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 constructs a custom model on top of the pre-trained base model employing transfer l
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 eyeglasses), \nsaves the trained model, and classifies new/test image
s using the model. Also called binary image classification model. \nIt constructs a c
ustom model on top of the pre-trained base model employing transfer learning.\nThe tr
ained model predicts/classifies the test/new images as: 0 - No Eye glasses & 1 - Eye
glasses present.\n\nThis program will be hosted online via Streamlit community cloud.
\nStreamlit web app will be created that allows users to upload an image, classify it
using pre-trained models, and\ndisplay the result.'
```

In [1]:
```
# Importing libraries
import os
from keras.applications import MobileNetV2
from keras.models import Sequential
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator

# Define training and validation directories
train_data_dir = 'train'
validation_data_dir = 'test'

# Image dimensions
img_width, img_height = 224, 224

# Setting up training parameters
epochs = 10
batch_size = 100

# Create base model with pre-trained weights on ImageNet
base_model = MobileNetV2(input_shape=(img_width, img_height, 3), include_top=False, we

# Freeze convolutional layers
for layer in base_model.layers:
    layer.trainable = False

# Build your own model on top of the base model
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer=Adam(lr=0.001), loss='binary_crossentropy', metrics=['accuracy

# Data augmentation for training
train_datagen = ImageDataGenerator(
```

```
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# Normalization for validation/testing
test_datagen = ImageDataGenerator(rescale=1. / 255)

# Create data generators
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    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')

# Train the model
history = model.fit_generator(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=epochs,  # You can adjust the number of epochs
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size)

# Save the trained model
model.save('Model4.h5')
```

```
Found 4418 images belonging to 2 classes.
```

```
C:\Users\tabal\anaconda3\lib\site-packages\keras\optimizers\legacy\adam.py:117: UserW
arning: The `lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

```
Found 2371 images belonging to 2 classes.
```

```
C:\Users\tabal\AppData\Local\Temp\ipykernel_9300\2759565614.py:61: UserWarning: `Mode
l.fit_generator` is deprecated and will be removed in a future version. Please use `M
odel.fit`, which supports generators.
  history = model.fit_generator(
```

```
Epoch 1/10
44/44 [==============================] - 138s 3s/step - loss: 0.2277 - accuracy: 0.90
50 - val_loss: 0.0788 - val_accuracy: 0.9678
Epoch 2/10
44/44 [==============================] - 139s 3s/step - loss: 0.1279 - accuracy: 0.95
00 - val_loss: 0.0587 - val_accuracy: 0.9791
Epoch 3/10
44/44 [==============================] - 143s 3s/step - loss: 0.1130 - accuracy: 0.95
99 - val_loss: 0.0541 - val_accuracy: 0.9835
Epoch 4/10
44/44 [==============================] - 143s 3s/step - loss: 0.0955 - accuracy: 0.96
62 - val_loss: 0.0431 - val_accuracy: 0.9857
Epoch 5/10
44/44 [==============================] - 142s 3s/step - loss: 0.0936 - accuracy: 0.96
78 - val_loss: 0.1024 - val_accuracy: 0.9539
Epoch 6/10
44/44 [==============================] - 142s 3s/step - loss: 0.0877 - accuracy: 0.96
69 - val_loss: 0.0476 - val_accuracy: 0.9809
Epoch 7/10
44/44 [==============================] - 142s 3s/step - loss: 0.0810 - accuracy: 0.97
06 - val_loss: 0.0595 - val_accuracy: 0.9761
Epoch 8/10
44/44 [==============================] - 142s 3s/step - loss: 0.0832 - accuracy: 0.96
99 - val_loss: 0.0563 - val_accuracy: 0.9783
Epoch 9/10
44/44 [==============================] - 143s 3s/step - loss: 0.0785 - accuracy: 0.97
36 - val_loss: 0.0648 - val_accuracy: 0.9730
Epoch 10/10
44/44 [==============================] - 142s 3s/step - loss: 0.0691 - accuracy: 0.97
55 - val_loss: 0.0540 - val_accuracy: 0.9796
```

In [3]:

```python
# Generating Classification report

import os
from keras.applications import MobileNetV2
from keras.models import Sequential
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator

# Define training and validation directories
train_data_dir = 'train'
validation_data_dir = 'test'

# Image dimensions
img_width, img_height = 224, 224

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

# Normalization for validation/testing
test_datagen = ImageDataGenerator(rescale=1. / 255)

# Create data generators
_from_directory(
    train_data_dir,
```

```
    target_size=(img_width, img_height),
    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')

from sklearn.metrics import classification_report

# Load the saved model
from keras.models import load_model
model = load_model('Model4.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))
```

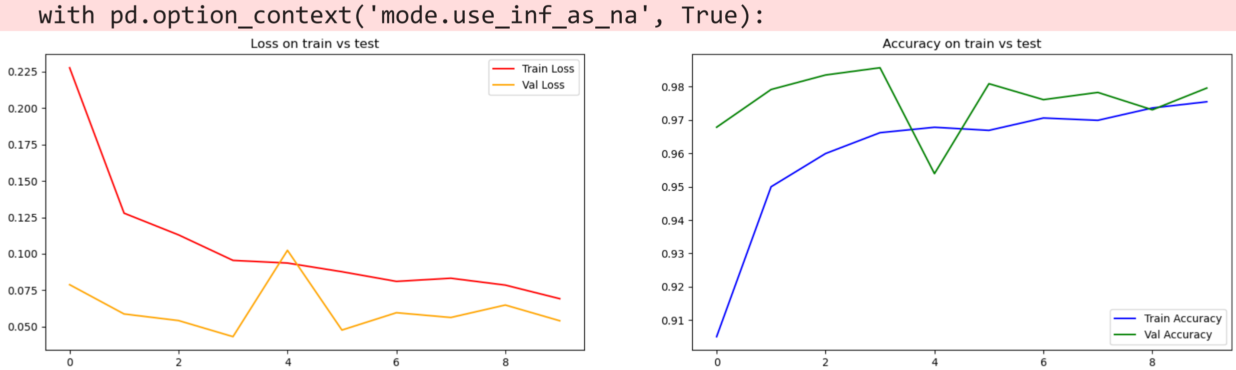
```
Found 4418 images belonging to 2 classes.
Found 2371 images belonging to 2 classes.
C:\Users\tabal\AppData\Local\Temp\ipykernel_16872\2778250781.py:46: UserWarning: `Mod
el.predict_generator` is deprecated and will be removed in a future version. Please u
se `Model.predict`, which supports generators.
  y_pred = model.predict_generator(validation_generator, steps=len(validation_generat
or), verbose=1)
24/24 [==============================] - 38s 2s/step
              precision    recall  f1-score   support

           0       0.63      0.64      0.64      1502
           1       0.37      0.36      0.36       869

    accuracy                           0.54      2371
   macro avg       0.50      0.50      0.50      2371
weighted avg       0.53      0.54      0.54      2371
```

In [2]:

```
# Plotting loss and accuracy over epochs
import matplotlib.pyplot as plt
import seaborn as sns
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
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()
```

```
C:\Users\tabal\anaconda3\lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\tabal\anaconda3\lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\tabal\anaconda3\lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\tabal\anaconda3\lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\tabal\anaconda3\lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\tabal\anaconda3\lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\tabal\anaconda3\lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\tabal\anaconda3\lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: u
se_inf_as_na option is deprecated and will be removed in a future version. Convert in
f values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [1]:

```
# Load the trained model for predictions, preprocess them,
# and use the trained model to predict the class for each image in batch.
# Also export classification results to an Excel file.

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
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
import numpy as np
import os

model = load_model('Model4.h5')

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

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

# 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('Occulusion_jpg_Model4.xlsx', index=False)
```

```
1/1 [==============================] - 1s 765ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 46ms/step
```

In [ ]: