - loss: 0.1248 - accuracy: 0.95
85 - val_loss: 0.1032 - val_accuracy: 0.9709
Epoch 10/100
44/44 [==============================] - 124s 3s/step - loss: 0.1253 - accuracy: 0.95
51 - val_loss: 0.0944 - val_accuracy: 0.9687
Epoch 11/100
44/44 [==============================] - 134s 3s/step - loss: 0.1200 - accuracy: 0.95
81 - val_loss: 0.0892 - val_accuracy: 0.9726
Epoch 12/100
44/44 [==============================] - 128s 3s/step - loss: 0.1159 - accuracy: 0.96
09 - val_loss: 0.0877 - val_accuracy: 0.9730
Epoch 13/100
44/44 [==============================] - 123s 3s/step - loss: 0.1157 - accuracy: 0.95
97 - val_loss: 0.0958 - val_accuracy: 0.9683
Epoch 14/100
44/44 [==============================] - 121s 3s/step - loss: 0.1099 - accuracy: 0.96
11 - val_loss: 0.0983 - val_accuracy: 0.9687
Epoch 15/100
44/44 [==============================] - 128s 3s/step - loss: 0.1127 - accuracy: 0.96
18 - val_loss: 0.0790 - val_accuracy: 0.9743
Epoch 16/100
44/44 [==============================] - 127s 3s/step - loss: 0.0993 - accuracy: 0.96
36 - val_loss: 0.0789 - val_accuracy: 0.9774
Epoch 17/100
44/44 [==============================] - 126s 3s/step - loss: 0.0932 - accuracy: 0.96
11 - val_loss: 0.0877 - val_accuracy: 0.9696
Epoch 18/100
44/44 [==============================] - 122s 3s/step - loss: 0.0998 - accuracy: 0.96
39 - val_loss: 0.0849 - val_accuracy: 0.9735
Epoch 19/100
44/44 [==============================] - 131s 3s/step - loss: 0.0911 - accuracy: 0.96
41 - val_loss: 0.0991 - val_accuracy: 0.9722
Epoch 20/100
```

```
44/44 [==============================] - 123s 3s/step - loss: 0.0918 - accuracy: 0.96
60 - val_loss: 0.0773 - val_accuracy: 0.9765
Epoch 21/100
44/44 [==============================] - 131s 3s/step - loss: 0.0936 - accuracy: 0.96
50 - val_loss: 0.0832 - val_accuracy: 0.9748
Epoch 22/100
44/44 [==============================] - 123s 3s/step - loss: 0.0901 - accuracy: 0.96
60 - val_loss: 0.0870 - val_accuracy: 0.9770
Epoch 23/100
44/44 [==============================] - 121s 3s/step - loss: 0.0917 - accuracy: 0.96
57 - val_loss: 0.1164 - val_accuracy: 0.9687
Epoch 24/100
44/44 [==============================] - 124s 3s/step - loss: 0.0911 - accuracy: 0.96
67 - val_loss: 0.0744 - val_accuracy: 0.9778
Epoch 25/100
44/44 [==============================] - 120s 3s/step - loss: 0.0806 - accuracy: 0.96
87 - val_loss: 0.0963 - val_accuracy: 0.9674
Epoch 26/100
44/44 [==============================] - 126s 3s/step - loss: 0.0800 - accuracy: 0.97
29 - val_loss: 0.0813 - val_accuracy: 0.9717
Epoch 27/100
44/44 [==============================] - 121s 3s/step - loss: 0.0833 - accuracy: 0.96
85 - val_loss: 0.0889 - val_accuracy: 0.9765
Epoch 28/100
44/44 [==============================] - 136s 3s/step - loss: 0.0760 - accuracy: 0.96
92 - val_loss: 0.0757 - val_accuracy: 0.9796
Epoch 29/100
44/44 [==============================] - 126s 3s/step - loss: 0.0770 - accuracy: 0.97
24 - val_loss: 0.0796 - val_accuracy: 0.9787
Epoch 30/100
44/44 [==============================] - 135s 3s/step - loss: 0.0694 - accuracy: 0.97
04 - val_loss: 0.0892 - val_accuracy: 0.9761
Epoch 31/100
44/44 [==============================] - 130s 3s/step - loss: 0.0742 - accuracy: 0.97
31 - val_loss: 0.0869 - val_accuracy: 0.9765
Epoch 32/100
44/44 [==============================] - 129s 3s/step - loss: 0.0755 - accuracy: 0.97
29 - val_loss: 0.0825 - val_accuracy: 0.9791
Epoch 33/100
44/44 [==============================] - 127s 3s/step - loss: 0.0711 - accuracy: 0.97
38 - val_loss: 0.0884 - val_accuracy: 0.9774
Epoch 34/100
44/44 [==============================] - 135s 3s/step - loss: 0.0679 - accuracy: 0.97
55 - val_loss: 0.0804 - val_accuracy: 0.9765
Epoch 35/100
44/44 [==============================] - 139s 3s/step - loss: 0.0681 - accuracy: 0.97
55 - val_loss: 0.0951 - val_accuracy: 0.9752
Epoch 36/100
44/44 [==============================] - 135s 3s/step - loss: 0.0657 - accuracy: 0.97
55 - val_loss: 0.0884 - val_accuracy: 0.9778
Epoch 37/100
44/44 [==============================] - 133s 3s/step - loss: 0.0586 - accuracy: 0.97
82 - val_loss: 0.0986 - val_accuracy: 0.9748
Epoch 38/100
44/44 [==============================] - 126s 3s/step - loss: 0.0636 - accuracy: 0.97
48 - val_loss: 0.0975 - val_accuracy: 0.9787
Epoch 39/100
44/44 [==============================] - 131s 3s/step - loss: 0.0636 - accuracy: 0.97
82 - val_loss: 0.0955 - val_accuracy: 0.9796
Epoch 40/100
```

```
44/44 [==============================] - 136s 3s/step - loss: 0.0569 - accuracy: 0.97
71 - val_loss: 0.1105 - val_accuracy: 0.9674
Epoch 41/100
44/44 [==============================] - 134s 3s/step - loss: 0.0597 - accuracy: 0.97
71 - val_loss: 0.0871 - val_accuracy: 0.9791
Epoch 42/100
44/44 [==============================] - 133s 3s/step - loss: 0.0513 - accuracy: 0.98
03 - val_loss: 0.0995 - val_accuracy: 0.9748
Epoch 43/100
44/44 [==============================] - 135s 3s/step - loss: 0.0519 - accuracy: 0.97
94 - val_loss: 0.0886 - val_accuracy: 0.9796
Epoch 44/100
44/44 [==============================] - 128s 3s/step - loss: 0.0526 - accuracy: 0.97
85 - val_loss: 0.0912 - val_accuracy: 0.9800
Epoch 45/100
44/44 [==============================] - 133s 3s/step - loss: 0.0598 - accuracy: 0.97
71 - val_loss: 0.0873 - val_accuracy: 0.9783
Epoch 46/100
44/44 [==============================] - 132s 3s/step - loss: 0.0518 - accuracy: 0.98
17 - val_loss: 0.0930 - val_accuracy: 0.9791
Epoch 47/100
44/44 [==============================] - 133s 3s/step - loss: 0.0537 - accuracy: 0.98
15 - val_loss: 0.1695 - val_accuracy: 0.9617
Epoch 48/100
44/44 [==============================] - 132s 3s/step - loss: 0.0590 - accuracy: 0.97
82 - val_loss: 0.0809 - val_accuracy: 0.9770
Epoch 49/100
44/44 [==============================] - 135s 3s/step - loss: 0.0521 - accuracy: 0.98
12 - val_loss: 0.1083 - val_accuracy: 0.9787
Epoch 50/100
44/44 [==============================] - 132s 3s/step - loss: 0.0515 - accuracy: 0.98
08 - val_loss: 0.1079 - val_accuracy: 0.9778
Epoch 51/100
44/44 [==============================] - 126s 3s/step - loss: 0.0488 - accuracy: 0.98
31 - val_loss: 0.0979 - val_accuracy: 0.9770
Epoch 52/100
44/44 [==============================] - 126s 3s/step - loss: 0.0548 - accuracy: 0.98
10 - val_loss: 0.0848 - val_accuracy: 0.9791
Epoch 53/100
44/44 [==============================] - 127s 3s/step - loss: 0.0497 - accuracy: 0.98
10 - val_loss: 0.1011 - val_accuracy: 0.9783
Epoch 54/100
44/44 [==============================] - 132s 3s/step - loss: 0.0476 - accuracy: 0.98
17 - val_loss: 0.0928 - val_accuracy: 0.9783
Epoch 55/100
44/44 [==============================] - 126s 3s/step - loss: 0.0432 - accuracy: 0.98
43 - val_loss: 0.1101 - val_accuracy: 0.9778
Epoch 56/100
44/44 [==============================] - 126s 3s/step - loss: 0.0466 - accuracy: 0.98
08 - val_loss: 0.1163 - val_accuracy: 0.9791
Epoch 57/100
44/44 [==============================] - 126s 3s/step - loss: 0.0492 - accuracy: 0.98
38 - val_loss: 0.1051 - val_accuracy: 0.9757
Epoch 58/100
44/44 [==============================] - 125s 3s/step - loss: 0.0423 - accuracy: 0.98
45 - val_loss: 0.1021 - val_accuracy: 0.9778
Epoch 59/100
44/44 [==============================] - 126s 3s/step - loss: 0.0415 - accuracy: 0.98
43 - val_loss: 0.1227 - val_accuracy: 0.9757
Epoch 60/100
```

```
44/44 [==============================] - 125s 3s/step - loss: 0.0434 - accuracy: 0.98
54 - val_loss: 0.1129 - val_accuracy: 0.9778
Epoch 61/100
44/44 [==============================] - 125s 3s/step - loss: 0.0366 - accuracy: 0.98
49 - val_loss: 0.1149 - val_accuracy: 0.9770
Epoch 62/100
44/44 [==============================] - 128s 3s/step - loss: 0.0430 - accuracy: 0.98
33 - val_loss: 0.1094 - val_accuracy: 0.9839
Epoch 63/100
44/44 [==============================] - 126s 3s/step - loss: 0.0383 - accuracy: 0.98
40 - val_loss: 0.1257 - val_accuracy: 0.9774
Epoch 64/100
44/44 [==============================] - 125s 3s/step - loss: 0.0403 - accuracy: 0.98
31 - val_loss: 0.1034 - val_accuracy: 0.9826
Epoch 65/100
44/44 [==============================] - 127s 3s/step - loss: 0.0351 - accuracy: 0.98
55 - val_loss: 0.1221 - val_accuracy: 0.9770
Epoch 66/100
44/44 [==============================] - 125s 3s/step - loss: 0.0406 - accuracy: 0.98
61 - val_loss: 0.1226 - val_accuracy: 0.9796
Epoch 67/100
44/44 [==============================] - 125s 3s/step - loss: 0.0386 - accuracy: 0.98
75 - val_loss: 0.1092 - val_accuracy: 0.9809
Epoch 68/100
44/44 [==============================] - 126s 3s/step - loss: 0.0385 - accuracy: 0.98
52 - val_loss: 0.1273 - val_accuracy: 0.9778
Epoch 69/100
44/44 [==============================] - 125s 3s/step - loss: 0.0405 - accuracy: 0.98
47 - val_loss: 0.1353 - val_accuracy: 0.9800
Epoch 70/100
44/44 [==============================] - 125s 3s/step - loss: 0.0384 - accuracy: 0.98
66 - val_loss: 0.1128 - val_accuracy: 0.9817
Epoch 71/100
44/44 [==============================] - 126s 3s/step - loss: 0.0393 - accuracy: 0.98
49 - val_loss: 0.1462 - val_accuracy: 0.9752
Epoch 72/100
44/44 [==============================] - 125s 3s/step - loss: 0.0363 - accuracy: 0.98
63 - val_loss: 0.1037 - val_accuracy: 0.9830
Epoch 73/100
44/44 [==============================] - 125s 3s/step - loss: 0.0331 - accuracy: 0.98
56 - val_loss: 0.1262 - val_accuracy: 0.9800
Epoch 74/100
44/44 [==============================] - 125s 3s/step - loss: 0.0300 - accuracy: 0.98
96 - val_loss: 0.1375 - val_accuracy: 0.9770
Epoch 75/100
44/44 [==============================] - 126s 3s/step - loss: 0.0352 - accuracy: 0.98
77 - val_loss: 0.1083 - val_accuracy: 0.9809
Epoch 76/100
44/44 [==============================] - 125s 3s/step - loss: 0.0316 - accuracy: 0.98
75 - val_loss: 0.1245 - val_accuracy: 0.9804
Epoch 77/100
44/44 [==============================] - 127s 3s/step - loss: 0.0408 - accuracy: 0.98
66 - val_loss: 0.1154 - val_accuracy: 0.9791
Epoch 78/100
44/44 [==============================] - 126s 3s/step - loss: 0.0290 - accuracy: 0.98
82 - val_loss: 0.1243 - val_accuracy: 0.9778
Epoch 79/100
44/44 [==============================] - 125s 3s/step - loss: 0.0372 - accuracy: 0.98
82 - val_loss: 0.1461 - val_accuracy: 0.9791
Epoch 80/100
```

```
44/44 [==============================] - 125s 3s/step - loss: 0.0358 - accuracy: 0.98
80 - val_loss: 0.1163 - val_accuracy: 0.9778
Epoch 81/100
44/44 [==============================] - 125s 3s/step - loss: 0.0307 - accuracy: 0.98
80 - val_loss: 0.1372 - val_accuracy: 0.9774
Epoch 82/100
44/44 [==============================] - 125s 3s/step - loss: 0.0290 - accuracy: 0.98
77 - val_loss: 0.1203 - val_accuracy: 0.9800
Epoch 83/100
44/44 [==============================] - 126s 3s/step - loss: 0.0307 - accuracy: 0.98
98 - val_loss: 0.1442 - val_accuracy: 0.9800
Epoch 84/100
44/44 [==============================] - 125s 3s/step - loss: 0.0335 - accuracy: 0.98
89 - val_loss: 0.1452 - val_accuracy: 0.9787
Epoch 85/100
44/44 [==============================] - 125s 3s/step - loss: 0.0295 - accuracy: 0.98
82 - val_loss: 0.1337 - val_accuracy: 0.9783
Epoch 86/100
44/44 [==============================] - 127s 3s/step - loss: 0.0303 - accuracy: 0.98
84 - val_loss: 0.1299 - val_accuracy: 0.9817
Epoch 87/100
44/44 [==============================] - 127s 3s/step - loss: 0.0311 - accuracy: 0.98
96 - val_loss: 0.1373 - val_accuracy: 0.9783
Epoch 88/100
44/44 [==============================] - 145s 3s/step - loss: 0.0308 - accuracy: 0.98
80 - val_loss: 0.1329 - val_accuracy: 0.9787
Epoch 89/100
44/44 [==============================] - 138s 3s/step - loss: 0.0246 - accuracy: 0.99
14 - val_loss: 0.1340 - val_accuracy: 0.9791
Epoch 90/100
44/44 [==============================] - 140s 3s/step - loss: 0.0295 - accuracy: 0.98
87 - val_loss: 0.1225 - val_accuracy: 0.9796
Epoch 91/100
44/44 [==============================] - 157s 4s/step - loss: 0.0289 - accuracy: 0.98
87 - val_loss: 0.1544 - val_accuracy: 0.9787
Epoch 92/100
44/44 [==============================] - 135s 3s/step - loss: 0.0334 - accuracy: 0.98
54 - val_loss: 0.1288 - val_accuracy: 0.9787
Epoch 93/100
44/44 [==============================] - 132s 3s/step - loss: 0.0306 - accuracy: 0.98
93 - val_loss: 0.1411 - val_accuracy: 0.9796
Epoch 94/100
44/44 [==============================] - 129s 3s/step - loss: 0.0304 - accuracy: 0.99
03 - val_loss: 0.1122 - val_accuracy: 0.9800
Epoch 95/100
44/44 [==============================] - 129s 3s/step - loss: 0.0274 - accuracy: 0.99
14 - val_loss: 0.1495 - val_accuracy: 0.9830
Epoch 96/100
44/44 [==============================] - 129s 3s/step - loss: 0.0284 - accuracy: 0.99
12 - val_loss: 0.1215 - val_accuracy: 0.9796
Epoch 97/100
44/44 [==============================] - 131s 3s/step - loss: 0.0285 - accuracy: 0.98
84 - val_loss: 0.1333 - val_accuracy: 0.9817
Epoch 98/100
44/44 [==============================] - 128s 3s/step - loss: 0.0282 - accuracy: 0.99
00 - val_loss: 0.1495 - val_accuracy: 0.9813
Epoch 99/100
44/44 [==============================] - 130s 3s/step - loss: 0.0258 - accuracy: 0.98
98 - val_loss: 0.1665 - val_accuracy: 0.9770
Epoch 100/100
```

```
44/44 [==============================] - 196s 4s/step - loss: 0.0235 - accuracy: 0.99
17 - val_loss: 0.1434 - val_accuracy: 0.9817
```

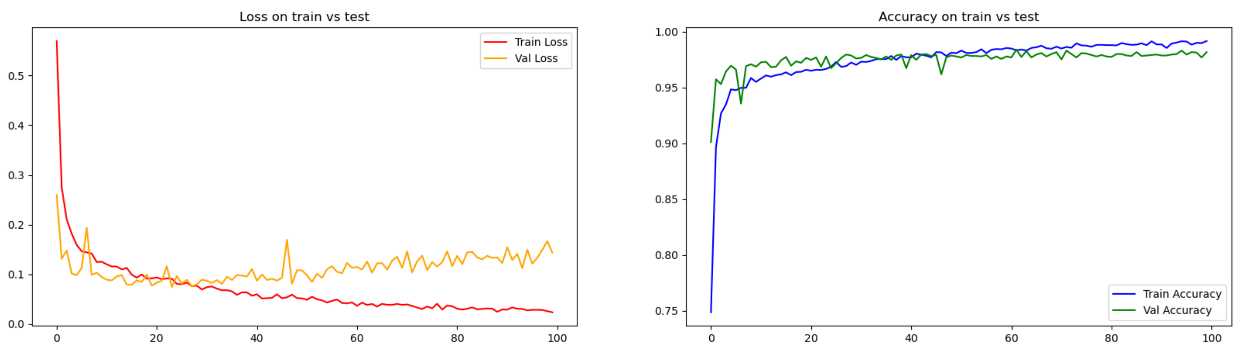
In [7]:

```
# Plotting loss and accuracy over epochs
plt.figure(figsize=(20,5))

# Plotting loss & validation loss
plt.subplot(1,2,1)
sns.lineplot(x=history.epoch, y=history.history['loss'], color='red', label='Train Los
sns.lineplot(x=history.epoch, y=history.history['val_loss'], color='orange', label='Va
plt.title('Loss on train vs test')
plt.legend(loc='best')

# Plotting accuracy and validation accuracy
plt.subplot(1,2,2)
sns.lineplot(x=history.epoch, y=history.history['accuracy'], color='blue', label='Trai
sns.lineplot(x=history.epoch, y=history.history['val_accuracy'], color='green', label=
plt.title('Accuracy on train vs test')
plt.legend(loc='best')

plt.show()
```



In [8]:

```
# Saving the trained model
model.save('Model3.h5')
```

In [3]:

```
# Generating Classification report
from sklearn.metrics import classification_report

# Setting up data augmentation for training
train_datagen = ImageDataGenerator(
        rescale=1. / 255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

# Setting up data normalization for testing
test_datagen = ImageDataGenerator(rescale=1. / 255)

# Creating data generators for training and validation
train_generator = train_datagen.flow_from_directory(train_data_dir,target_size=(img_wi
batch_size=batch_size, class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='binary')
```

```
# Load the saved model
from keras.models import load_model
model = load_model('Model3.h5')

# Generate predictions for the validation dataset
validation_generator.reset()  # Reset generator to start from beginning
y_pred = model.predict_generator(validation_generator, steps=len(validation_generator)
y_pred_binary = (y_pred > 0.5).astype(int)  # Convert probabilities to binary predicti

# Get true labels
y_true = validation_generator.classes

# Generate classification report
print(classification_report(y_true, y_pred_binary))
```

```
Found 4418 images belonging to 2 classes.
Found 2371 images belonging to 2 classes.
24/24 [==============================] - 17s 670ms/step
              precision    recall  f1-score   support

           0       0.64      0.64      0.64      1502
           1       0.38      0.39      0.38       869

    accuracy                           0.55      2371
   macro avg       0.51      0.51      0.51      2371
weighted avg       0.55      0.55      0.55      2371
```

In [4]:

```
# Loading the trained model for predictions
from keras.models import load_model
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg16 import decode_predictions
from keras.applications.vgg16 import VGG16
import os
import numpy as np

model = load_model('Model3.h5')
```

In [11]:

```
# Making predictions on test/new images using the trained model from the above.
import glob

# Setting the directory for testing images
folder_dir = "/Users/tabal/OneDrive/Desktop/Data Capstone/Pictures for Testing"

# Looping through each image in the directory
for image in glob.iglob(f'{folder_dir}/*'):

    # Loading and preprocessing the image
    load_image = load_img(image, target_size=(224, 224))
    img = img_to_array(load_image)
    img = preprocess_input(img.reshape(1,224,224,3))

    # Making predictions using the loaded model
    label = model.predict(img)

    # Displaying the predicted class (0 - No Eye glasses , 1 - Eye glasses present) fo
    print("Predicted Class (0 - None , 1- Present) for ", os.path.basename(image), " i
```

```
1/1 [==============================] - 0s 27ms/step
Predicted Class (0 - None , 1- Present) for  1.jpg  is: 0
1/1 [==============================] - 0s 25ms/step
Predicted Class (0 - None , 1- Present) for  10.jpg  is: 0
1/1 [==============================] - 0s 23ms/step
Predicted Class (0 - None , 1- Present) for  11.png  is: 1
1/1 [==============================] - 0s 23ms/step
Predicted Class (0 - None , 1- Present) for  12.png  is: 1
1/1 [==============================] - 0s 15ms/step
Predicted Class (0 - None , 1- Present) for  13.jpg  is: 1
1/1 [==============================] - 0s 22ms/step
Predicted Class (0 - None , 1- Present) for  14.jpg  is: 1
1/1 [==============================] - 0s 25ms/step
Predicted Class (0 - None , 1- Present) for  15.jpg  is: 0
1/1 [==============================] - 0s 24ms/step
Predicted Class (0 - None , 1- Present) for  16.jpg  is: 1
1/1 [==============================] - 0s 25ms/step
Predicted Class (0 - None , 1- Present) for  17.jpg  is: 1
1/1 [==============================] - 0s 33ms/step
Predicted Class (0 - None , 1- Present) for  18.jpg  is: 1
1/1 [==============================] - 0s 22ms/step
Predicted Class (0 - None , 1- Present) for  19.jpg  is: 1
1/1 [==============================] - 0s 23ms/step
Predicted Class (0 - None , 1- Present) for  2.jpg  is: 0
1/1 [==============================] - 0s 25ms/step
Predicted Class (0 - None , 1- Present) for  20.jpg  is: 1
1/1 [==============================] - 0s 21ms/step
Predicted Class (0 - None , 1- Present) for  21.jpg  is: 0
1/1 [==============================] - 0s 22ms/step
Predicted Class (0 - None , 1- Present) for  22.jpg  is: 1
1/1 [==============================] - 0s 22ms/step
Predicted Class (0 - None , 1- Present) for  23.jpg  is: 1
1/1 [==============================] - 0s 20ms/step
Predicted Class (0 - None , 1- Present) for  24.png  is: 0
1/1 [==============================] - 0s 25ms/step
Predicted Class (0 - None , 1- Present) for  25.png  is: 1
1/1 [==============================] - 0s 24ms/step
Predicted Class (0 - None , 1- Present) for  26.png  is: 1
1/1 [==============================] - 0s 20ms/step
Predicted Class (0 - None , 1- Present) for  27.png  is: 1
1/1 [==============================] - 0s 21ms/step
Predicted Class (0 - None , 1- Present) for  28.png  is: 0
1/1 [==============================] - 0s 16ms/step
Predicted Class (0 - None , 1- Present) for  29.png  is: 0
1/1 [==============================] - 0s 25ms/step
Predicted Class (0 - None , 1- Present) for  3.png  is: 0
1/1 [==============================] - 0s 35ms/step
Predicted Class (0 - None , 1- Present) for  30.png  is: 0
1/1 [==============================] - 0s 22ms/step
Predicted Class (0 - None , 1- Present) for  31.png  is: 0
1/1 [==============================] - 0s 26ms/step
Predicted Class (0 - None , 1- Present) for  32.png  is: 0
1/1 [==============================] - 0s 25ms/step
Predicted Class (0 - None , 1- Present) for  33.jpg  is: 1
1/1 [==============================] - 0s 22ms/step
Predicted Class (0 - None , 1- Present) for  34.jpg  is: 1
1/1 [==============================] - 0s 15ms/step
Predicted Class (0 - None , 1- Present) for  35.jpg  is: 0
1/1 [==============================] - 0s 31ms/step
Predicted Class (0 - None , 1- Present) for  4.jpg  is: 0
```

```
1/1 [==============================] - 0s 22ms/step
Predicted Class (0 - None , 1- Present) for  5.jpg  is: 0
1/1 [==============================] - 0s 21ms/step
Predicted Class (0 - None , 1- Present) for  6.jpg  is: 0
1/1 [==============================] - 0s 24ms/step
Predicted Class (0 - None , 1- Present) for  7.jpg  is: 0
1/1 [==============================] - 0s 21ms/step
Predicted Class (0 - None , 1- Present) for  8.jpg  is: 0
1/1 [==============================] - 0s 21ms/step
Predicted Class (0 - None , 1- Present) for  9.png  is: 0
```

In [10]:

```
# Making predictions on test/new images using the trained model from the above and exp
# results to an Excel file.

import glob
import pandas as pd

# Setting the directory for testing images
# folder_dir = "/Users/tabal/OneDrive/Desktop/Data Capstone/Pictures for Testing"
folder_dir = "/Users/tabal/OneDrive/Desktop/Data Capstone/Occulusion_jpg"

# Create an empty list to store prediction results
prediction_results = []

# Looping through each image in the directory
for image in glob.iglob(f'{folder_dir}/*'):

    # Loading and preprocessing the image
    load_image = load_img(image, target_size=(224, 224))
    img = img_to_array(load_image)
    img = preprocess_input(img.reshape(1,224,224,3))

    # Making predictions using the loaded model
    label = model.predict(img)

    # Append the prediction results to the list
    prediction_results.append({
        'Image Name': os.path.basename(image),
        'Prediction': round(label[0][0])})

# Create a DataFrame from the list
df = pd.DataFrame(prediction_results)

# Save the DataFrame to an Excel file
df.to_excel('prediction_results_Model3', index=False)
```

```
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 34ms/step
```

```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 31ms/step
```

```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 34ms/step
```

In [ ]: