Final Assignment

Amy Rosato

Chetan Shah

Swagat Kumar Pendayla

FakeBook!

In [1]:
```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\shah_\anaconda3\lib\site-pac
kages (4.6.0.66)
Requirement already satisfied: numpy>=1.14.5 in c:\users\shah_\anaconda3\lib\site-pac
kages (from opencv-python) (1.23.4)
Note: you may need to restart the kernel to use updated packages.
```

Importing the libraries and datasets

In [2]:
```
#import the libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
def print_image_count_from_directory(path):
  # Create an empty list to store the images
  images = []

  # Loop through all files in the dataset directory
  for file_name in os.listdir(path):
  # Check if the file is an image
   if file_name.endswith(".jpg") or file_name.endswith(".png"):
      # Load the image using cv2 and the IMREAD_COLOR flag
    image = cv2.imread(os.path.join(path, file_name), cv2.IMREAD_COLOR)
    # Convert the image from BGR to RGB
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # Add the image to the list
    images.append(image)

  # Print the number of images in the dataset
  print(f"Number of images: {len(images)}")
  return images
```

In [7]:
```
# Set the directory where the fake photos are located
real_dir = "C:\\Users\\shah_\\Downloads\\archive\\real_and_fake_face\\training_real"

real_images =  print_image_count_from_directory(real_dir)
```

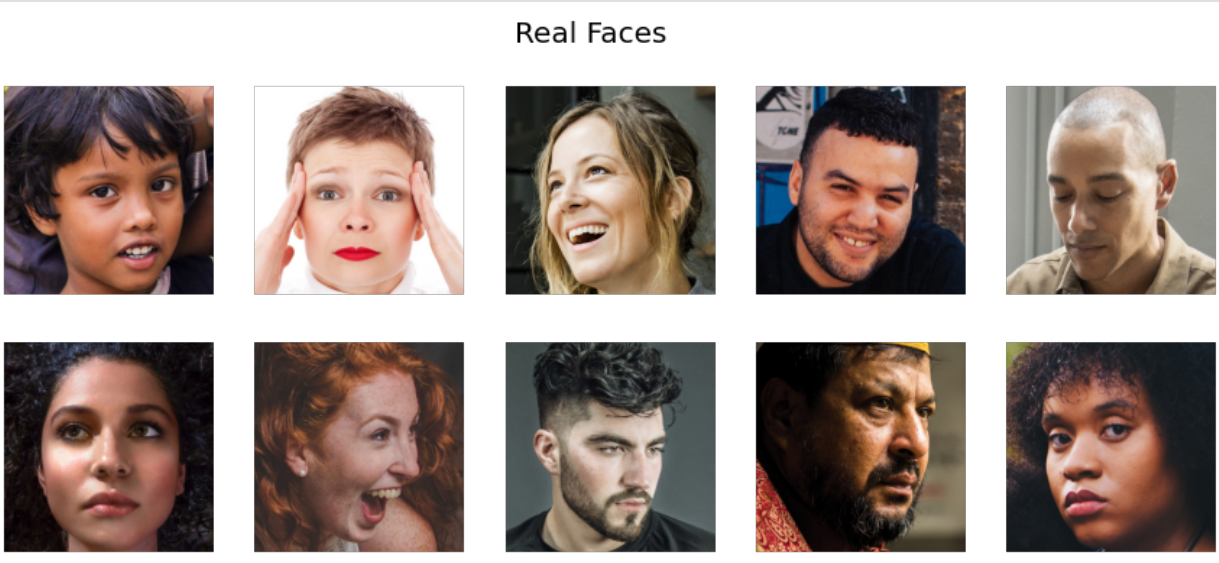
```
Number of images: 1081
```

In [8]:
```
# Set the directory where the fake photos are located
fake_dir = "C:\\Users\\shah_\\Downloads\\archive\\real_and_fake_face\\training_fake"

fake_images =  print_image_count_from_directory(fake_dir)
```

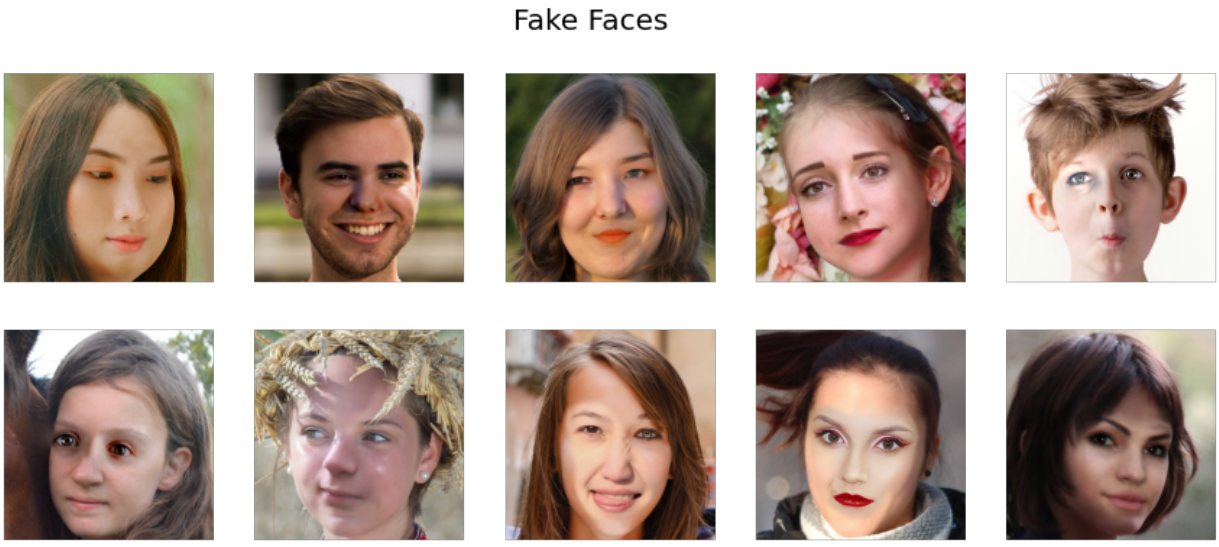
```
Number of images: 960
```

In [9]:
```
def print_images_from_listofImages(images,label):
  # plot the first 10 images in colour of real dataset
  fig, ax = plt.subplots(2, 5, figsize=(15, 6))
  for i in range(10):
      ax[i//5, i%5].imshow(images[i])
      ax[i//5, i%5].axis("off")
      plt.suptitle(label, fontsize=20)
  plt.show()
```

In [10]:
```
print_images_from_listofImages(real_images,"Real Faces")
```



In [11]:
```
print_images_from_listofImages(fake_images,"Fake Faces")
```



In [12]:
```
# Create a list of labels for the real images
real_labels = ["real" for image in real_images]

# Create a list of labels for the fake images
fake_labels = ["fake" for image in fake_images]
```

In [13]:
```
#print the shape of the first image in the real dataset
print(real_images[0].shape)
```

```
(600, 600, 3)
```

In [14]:
```
#print the shape of the first image in the fake dataset
print(fake_images[0].shape)
```

```
(600, 600, 3)
```

Preprocessing

In [15]:
```
#combine the real and fake images directories
images = fake_images + real_images

# Print the number of images in the dataset
print(f"Number of images: {len(images)}")
```

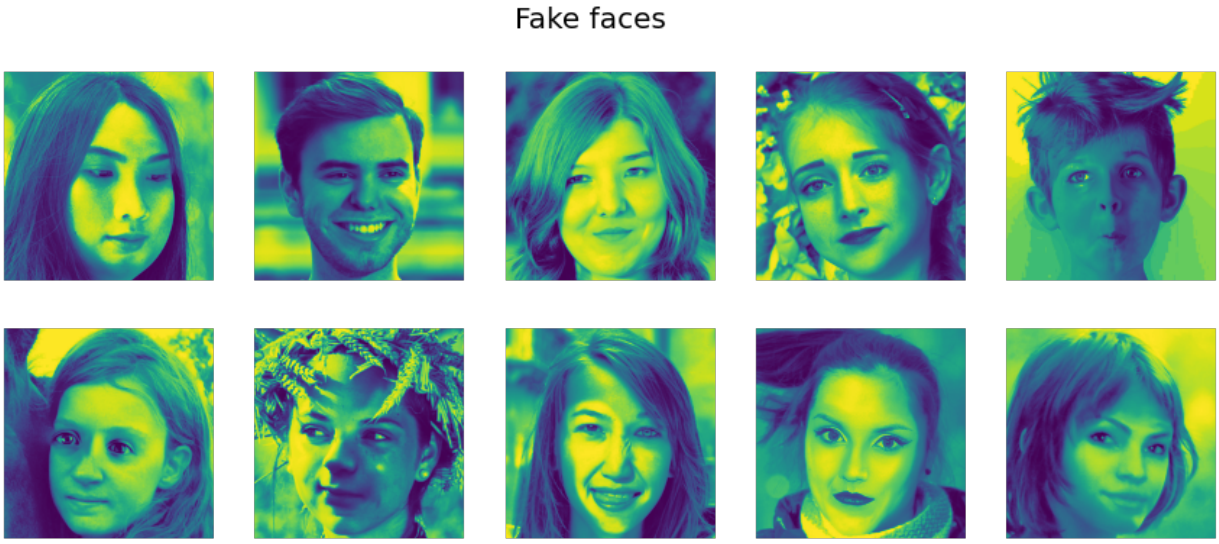
```
Number of images: 2041
```

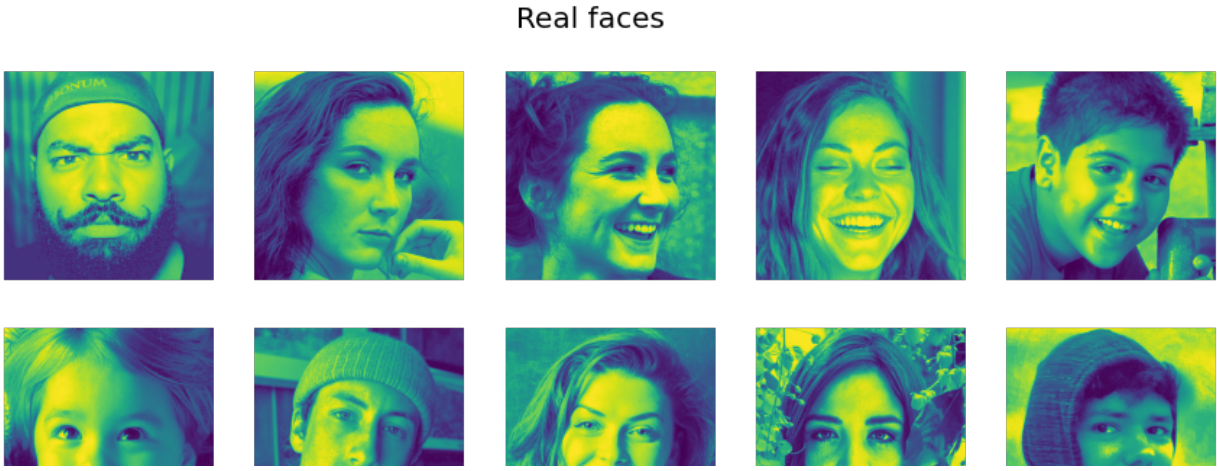
In [24]:
```
# Combine the real and fake labels into a single list of labels
labels = real_labels + fake_labels
```

In [16]:

```
#perform preprocessing on the fake_images and real_images
# Create an empty list to store the preprocessed images
preprocessed_images = []

# Loop through all images in the dataset
for image in images:
    # Convert the image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    # Apply histogram equalization
    equalized = cv2.equalizeHist(gray)
    # Add the preprocessed image to the list
    preprocessed_images.append(equalized)

# Print the number of images in the dataset
print(f"Number of images: {len(preprocessed_images)}")


print_images_from_listofImages(preprocessed_images,"Fake faces")
#plot the first 10 images in grayscale of fake dataset
#fig, ax = plt.subplots(2, 5, figsize=(15, 6))
#for i in range(10):
#    ax[i//5, i%5].imshow(preprocessed_images[i], cmap="gray")
#    ax[i//5, i%5].axis("off")
#    plt.suptitle("Fake faces", fontsize=20)
#plt.show()


print_images_from_listofImages(preprocessed_images[1000:],"Real faces")

#plot the first 10 images in grayscale of real dataset
#fig, ax = plt.subplots(2, 5, figsize=(15, 6))
#for i in range(10):
#    ax[i//5, i%5].imshow(preprocessed_images[i+1000], cmap="gray")
#    ax[i//5, i%5].axis("off")
#    plt.suptitle("Real faces", fontsize=20)
#plt.show()
```

```
Number of images: 2041
```

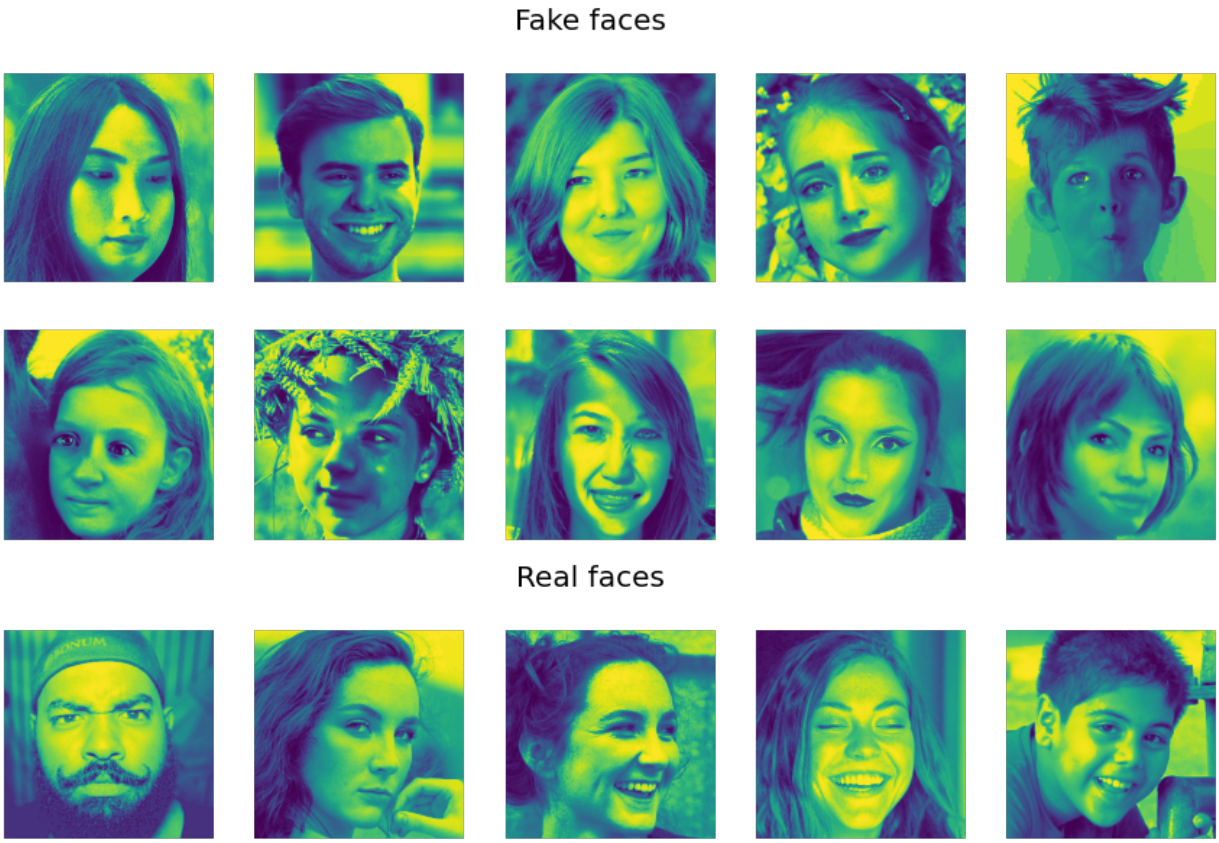




In [80]:

```
#make sure all the images are the same size and shape store in preprocessed_images va
# Create an empty list to store the preprocessed images
resized_images = []

# Loop through all images in the dataset
for image in preprocessed_images:
    # Resize the image to 100x100 pixels
    resized = cv2.resize(image, (244, 244))
    # Add the preprocessed image to the list
    resized_images.append(resized)

# Print the number of images in the dataset
print(f"Number of images: {len(resized_images)}")

print_images_from_listofImages(resized_images,"Fake faces")

#plot the first 10 images in grayscale of fake dataset
#fig, ax = plt.subplots(2, 5, figsize=(15, 6))
#for i in range(10):
#    ax[i//5, i%5].imshow(resized_images[i], cmap="gray")
#    ax[i//5, i%5].axis("off")
#    plt.suptitle("Fake faces", fontsize=20)
#plt.show()


print_images_from_listofImages(resized_images[1000:],"Real faces")

#plot the first 10 images in grayscale of real dataset
#fig, ax = plt.subplots(2, 5, figsize=(15, 6))
#for i in range(10):
#    ax[i//5, i%5].imshow(resized_images[i+1000], cmap="gray")
#    ax[i//5, i%5].axis("off")
#    plt.suptitle("Real faces", fontsize=20)
#plt.show()
```

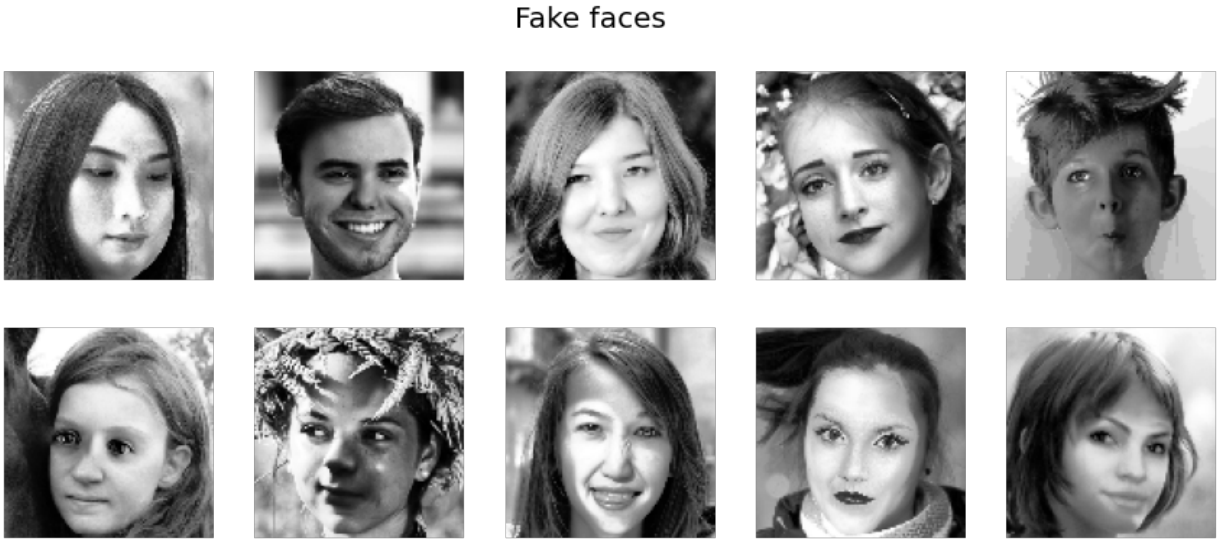
```
Number of images: 2041
```

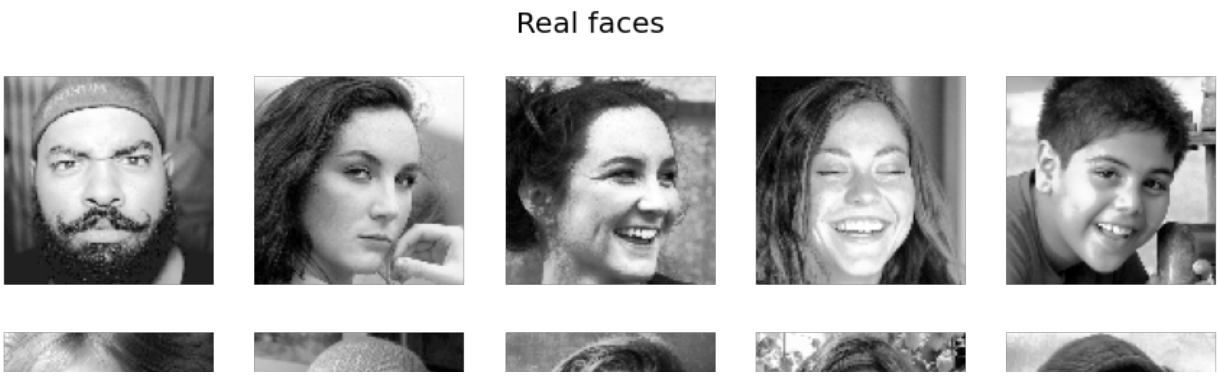

Fake faces
Real faces

In [18]:

```
#reshape the images to be 100x100x1
# Create an empty list to store the preprocessed images
reshaped_images = []

# Loop through all images in the dataset
for image in resized_images:
    # Reshape the image to 3 dimensions
    reshaped = image.reshape( 100, 100,1)
    # Add the preprocessed image to the list
    reshaped_images.append(reshaped)

# Print the number of images in the dataset
print(f"Number of images: {len(reshaped_images)}")


#print_images_from_listofImages(reshaped_images,"Fake faces")

#plot the first 10 images in grayscale of fake dataset
fig, ax = plt.subplots(2, 5, figsize=(15, 6))
for i in range(10):
    ax[i//5, i%5].imshow(reshaped_images[i].squeeze(), cmap="gray")
    ax[i//5, i%5].axis("off")
    plt.suptitle("Fake faces", fontsize=20)
plt.show()


#plot the first 10 images in grayscale of real dataset
fig, ax = plt.subplots(2, 5, figsize=(15, 6))
for i in range(10):
    ax[i//5, i%5].imshow(reshaped_images[i+1000].squeeze(), cmap="gray")
    ax[i//5, i%5].axis("off")
    plt.suptitle("Real faces", fontsize=20)
plt.show()
```

```
Number of images: 2041
```

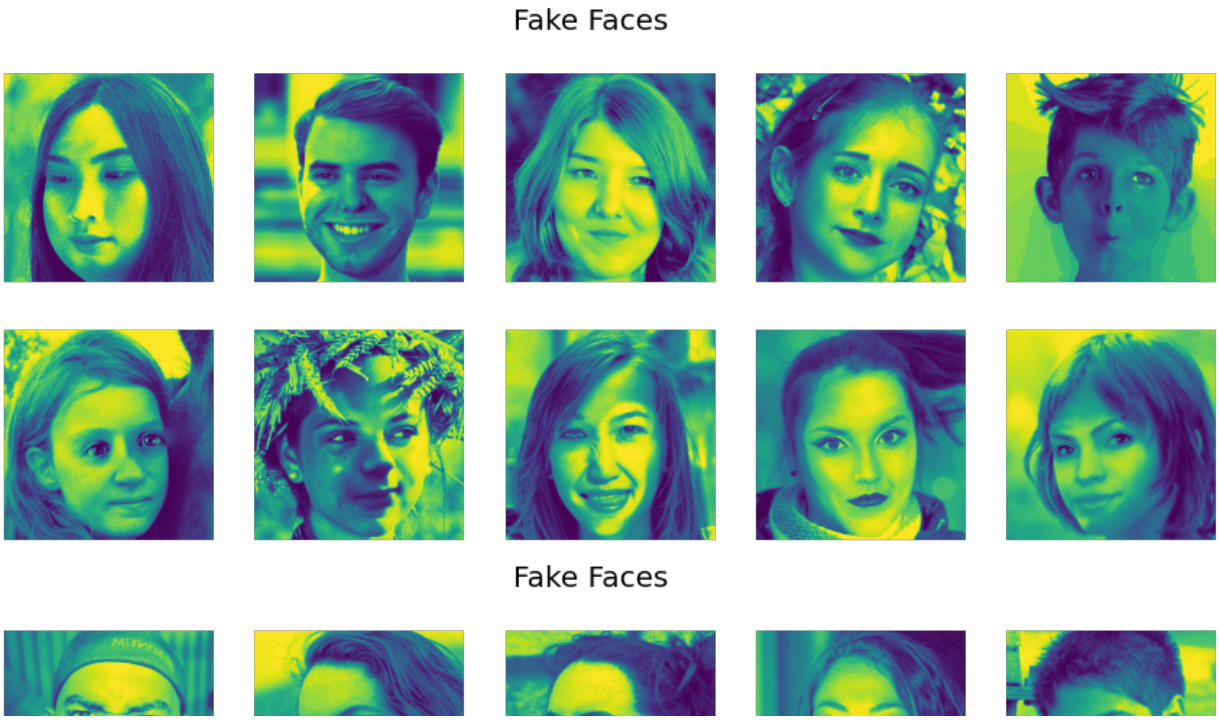




In [81]:

```
#apply data augmentation to the dataset
# Create an empty list to store the preprocessed images
augmented_images = []

# Loop through all images in the dataset
for image in resized_images:
    # Flip the image horizontally
    flipped = cv2.flip(image, 1)
    # Add the preprocessed image to the list
    augmented_images.append(flipped)

# Print the number of images in the dataset
print(f"Number of augmented images: {len(augmented_images)}")

# Create a new list of labels for the augmented images
#augmented_labels = ["real" if label == "real" else "fake" for label in labels]

print_images_from_listofImages(augmented_images,"Fake Faces")

#plot the first 10 images in grayscale of fake dataset


print_images_from_listofImages(augmented_images[1000:],"Fake Faces")

#plot the first 10 images in grayscale of real dataset

#combine the real and fake images directories
#augmented_images = augmented_images + reshaped_images

# Print the number of images in the dataset
print(f"Number of images: {len(augmented_images)}")
```

```
Number of augmented images: 2041
```



IMPORTING THE PRETRAINED MODEL

In [21]:
```
#install tensorflow
#!pip install tensorflow
```

```
Collecting tensorflow
  Downloading tensorflow-2.11.0-cp38-cp38-win_amd64.whl (1.9 kB)
Collecting tensorflow-intel==2.11.0
  Downloading tensorflow_intel-2.11.0-cp38-cp38-win_amd64.whl (266.3 MB)
     ---------------------------------------- 266.3/266.3 MB 2.1 MB/s eta 0:00:00
Collecting tensorflow-io-gcs-filesystem>=0.23.1
  Downloading tensorflow_io_gcs_filesystem-0.28.0-cp38-cp38-win_amd64.whl (1.5 MB)
     ---------------------------------------- 1.5/1.5 MB 5.6 MB/s eta 0:00:00
Collecting termcolor>=1.1.0
  Downloading termcolor-2.1.1-py3-none-any.whl (6.2 kB)
Collecting libclang>=13.0.0
  Using cached libclang-14.0.6-py2.py3-none-win_amd64.whl (14.2 MB)
Collecting tensorboard<2.12,>=2.11
  Downloading tensorboard-2.11.0-py3-none-any.whl (6.0 MB)
     ---------------------------------------- 6.0/6.0 MB 5.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.20 in c:\users\shah_\anaconda3\lib\site-packa
ges (from tensorflow-intel==2.11.0->tensorflow) (1.23.4)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\shah_\anaconda3\l
ib\site-packages (from tensorflow-intel==2.11.0->tensorflow) (3.7.4.3)
Requirement already satisfied: h5py>=2.9.0 in c:\users\shah_\anaconda3\lib\site-packa
ges (from tensorflow-intel==2.11.0->tensorflow) (2.10.0)
Collecting flatbuffers>=2.0
  Downloading flatbuffers-22.11.23-py2.py3-none-any.whl (26 kB)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\shah_\anaconda3\lib\site-pac
kages (from tensorflow-intel==2.11.0->tensorflow) (1.12.1)
Requirement already satisfied: setuptools in c:\users\shah_\anaconda3\lib\site-packag
es (from tensorflow-intel==2.11.0->tensorflow) (52.0.0.post20210125)
Collecting opt-einsum>=2.3.2
  Using cached opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Requirement already satisfied: packaging in c:\users\shah_\anaconda3\lib\site-package
s (from tensorflow-intel==2.11.0->tensorflow) (20.9)
```

```
Collecting tensorflow-estimator<2.12,>=2.11.0
  Downloading tensorflow_estimator-2.11.0-py2.py3-none-any.whl (439 kB)
     ---------------------------------------- 439.2/439.2 kB 2.0 MB/s eta 0:00:00
Collecting keras<2.12,>=2.11.0
  Downloading keras-2.11.0-py2.py3-none-any.whl (1.7 MB)
     ---------------------------------------- 1.7/1.7 MB 7.1 MB/s eta 0:00:00
Collecting astunparse>=1.6.0
  Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting google-pasta>=0.1.1
  Using cached google_pasta-0.2.0-py3-none-any.whl (57 kB)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in c:\users\shah_\anaconda3\lib\
site-packages (from tensorflow-intel==2.11.0->tensorflow) (3.19.6)
Collecting absl-py>=1.0.0
  Using cached absl_py-1.3.0-py3-none-any.whl (124 kB)
Requirement already satisfied: six>=1.12.0 in c:\users\shah_\anaconda3\lib\site-packa
ges (from tensorflow-intel==2.11.0->tensorflow) (1.15.0)
Collecting grpcio<2.0,>=1.24.3
  Downloading grpcio-1.51.1-cp38-cp38-win_amd64.whl (3.7 MB)
     ---------------------------------------- 3.7/3.7 MB 8.3 MB/s eta 0:00:00
Collecting gast<=0.4.0,>=0.2.1
  Using cached gast-0.4.0-py3-none-any.whl (9.8 kB)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\shah_\anaconda3\lib\sit
e-packages (from astunparse>=1.6.0->tensorflow-intel==2.11.0->tensorflow) (0.36.2)
Collecting google-auth-oauthlib<0.5,>=0.4.1
  Using cached google_auth_oauthlib-0.4.6-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\shah_\anaconda3\lib\si
te-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (2.2
5.1)
Collecting markdown>=2.6.8
  Using cached Markdown-3.4.1-py3-none-any.whl (93 kB)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\users\shah_\anacon
da3\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensor
flow) (1.8.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\users\shah
_\anaconda3\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0
->tensorflow) (0.6.1)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\shah_\anaconda3\lib\site-p
ackages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (1.0.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\shah_\anaconda3\lib\
site-packages (from tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow)
(2.14.0)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\shah_\anaconda3\lib\site-
packages (from packaging->tensorflow-intel==2.11.0->tensorflow) (2.4.7)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\users\shah_\anaconda3\li
b\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-inte
l==2.11.0->tensorflow) (5.2.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\shah_\anaconda3\lib\
site-packages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel=
=2.11.0->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\shah_\anaconda3\lib\site-pac
kages (from google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0-
>tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\shah_\anaconda3\l
ib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->ten
sorflow-intel==2.11.0->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in c:\users\shah_\anaconda3\li
b\site-packages (from markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow-intel==2.1
1.0->tensorflow) (5.0.0)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\shah_\anaconda3\lib\site
-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.
0->tensorflow) (4.0.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shah_\anaconda3\lib\sit
```

```
e-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-intel==2.1
1.0->tensorflow) (2022.9.14)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\shah_\anaconda3\lib\
site-packages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-intel==
2.11.0->tensorflow) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\shah_\anaconda3\lib\site-pack
ages (from requests<3,>=2.21.0->tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->te
nsorflow) (2.10)
Requirement already satisfied: zipp>=0.5 in c:\users\shah_\anaconda3\lib\site-package
s (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.12,>=2.11->tensorflow
-intel==2.11.0->tensorflow) (3.4.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\shah_\anaconda3\lib\s
ite-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.12,>=
2.11->tensorflow-intel==2.11.0->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\shah_\anaconda3\lib\site-p
ackages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard
<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow) (3.2.2)
Installing collected packages: libclang, flatbuffers, termcolor, tensorflow-io-gcs-fi
lesystem, tensorflow-estimator, opt-einsum, keras, grpcio, google-pasta, gast, astunp
arse, absl-py, markdown, google-auth-oauthlib, tensorboard, tensorflow-intel, tensorf
low
Successfully installed absl-py-1.3.0 astunparse-1.6.3 flatbuffers-22.11.23 gast-0.4.0
google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.51.1 keras-2.11.0 libclang-14.
0.6 markdown-3.4.1 opt-einsum-3.3.0 tensorboard-2.11.0 tensorflow-2.11.0 tensorflow-e
stimator-2.11.0 tensorflow-intel-2.11.0 tensorflow-io-gcs-filesystem-0.28.0 termcolor
```

In [21]:

```
#install keras
#!pip install keras
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
ublic/simple/
Requirement already satisfied: keras in /usr/local/lib/python3.8/dist-packages (2.9.
0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflict
ing behaviour with the system package manager. It is recommended to use a virtual env
ironment instead: https://pip.pypa.io/warnings/venv
```

In [89]:

```
#importing the libraries
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.callbacks import Callback, ModelCheckpoint
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
```

In [109…

```
# Split the dataset into training and testing
X_train, X_test, y_train, y_test = train_test_split(augmented_images, labels, test_s

# Print the number of images in the training and testing sets
print(f"Number of training images: {len(X_train)}")
print(f"Number of testing images: {len(X_test)}")
print(f"Number of testing images: {len(y_train)}")
print(f"Number of testing images: {len(y_test)}")

#y_train = tf.keras.utils.to_categorical(np.array(y_train,dtype="int"),dtype="string"
#y_test = tf.keras.utils.to_categorical(np.array(y_test,dtype="int"),dtype="string")

type(y_train)
```

```
Number of training images: 1632
Number of testing images: 409
Number of testing images: 1632
Number of testing images: 409
```

Out[109…

```
list
```

In [133…

```
# Create an ImageDataGenerator object
datagen =ImageDataGenerator(rescale=1./255,
            shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True,
            validation_split=0.2)

# Create a data generator for the training set
train_generator = datagen.flow_from_directory("C:\\Users\\shah_\\Downloads\\archive\\
                                              target_size=(100, 100),
                                              batch_size=32,
                                                 subset="training")


# Create a data generator for the testing set
val_generator = datagen.flow_from_directory("C:\\Users\\shah_\\Downloads\\archive\\re
                                              target_size=(100, 100),
                                              batch_size=32,
                                                 subset="validation")



# Create a callback to save the best model
checkpoint = ModelCheckpoint("best_model.h5", monitor="val_accuracy", save_best_only=

# Create a callback to stop training when the model stops improving
early_stopping = tf.keras.callbacks.EarlyStopping(monitor="val_accuracy", patience=5)

# Create a callback to reduce the learning rate when the model stops improving
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor="val_accuracy", factor=0.1,

# Create a callback to log the training and testing metrics
csv_logger = tf.keras.callbacks.CSVLogger("training.log")
```

```
Found 1633 images belonging to 2 classes.
Found 408 images belonging to 2 classes.
```

In [135…

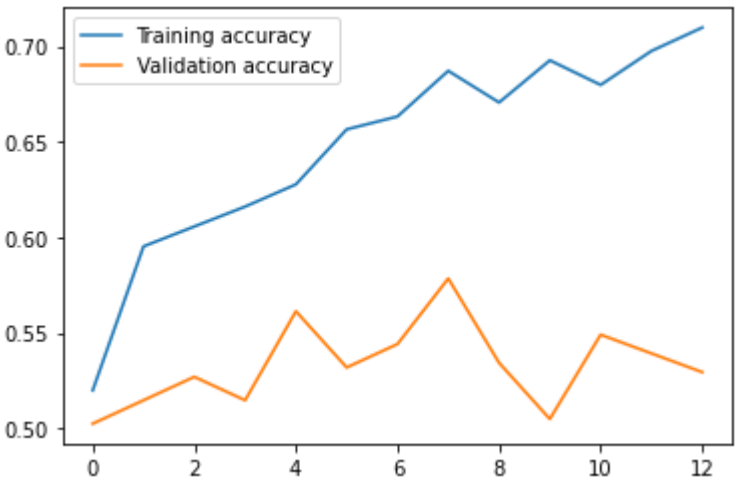
```
#Load the pre-trained model
# Load the pre-trained model
base_model = DenseNet121(include_top=False, weights="imagenet", input_shape=(100, 100

# Freeze the pre-trained model
base_model.trainable = False

# Create a new model on top
model = Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.BatchNormalization(),
    layers.Dropout(0.2),
    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.2),
    layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001), loss="binary_crossentropy", metri

# Fit the model
history = model.fit_generator(train_generator, epochs=25, validation_data=val_generat
```

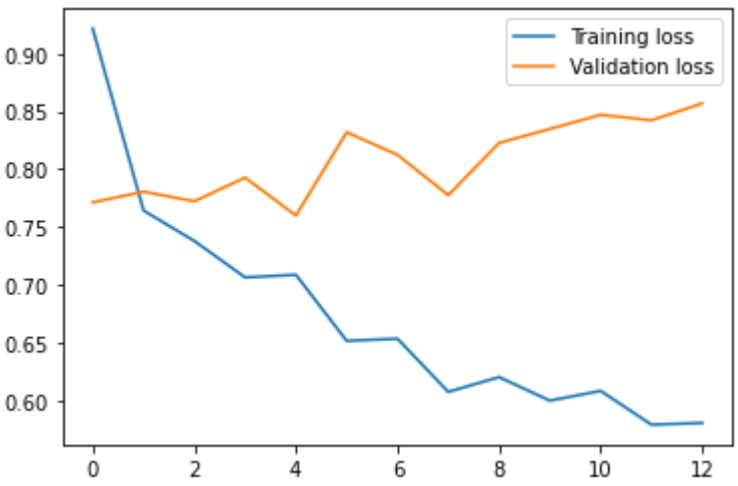
```
Epoch 1/25
52/52 [==============================] - 196s 3s/step - loss: 0.9216 - accuracy: 0.51
99 - val_loss: 0.7711 - val_accuracy: 0.5025 - lr: 1.0000e-04
Epoch 2/25
52/52 [==============================] - 128s 2s/step - loss: 0.7641 - accuracy: 0.59
52 - val_loss: 0.7804 - val_accuracy: 0.5147 - lr: 1.0000e-04
Epoch 3/25
52/52 [==============================] - 123s 2s/step - loss: 0.7378 - accuracy: 0.60
56 - val_loss: 0.7721 - val_accuracy: 0.5270 - lr: 1.0000e-04
Epoch 4/25
52/52 [==============================] - 126s 2s/step - loss: 0.7062 - accuracy: 0.61
60 - val_loss: 0.7926 - val_accuracy: 0.5147 - lr: 1.0000e-04
Epoch 5/25
52/52 [==============================] - 125s 2s/step - loss: 0.7085 - accuracy: 0.62
77 - val_loss: 0.7598 - val_accuracy: 0.5613 - lr: 1.0000e-04
Epoch 6/25
52/52 [==============================] - 122s 2s/step - loss: 0.6513 - accuracy: 0.65
65 - val_loss: 0.8318 - val_accuracy: 0.5319 - lr: 1.0000e-04
Epoch 7/25
52/52 [==============================] - 125s 2s/step - loss: 0.6532 - accuracy: 0.66
32 - val_loss: 0.8123 - val_accuracy: 0.5441 - lr: 1.0000e-04
Epoch 8/25
52/52 [==============================] - 126s 2s/step - loss: 0.6072 - accuracy: 0.68
71 - val_loss: 0.7774 - val_accuracy: 0.5784 - lr: 1.0000e-04
Epoch 9/25
52/52 [==============================] - 126s 2s/step - loss: 0.6199 - accuracy: 0.67
05 - val_loss: 0.8226 - val_accuracy: 0.5343 - lr: 1.0000e-04
Epoch 10/25
52/52 [==============================] - 136s 3s/step - loss: 0.5996 - accuracy: 0.69
26 - val_loss: 0.8347 - val_accuracy: 0.5049 - lr: 1.0000e-04
Epoch 11/25
52/52 [==============================] - 130s 2s/step - loss: 0.6080 - accuracy: 0.67
97 - val_loss: 0.8469 - val_accuracy: 0.5490 - lr: 1.0000e-04
Epoch 12/25
```

```
52/52 [==============================] - 123s 2s/step - loss: 0.5788 - accuracy: 0.69
75 - val_loss: 0.8422 - val_accuracy: 0.5392 - lr: 1.0000e-05
Epoch 13/25
52/52 [==============================] - 119s 2s/step - loss: 0.5802 - accuracy: 0.70
```

In [136…

```
# Plot the training and validation accuracy
plt.plot(history.history["accuracy"], label="Training accuracy")
plt.plot(history.history["val_accuracy"], label="Validation accuracy")
plt.legend()
plt.show()

# Plot the training and validation loss
plt.plot(history.history["loss"], label="Training loss")
plt.plot(history.history["val_loss"], label="Validation loss")
plt.legend()
plt.show()
```





In [137…

```
#Load the best model
# Load the best model
model = tf.keras.models.load_model("best_model.h5")

# Evaluate the model on the test set
val_loss, val_acc = model.evaluate(val_generator)

# Print the test accuracy
print(f"val accuracy: {val_acc:.3f}")
```

```
13/13 [==============================] - 33s 2s/step - loss: 0.7936 - accuracy: 0.546
6
val accuracy: 0.547
```

In [139…

```
#Load the test images
# Load the test images
#test_image = cv2.imread("test_image.jpg")
#plt.imshow(test_image)

#resized_test_image = cv2.resize(test_image, (100,100))
# Reshape the test images
#reshaped_test_image = resized_test_image.reshape(100,100,1) # [image.reshape(100, 1

# Create a data generator for the test images
test_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True)


test_generator = test_datagen.flow_from_directory("C:\\Users\\shah_\\Downloads\\test'
                                          target_size=(100, 100),
                                          batch_size=32)

# Make predictions on the test images
predictions = model.predict(test_generator)


#rounded_predictions = model.predict_classes(x = X_test, batch_size=10, verbose=0)

#for i in rounded_predictions[:10]:
#    print(i)
# Print the first 10 predictions
#print(predictions[:10])

# Print the first 10 predictions rounded to the nearest integer
#print(np.round(predictions[:10]))

# Print the number of fake and real images in the test set
print(f"Number of fake images: {np.sum(np.round(predictions))}")

# Print the number of fake and real images in the test set
print(f"Number of real images: {len(predictions) - np.sum(np.round(predictions))}")
```

```
Found 59 images belonging to 2 classes.
2/2 [==============================] - 5s 2s/step
Number of fake images: 21.0
Number of real images: 38.0
```

In [144…

```
#!pip install Dash
#! pip install dash-html-components
#! pip install dash-core-components
#! pip install plotly
```

```
Requirement already satisfied: dash-html-components in c:\users\shah_\anaconda3\lib\s
ite-packages (2.0.0)
Requirement already satisfied: dash-core-components in c:\users\shah_\anaconda3\lib\s
ite-packages (2.0.0)
Requirement already satisfied: plotly in c:\users\shah_\anaconda3\lib\site-packages
(5.11.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\shah_\anaconda3\lib\site-p
ackages (from plotly) (8.1.0)
```

In [155…

```
#build a DASH app to upload images and predict
# Import the necessary libraries
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import plotly.graph_objects as go
import plotly.express as px
import plotly.io as pio
import plotly.figure_factory as ff
import pandas as pd
import numpy as np
import cv2
import os
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.optimizers import Adam
import base64
# Create a Dash application
app = dash.Dash('__Fakebook__')

# Create the app layout
app.layout = html.Div([
    html.H1("Face Detection App", style={"textAlign": "center"}),
    html.Div([
        html.Div([
            dcc.Upload(
                id="upload-image",
                children=html.Div([
                    "Drag and drop or click to select an image to upload."
                ]),
                style={
                    "width": "100%",
                    "height": "60px",
                    "lineHeight": "60px",
                    "borderWidth": "1px",
                    "borderStyle": "dashed",
                    "borderRadius": "5px",
                    "textAlign": "center",
                    "margin": "10px"},
                multiple=False
            ),
            html.Div(id="output-image-upload"),
        ], className="six columns"),
        html.Div([
            html.H3("Prediction"),
            html.Div(id="output-image-upload-prediction")
        ], className="six columns"),
    ], className="row")
])

# Define a callback function to display the image
@app.callback(Output("output-image-upload", "children"),
                [Input("upload-image", "contents")])
def update_output(list_of_contents):
```

```
    if list_of_contents is not None:
        children = [
            html.H5("Uploaded Image"),
            # Decode the image
            html.Img(src=list_of_contents, style={"width": "60%", "height":"60%"})
        ]
        return children

# Define a callback function to display the prediction of the image
@app.callback(Output("output-image-upload-prediction", "children"),
                [Input("upload-image", "contents")])
def update_output(list_of_contents):
    if list_of_contents is not None:
        # Decode the image
        decoded = base64.b64decode(list_of_contents.split(",")[1])
        # Convert the image to a numpy array
        image = np.asarray(bytearray(decoded), dtype="uint8")
        # Read the image
        image = cv2.imdecode(image, cv2.IMREAD_COLOR)
        # Resize the image
        resized_image = cv2.resize(image, (100,100))
        # Reshape the image
        reshaped_image = resized_image.reshape(1,100,100,3)
        # Make a prediction
        prediction = model.predict(reshaped_image)
        # Get the class of the prediction
        class_ = np.argmax(prediction, axis=1)
        if(class_[0] == 0):
            return f"The Prediction is fake"
        else:
            return f"The Prediction is real"
        # Return the prediction
        #return f"The prediction is: {class_[0]}"

# Run the app
if __name__ == "__main__":
    app.run_server(debug=True,use_reloader=False)
```

```
Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/

Dash is running on http://127.0.0.1:8050/
```

```
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app "__Fakebook__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
1/1 [==============================] - 0s 392ms/step
```

In [ ]: