In [36]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from keras.models import Sequential
from keras.layers import Dense, Flatten,Conv2D
from keras.datasets import mnist
import tensorflow as tf
```

In [18]:

```
import os
!pip install opencv-python
import cv2
```

```
Collecting opencv-python
  Downloading opencv_python-4.8.0.76-cp37-abi3-win_amd64.whl (38.1 MB)
     ---------------------------------------- 38.1/38.1 MB 96.5 kB/s eta 0:
00:00
Requirement already satisfied: numpy>=1.21.2 in c:\users\shahzad\anaconda3
\lib\new folder\lib\site-packages (from opencv-python) (1.23.5)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.8.0.76
```

In [8]:

```
!pip install tensorflow
!pip install keras
```

```
Collecting tensorflow
  Using cached tensorflow-2.13.0-cp310-cp310-win_amd64.whl (1.9 kB)
Collecting tensorflow-intel==2.13.0
  Downloading tensorflow_intel-2.13.0-cp310-cp310-win_amd64.whl (276.5 M
B)
     ---------------                        117.6/276.5 MB 41.0 kB/s eta
1:04:41

ERROR: Exception:
Traceback (most recent call last):
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
\_vendor\urllib3\response.py", line 437, in _error_catcher
    yield
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
\_vendor\urllib3\response.py", line 560, in read
    data = self._fp_read(amt) if not fp_closed else b""
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
\_vendor\urllib3\response.py", line 526, in _fp_read
    return self._fp.read(amt) if amt is not None else self._fp.read()
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
```

In [13]:

```
folder_path = ['C:\\Users\\shahzad\\Desktop\\train\\female','C:\\Users\\shahzad\\Desktop\
```

In [20]:

```
dataset=[]
```

In [28]:

```
# Iterate over the folder paths
for i in folder_path:
    folder_name = os.path.basename(i)

 # Iterate over the images in the subdirectory
    for file_name in os.listdir(i):
        image_path = os.path.join(i, file_name)
        if os.path.isfile(image_path): # Only consider files
            # Load the image using OpenCV

            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
            # If the image was successfully loaded
            if image is not None:
                # Resize the grayscale image to 100X100 pixels
                resized_image = cv2.resize(image, (100, 100))
                # Flatten the image and append each pixel as a separate feature along wit
                flattened_image = resized_image.flatten().tolist()
                label = 'female' if 'female' in file_name else 'male'  # Adjust label ass
            dataset.append(flattened_image + [label])
        else:
            print(f"Error loading image: {image_path}")

image_size = 100
num_pixels = image_size * image_size
column_names = [f'pixel_{i+1}' for i in range(num_pixels)] + ['label']
df = pd.DataFrame(dataset, columns=column_names)

print(df.head())
```

```
   pixel_1  pixel_2  pixel_3  pixel_4  pixel_5  pixel_6  pixel_7  pixel_8
\
0       42       37       32       32       35       29       22       17
1       53       57       55       51       58       64       64       64
2       12       11        9        8        9       13       15       17
3      140      148      150      159      178      191      188      174
4       32       20       12       11        1        8       18       20

   pixel_9  pixel_10  ...  pixel_9992  pixel_9993  pixel_9994  pixel_9995
\
0       17        20  ...           2           1           0           1
1       68        67  ...         109         109         108         106
2       19        20  ...          19          23          30          46
3      166       172  ...          39          28          25          25
4       17        26  ...         108         103         112         108

   pixel_9996  pixel_9997  pixel_9998  pixel_9999  pixel_10000   label
0           7          15          21          21           17  female
1         103          99          94          89           85  female
2          59          48          47          48           41  female
3          29          36          39          38           35  female
4          90          94         104          96          101  female

[5 rows x 10001 columns]
```

In [45]:

```
df=pd.DataFrame(dataset)
df
```

Out[45]:

	0	1	2	3	4	5	6	7	8	9	...	9991	9992	9993	9994	9995
0	42	37	32	32	35	29	22	17	17	20	...	2	1	0	1	7
1	53	57	55	51	58	64	64	64	68	67	...	109	109	108	106	103
2	12	11	9	8	9	13	15	17	19	20	...	19	23	30	46	59
3	140	148	150	159	178	191	188	174	166	172	...	39	28	25	25	29
4	32	20	12	11	1	8	18	20	17	26	...	108	103	112	108	90
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6977	34	30	33	42	52	57	42	36	36	45	...	230	235	229	220	208
6978	56	58	52	49	49	51	53	58	65	70	...	153	186	211	222	228
6979	51	38	46	60	66	73	78	84	88	91	...	85	78	69	65	71
6980	204	148	86	46	35	39	39	38	49	57	...	134	137	140	140	139
6981	56	52	47	43	43	46	49	53	56	55	...	20	17	12	28	39

6982 rows × 10001 columns

In [61]:

```
images = []
labels = []
```

In [73]:

```
images=np.array(images)
labels=np.array(labels)
```

In [76]:

```
images=images/255
```

In [29]:

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['label_encoded'] = label_encoder.fit_transform(df['label'])
```

In [ ]:

```

```

In [30]:

```
label = 'female' if 'female' in file_name.lower() else 'male'
```

In [31]:

```
unique_labels = df['label'].unique()
print(unique_labels)
```

```
['female' 'male']
```

In [33]:

```
!pip install tensorflow
```

```
\_vendor\resolvelib\resolvers.py", line 481, in resolve
    state = resolution.resolve(requirements, max_rounds=max_rounds)
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
\_vendor\resolvelib\resolvers.py", line 373, in resolve
    failure_causes = self._attempt_to_pin_criterion(name)
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
\_vendor\resolvelib\resolvers.py", line 213, in _attempt_to_pin_criterio
n
    criteria = self._get_updated_criteria(candidate)
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
\_vendor\resolvelib\resolvers.py", line 204, in _get_updated_criteria
    self._add_to_criteria(criteria, requirement, parent=candidate)
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
\_vendor\resolvelib\resolvers.py", line 172, in _add_to_criteria
    if not criterion.candidates:
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
\_vendor\resolvelib\structs.py", line 151, in __bool__
    return bool(self._sequence)
  File "C:\Users\shahzad\anaconda3\Lib\New folder\lib\site-packages\pip
```

In [38]:

```
label_counts = df['label'].value_counts()
print(label_counts)
```

```
female    3491
male      3491
Name: label, dtype: int64
```

In [81]:

```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(df.drop(columns=[0]),df[0],test_size=


# Print the shapes of the resulting sets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape
```

```
X_train shape: (5585, 10000)
X_test shape: (1397, 10000)
y_train shape: (5585,)
y_test shape: (1397,)
```

In [ ]:

```

```

In [ ]:

```

```

In [37]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Create a sequential model
model = Sequential()

# Add convolutional layers
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten the output from convolutional layers
model.add(Flatten())

# Add dense layers
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))  # Output layer for binary classification (0 fo

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Print model summary
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2  (None, 63, 63, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 61, 61, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 30, 30, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 128)       0
 g2D)

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 128)               3211392

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 3312961 (12.64 MB)
Trainable params: 3312961 (12.64 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [94]:

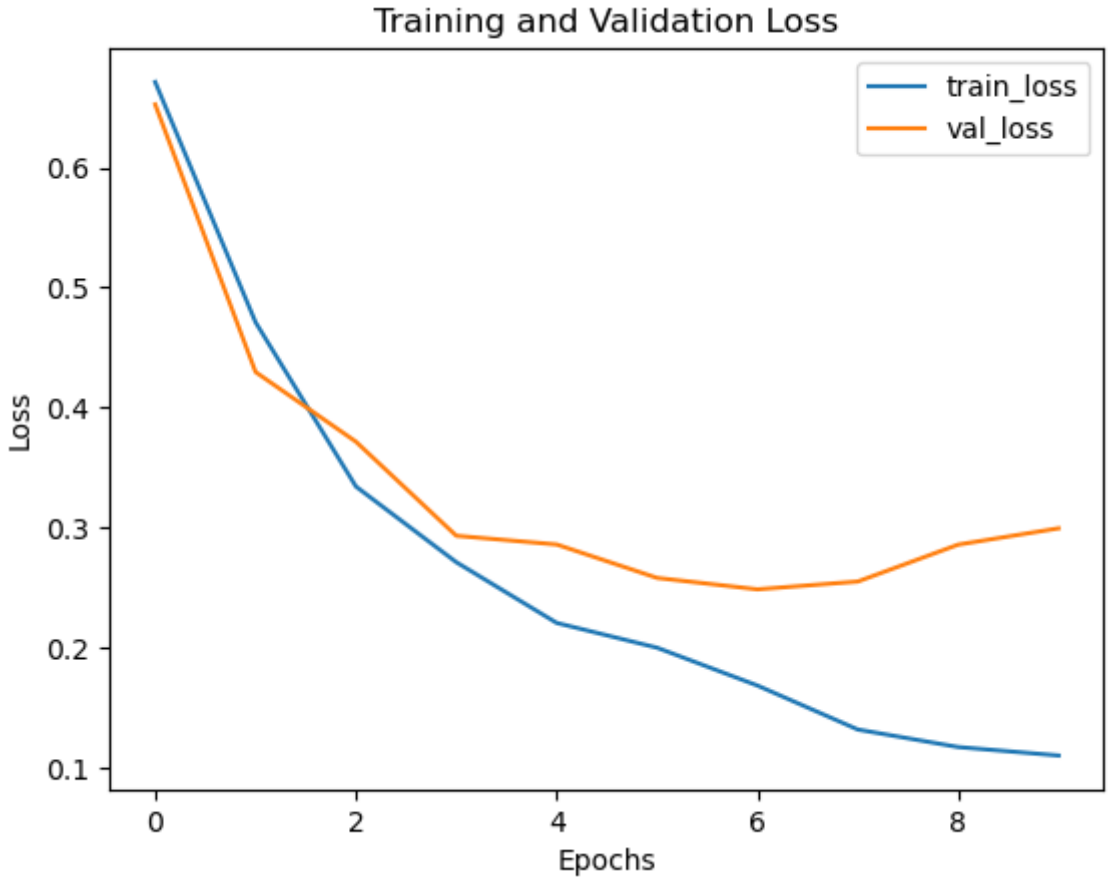
```
from sklearn.preprocessing import LabelEncoder

# Initialize label encoder
label_encoder = LabelEncoder()

# Encode labels
y_train_encoded = label_encoder.fit_transform(y_train)
```

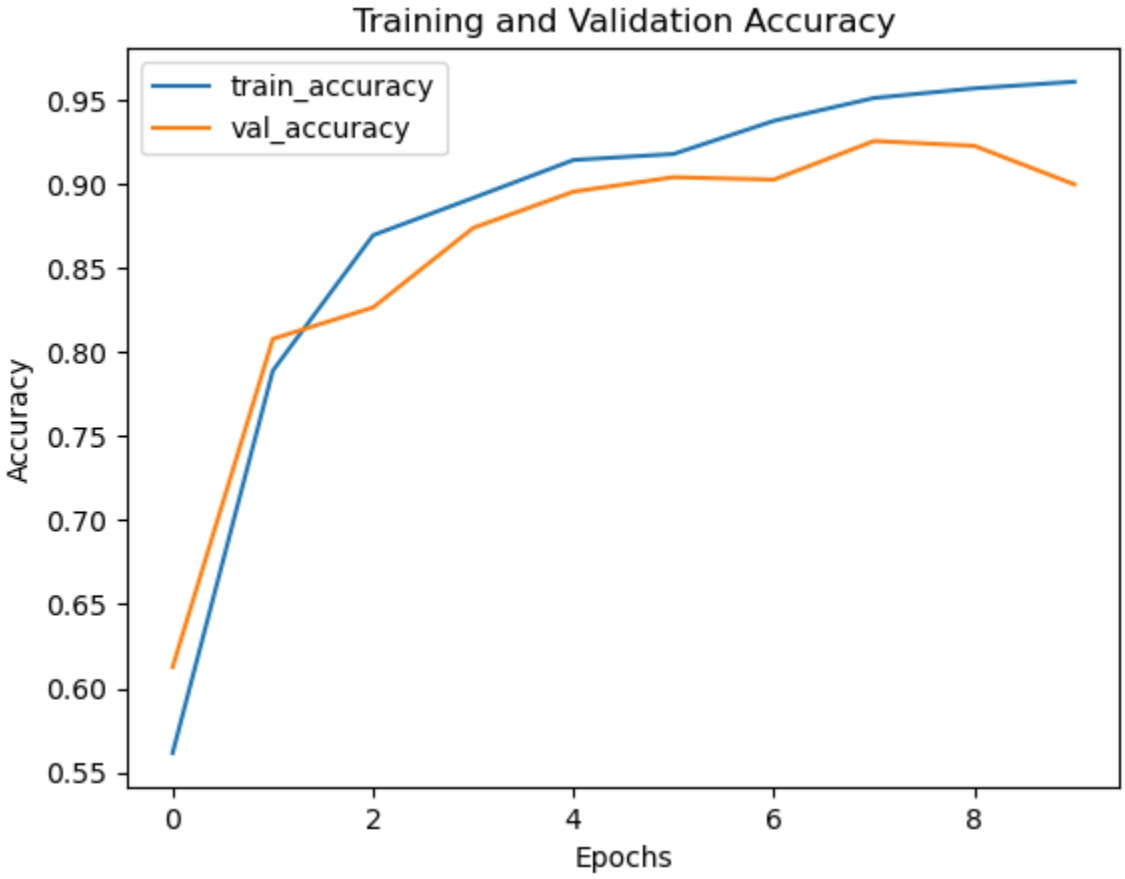
In [106]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

# Path to the directory containing male and female subdirectories
data_dir = 'C:\\Users\\shahzad\\Desktop\\train'

# Define data preprocessing and augmentation
image_size = (128, 128)
batch_size = 32

datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    validation_split=0.2
)

train_generator = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='validation'
)

# Train the model
epochs = 10
history = model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=len(validation_generator)
)

# Evaluate the model
loss, accuracy = model.evaluate(validation_generator)
print(f"Validation Loss: {loss:.4f}")
print(f"Validation Accuracy: {accuracy:.4f}")

# Plot training history
plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.show()

plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.legend()
plt.xlabel('Epochs')
```

```
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.show()
```

```
Found 2794 images belonging to 2 classes.
Found 697 images belonging to 2 classes.
Epoch 1/10
88/88 [==============================] - 219s 2s/step - loss: 0.6712 - acc
uracy: 0.5616 - val_loss: 0.6525 - val_accuracy: 0.6126
Epoch 2/10
88/88 [==============================] - 201s 2s/step - loss: 0.4711 - acc
uracy: 0.7888 - val_loss: 0.4296 - val_accuracy: 0.8077
Epoch 3/10
88/88 [==============================] - 240s 3s/step - loss: 0.3342 - acc
uracy: 0.8694 - val_loss: 0.3715 - val_accuracy: 0.8264
Epoch 4/10
88/88 [==============================] - 200s 2s/step - loss: 0.2711 - acc
uracy: 0.8916 - val_loss: 0.2932 - val_accuracy: 0.8737
Epoch 5/10
88/88 [==============================] - 226s 3s/step - loss: 0.2205 - acc
uracy: 0.9141 - val_loss: 0.2859 - val_accuracy: 0.8953
Epoch 6/10
88/88 [==============================] - 210s 2s/step - loss: 0.1999 - acc
uracy: 0.9177 - val_loss: 0.2580 - val_accuracy: 0.9039
Epoch 7/10
88/88 [==============================] - 201s 2s/step - loss: 0.1683 - acc
uracy: 0.9374 - val_loss: 0.2485 - val_accuracy: 0.9024
Epoch 8/10
88/88 [==============================] - 209s 2s/step - loss: 0.1317 - acc
uracy: 0.9510 - val_loss: 0.2550 - val_accuracy: 0.9254
Epoch 9/10
88/88 [==============================] - 209s 2s/step - loss: 0.1171 - acc
uracy: 0.9567 - val_loss: 0.2858 - val_accuracy: 0.9225
Epoch 10/10
88/88 [==============================] - 204s 2s/step - loss: 0.1100 - acc
uracy: 0.9606 - val_loss: 0.2994 - val_accuracy: 0.8996
22/22 [==============================] - 15s 690ms/step - loss: 0.2994 - a
ccuracy: 0.8996
Validation Loss: 0.2994
Validation Accuracy: 0.8996
```


Training and Validation Loss
train_loss
val_loss
Loss
Epochs
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8


Training and Validation Accuracy
train_accuracy
val_accuracy
Accuracy
Epochs
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
2
4
6
8

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: