In [ ]:

In [2]:

```
import numpy as np
import matplotlib .pyplot as plt
import matplotlib .pyplot as op
import os
import pydot
import graphviz
import datetime as dt
import random
import cv2
from pathlib import Path
import glob
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedShuffleSplit,train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_a
from tensorflow.keras import layers
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.mobilenet import MobileNet
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.models import Model
from tensorflow.keras.utils import plot_model
from sklearn.metrics import confusion_matrix, classification_report, recall_s
from tensorflow.keras.preprocessing import image
from keras.models import Sequential
from keras.layers import GlobalAveragePooling2D,Dense,Dropout


for dirname, _, filenames in os.walk('E:\data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
E:\data\non-vehicles\extra1000.png
E:\data\non-vehicles\extra1001.png
E:\data\non-vehicles\extra1002.png
E:\data\non-vehicles\extra1003.png
E:\data\non-vehicles\extra1004.png
E:\data\non-vehicles\extra1005.png
E:\data\non-vehicles\extra1006.png
E:\data\non-vehicles\extra1007.png
E:\data\non-vehicles\extra1008.png
E:\data\non-vehicles\extra1009.png
E:\data\non-vehicles\extra101.png
E:\data\non-vehicles\extra1010.png
E:\data\non-vehicles\extra1011.png
E:\data\non-vehicles\extra1012.png
E:\data\non-vehicles\extra1013.png
E:\data\non-vehicles\extra1014.png
E:\data\non-vehicles\extra1015.png
E:\data\non-vehicles\extra102.png
E:\data\non-vehicles\extra103.png
E:\data\non-vehicles\extra1032.png
```

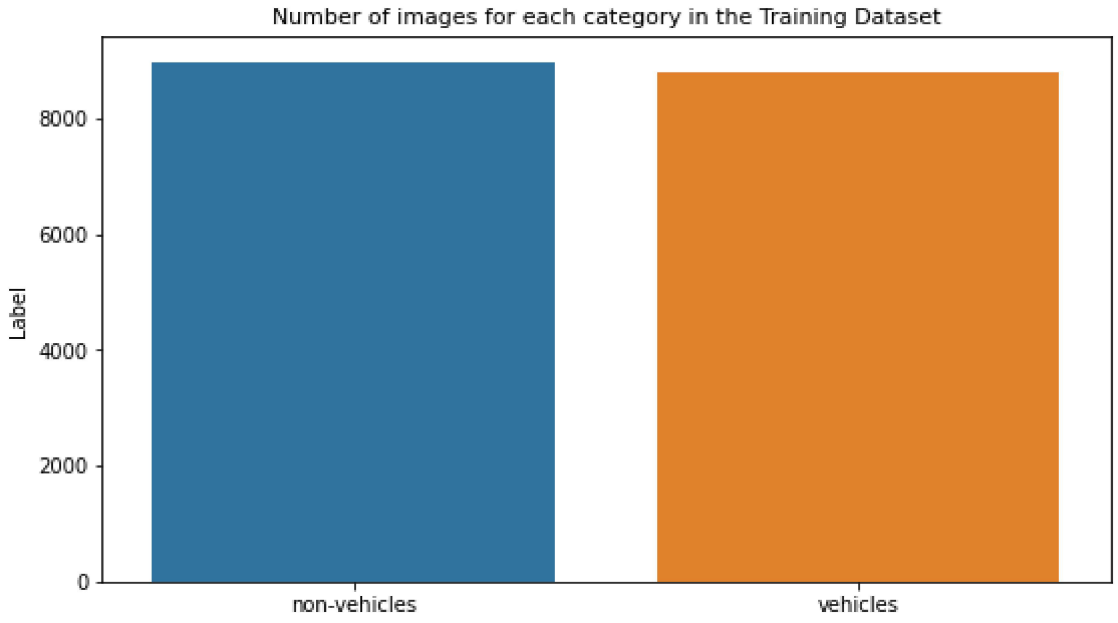
In [3]:
```
maindir = "E:\data"
os.listdir(maindir)
```

Out[3]:
```
['non-vehicles', 'vehicles']
```

In [4]:
```
vehicle_dir = "E/data/vehicles"
nonvehicle_dir = "E/data/non-vehicles"
vehicle = os.listdir(maindir+"/vehicles")
non_vehicle = os.listdir(maindir+"/non-vehicles")

print(f"Number of Vehicle Images: {len(vehicle)}")
print(f"Number of Non Vehicle Images: {len(non_vehicle)}")
```

```
Number of Vehicle Images: 8792
Number of Non Vehicle Images: 8968
```

In [5]:
```
# Selecting Dataset Folder Paths
dir_ = Path('E:\data')
filepaths = list(dir_.glob(r'**/*.png'))
# Mapping the Labels
labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths)

# Paths & Labels femalee eyes
filepaths = pd.Series(filepaths, name = 'File').astype(str)
labels = pd.Series(labels, name='Label')

# Concatenating...
df = pd.concat([filepaths, labels], axis=1)

df = df.sample(frac = 1, random_state = 56).reset_index(drop = True)

vc = df['Label'].value_counts()
plt.figure(figsize = (9, 5))
sns.barplot(x = vc.index, y = vc)
plt.title("Number of images for each category in the Training Dataset", fonts
plt.show()
```

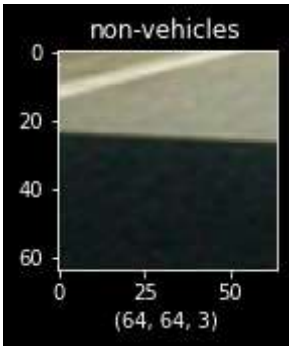


In [6]:
```
plt.style.use("dark_background")
```

In [ ]:
```

```

In [8]:
```
figure = plt.figure(figsize=(2,2))
x = plt.imread(df["File"][34])
plt.imshow(x)
plt.xlabel(x.shape)
plt.title(df["Label"][34])
```

Out[8]:
```
Text(0.5, 1.0, 'non-vehicles')
```



In [9]:
```
fig, axes = plt.subplots(nrows = 6,
                         ncols = 6,
                         figsize = (10, 10),
                         subplot_kw = {"xticks":[],"yticks":[]})

for i,ax in enumerate(axes.flat):
    ax.imshow(plt.imread(df["File"][i]))
    ax.set_title(df["Label"][i])
plt.tight_layout()
plt.show()
```

In [10]:
```
trainset_df, testset_df = train_test_split(df, train_size = 0.90, random_stat

display(trainset_df.head())

testset_df.head()
```

	File	Label
5758	E:\data\non-vehicles\extra3217.png	non-vehicles
11522	E:\data\vehicles\3533.png	vehicles
3143	E:\data\vehicles\left (229).png	vehicles
8774	E:\data\non-vehicles\image801.png	non-vehicles
9845	E:\data\vehicles\1274.png	vehicles

Out[10]:

	File	Label
15314	E:\data\non-vehicles\image3846.png	non-vehicles
11728	E:\data\non-vehicles\image562.png	non-vehicles
10379	E:\data\vehicles\2287.png	vehicles
6323	E:\data\vehicles\3759.png	vehicles
5155	E:\data\vehicles\4221.png	vehicles

In [11]:
```
LE = LabelEncoder()

y_test = LE.fit_transform(testset_df["Label"])
```

In [12]:
```
print('Training Dataset:')

print(f'Number of images: {trainset_df.shape[0]}')

print(f'Number of images with vehicle: {trainset_df["Label"].value_counts()[0
print(f'Number of images with non vehicle: {trainset_df["Label"].value_counts

# Viewing data in test dataset
print('Test Dataset:')

print(f'Number of images: {testset_df.shape[0]}')

print(f'Number of images with vehicle: {testset_df["Label"].value_counts()[0]
print(f'Number of images with non vehicle: {testset_df["Label"].value_counts(
```

```
Training Dataset:
Number of images: 15984
Number of images with vehicle: 8076
Number of images with non vehicle: 7908

Test Dataset:
Number of images: 1776
Number of images with vehicle: 892
Number of images with non vehicle: 884
```

In [13]:
```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.1,
                                   rotation_range = 20,
                                   width_shift_range = 0.1,
                                   height_shift_range = 0.1,
                                   horizontal_flip = True,
                                   vertical_flip = True,
                                   validation_split = 0.1)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

In [14]:
```
print("Preparing the training dataset ...")
training_set = train_datagen.flow_from_dataframe(
    dataframe = trainset_df,
    x_col = "File",
    y_col = "Label",
    target_size = (75, 75),
    color_mode = "rgb",
    class_mode = "binary",
    batch_size = 32,
    shuffle = True,
    seed = 2,
    subset = "training")
print("Preparing the validation dataset ...")
validation_set = train_datagen.flow_from_dataframe(
    dataframe = trainset_df,
    x_col = "File",
    y_col = "Label",
    target_size = (75, 75),
    color_mode ="rgb",
    class_mode = "binary",
    batch_size = 32,
    shuffle = True,
    seed = 2,
    subset = "validation")
print("Preparing the test dataset ...")
test_set = test_datagen.flow_from_dataframe(
    dataframe = testset_df,
    x_col = "File",
    y_col = "Label",
    target_size = (75, 75),
    color_mode ="rgb",
    class_mode = "binary",
    shuffle = False,
    batch_size = 32)

print('Data generators are ready!')
```

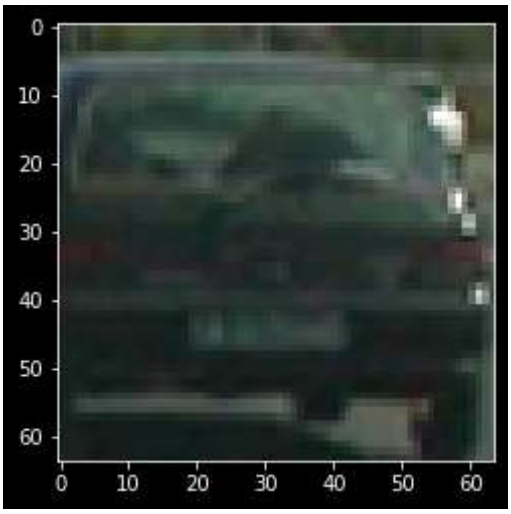
```
Preparing the training dataset ...
Found 14386 validated image filenames belonging to 2 classes.
Preparing the validation dataset ...
Found 1598 validated image filenames belonging to 2 classes.
Preparing the test dataset ...
Found 1776 validated image filenames belonging to 2 classes.
Data generators are ready!
```

In [15]:

```
from PIL import Image, ImageEnhance
import random
file_path_type = ["E:/data/vehicles/*.png"]
images = glob.glob(random.choice(file_path_type))
random_image = random.choice(images)
imgo = Image.open(random_image)
print(imgo.format)
print(imgo.mode)
print(imgo.size)

#show the image
plt.imshow(imgo)
```

```
PNG
RGB
(64, 64)
```

Out[15]: <matplotlib.image.AxesImage at 0x22b425a4640>



In [16]:

```
#CNN Model
#INCEPTION


#1. Base model creation

CNN_base_inc = InceptionV3(input_shape = (75, 75, 3), include_top = False, we

for layer in CNN_base_inc.layers:
    layer.trainable = False

#2. Flattening

x = layers.Flatten()(CNN_base_inc.output)
```

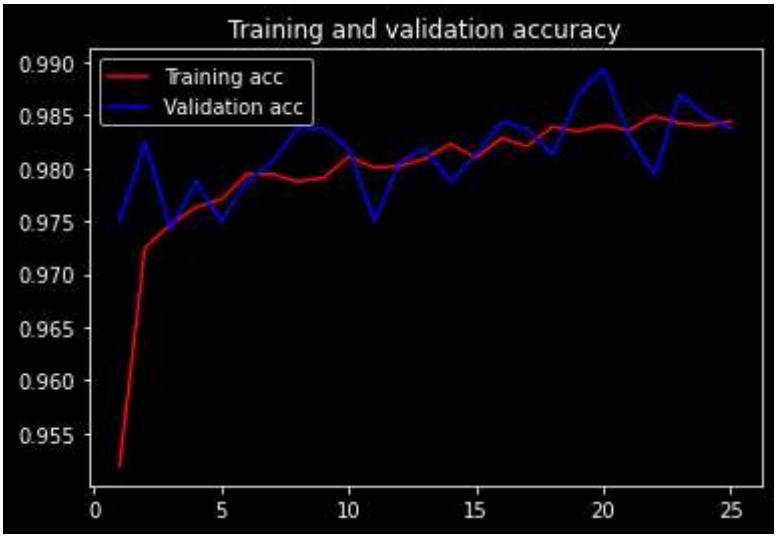
In [17]:

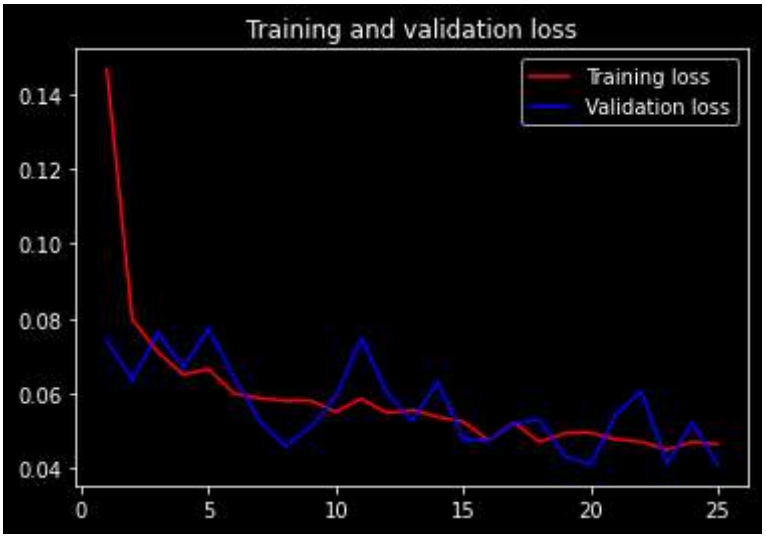
```
#3. Dense Neural Network

x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.1)(x)
x = layers.Dense(1, activation='sigmoid')(x)
cb=0;
CNN_inc = Model(CNN_base_inc.input, x)

#4. Model Compilation & Training

CNN_inc.compile(optimizer = RMSprop(lr = 0.0001), loss = 'binary_crossentropy

# Start of counting time
start = dt.datetime.now()

# Training and validation
CNN_inc_history = CNN_inc.fit(training_set, epochs = 25, validation_data = va

# End of Time Counting
end = dt.datetime.now()
time_CNN_inc = end - start
print ('\nTraining and validation time is: ', time_CNN_inc)
```

```
C:\Users\hp\anaconda3\envs\python1\lib\site-packages\keras\optimizers\opt
imizer_v2\rmsprop.py:135: UserWarning: The `lr` argument is deprecated, u
se `learning_rate` instead.
  super(RMSprop, self).__init__(name, **kwargs)
```

In [18]:

```
# 5.Model training history

acc = CNN_inc_history.history['accuracy']
val_acc = CNN_inc_history.history['val_accuracy']
loss = CNN_inc_history.history['loss']
val_loss = CNN_inc_history.history['val_loss']
epochs = range(1, len(acc) + 1)

plt.title('Training and validation accuracy')
plt.plot(epochs, acc, 'red', label='Training acc')
plt.plot(epochs, val_acc, 'blue', label='Validation acc')
plt.legend()

plt.figure()
plt.title('Training and validation loss')
plt.plot(epochs, loss, 'red', label='Training loss')
plt.plot(epochs, val_loss, 'blue', label='Validation loss')

plt.legend()

plt.show()
```





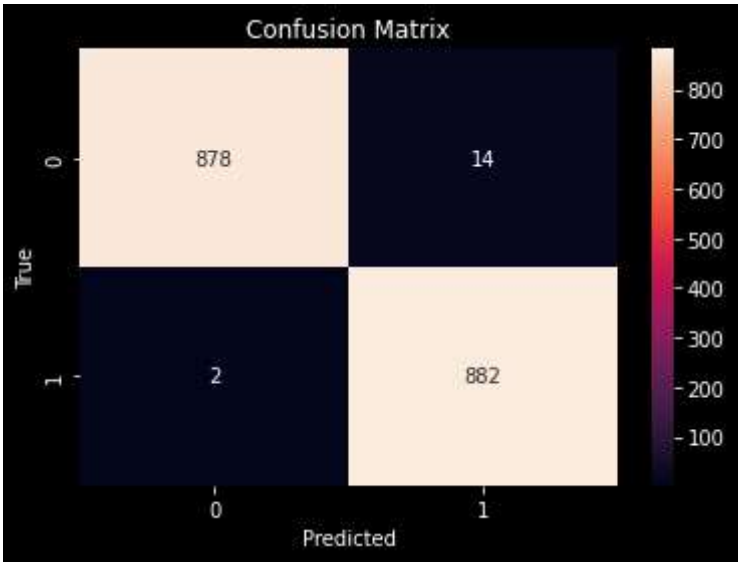
In [19]:
```
# 6.Viewing results and generating forecasts

score_inc = CNN_inc.evaluate(test_set)
print("Test Loss:", score_inc[0])
print("Test Accuracy:", score_inc[1])
```

```
56/56 [==============================] - 16s 279ms/step - loss: 0.0338 - ac
curacy: 0.9910
Test Loss: 0.0337829627096653
Test Accuracy: 0.9909909963607788
```

In [21]:
```
y_pred_inc = CNN_inc.predict(test_set)
y_pred_inc = np.round(y_pred_inc)

recall_inc = recall_score(y_test, y_pred_inc)
precision_inc = precision_score(y_test, y_pred_inc)
f1_inc = f1_score(y_test, y_pred_inc)
roc_inc = roc_auc_score(y_test, y_pred_inc)

print(classification_report(y_test, y_pred_inc))
```

```
56/56 [==============================] - 14s 246ms/step
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       892
           1       0.98      1.00      0.99       884

    accuracy                           0.99      1776
   macro avg       0.99      0.99      0.99      1776
weighted avg       0.99      0.99      0.99      1776
```

In [22]:
```
plt.figure(figsize = (6, 4))

sns.heatmap(confusion_matrix(y_test, y_pred_inc),annot = True, fmt = 'd')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")

plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [23]:
```
CNN_base_mobilenet = MobileNet(input_shape = (75, 75, 3), include_top = False
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is n
ot in [128, 160, 192, 224]. Weights for input shape (224, 224) will be load
ed as the default.
```

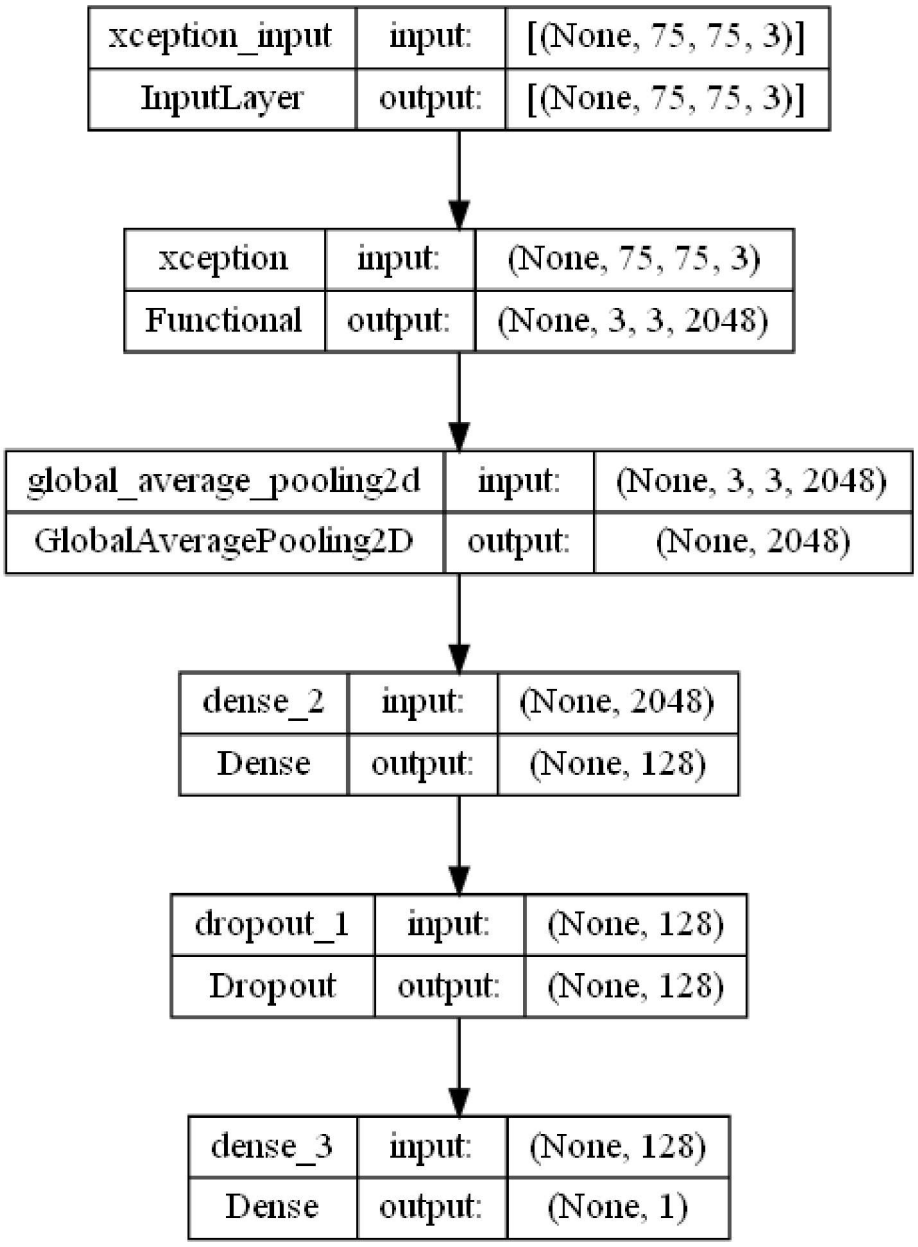
In [24]:
```
#XCEPTIONN
# 1.Base model creation
CNN_base_xcep = Xception(input_shape = (75, 75, 3), include_top = False, weig
CNN_base_xcep.trainable = False
```

In [25]:
```
#2.Dense Neural Networks
CNN_xcep = Sequential()
CNN_xcep.add(CNN_base_xcep)
CNN_xcep.add(GlobalAveragePooling2D())
CNN_xcep.add(Dense(128))
CNN_xcep.add(Dropout(0.1))
CNN_xcep.add(Dense(1, activation = 'sigmoid'))

CNN_xcep.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 3, 3, 2048)        20861480

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense_2 (Dense)             (None, 128)               262272

 dropout_1 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 1)                 129

=================================================================
Total params: 21,123,881
Trainable params: 262,401
Non-trainable params: 20,861,480
_________________________________________________________________
```

In [26]:
```
plot_model(CNN_xcep, show_layer_names = True , show_shapes = True)
```

Out[26]:



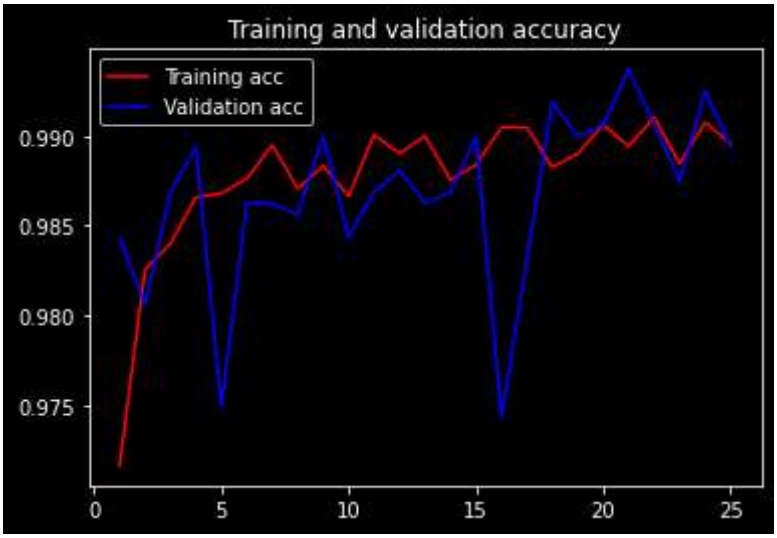
In [27]:
```
# 4. Model compilation and training
# Compilation
CNN_xcep.compile(optimizer='adam', loss = 'binary_crossentropy',metrics=['acc

# Start of counting time
start = dt.datetime.now()

# Training and validation
CNN_xcep_history = CNN_xcep.fit(training_set, epochs = 25, validation_data =

# End of Time Counting
end = dt.datetime.now()
time_CNN_xcep = end - start
print ('\nTraining and validation time: ', time_CNN_xcep)
```

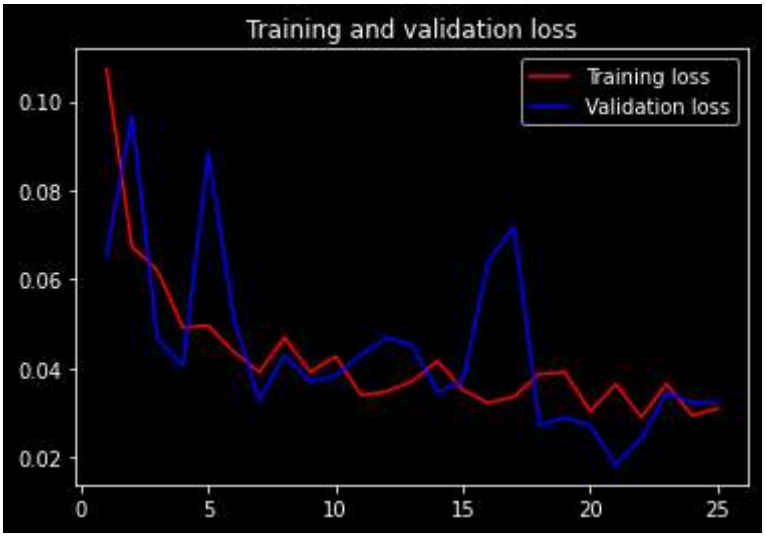
```
Epoch 1/25
450/450 [==============================] - 350s 767ms/step - loss: 0.1073 -
accuracy: 0.9716 - val_loss: 0.0654 - val_accuracy: 0.9844
Epoch 2/25
450/450 [==============================] - 375s 833ms/step - loss: 0.0673 -
accuracy: 0.9826 - val_loss: 0.0967 - val_accuracy: 0.9806
Epoch 3/25
450/450 [==============================] - 376s 836ms/step - loss: 0.0620 -
accuracy: 0.9840 - val_loss: 0.0466 - val_accuracy: 0.9869
Epoch 4/25
450/450 [==============================] - 351s 779ms/step - loss: 0.0491 -
accuracy: 0.9866 - val_loss: 0.0406 - val_accuracy: 0.9894
Epoch 5/25
450/450 [==============================] - 340s 756ms/step - loss: 0.0496 -
accuracy: 0.9868 - val_loss: 0.0884 - val_accuracy: 0.9750
Epoch 6/25
450/450 [==============================] - 339s 753ms/step - loss: 0.0436 -
accuracy: 0.9876 - val_loss: 0.0513 - val_accuracy: 0.9862
Epoch 7/25
450/450 [==============================] - 336s 747ms/step - loss: 0.0392 -
accuracy: 0.9895 - val_loss: 0.0328 - val_accuracy: 0.9862
Epoch 8/25
450/450 [==============================] - 335s 744ms/step - loss: 0.0469 -
accuracy: 0.9871 - val_loss: 0.0429 - val_accuracy: 0.9856
Epoch 9/25
450/450 [==============================] - 336s 748ms/step - loss: 0.0391 -
accuracy: 0.9883 - val_loss: 0.0370 - val_accuracy: 0.9900
Epoch 10/25
450/450 [==============================] - 336s 747ms/step - loss: 0.0427 -
accuracy: 0.9867 - val_loss: 0.0383 - val_accuracy: 0.9844
Epoch 11/25
450/450 [==============================] - 348s 774ms/step - loss: 0.0339 -
accuracy: 0.9901 - val_loss: 0.0430 - val_accuracy: 0.9869
Epoch 12/25
450/450 [==============================] - 336s 747ms/step - loss: 0.0347 -
accuracy: 0.9890 - val_loss: 0.0469 - val_accuracy: 0.9881
Epoch 13/25
450/450 [==============================] - 346s 770ms/step - loss: 0.0370 -
accuracy: 0.9900 - val_loss: 0.0452 - val_accuracy: 0.9862
Epoch 14/25
450/450 [==============================] - 339s 754ms/step - loss: 0.0416 -
```

```
accuracy: 0.9876 - val_loss: 0.0347 - val_accuracy: 0.9869
Epoch 15/25
450/450 [==============================] - 339s 754ms/step - loss: 0.0351 -
accuracy: 0.9884 - val_loss: 0.0372 - val_accuracy: 0.9900
Epoch 16/25
450/450 [==============================] - 334s 742ms/step - loss: 0.0322 -
accuracy: 0.9905 - val_loss: 0.0642 - val_accuracy: 0.9743
Epoch 17/25
450/450 [==============================] - 335s 744ms/step - loss: 0.0336 -
accuracy: 0.9905 - val_loss: 0.0718 - val_accuracy: 0.9831
Epoch 18/25
450/450 [==============================] - 335s 745ms/step - loss: 0.0387 -
accuracy: 0.9883 - val_loss: 0.0272 - val_accuracy: 0.9919
Epoch 19/25
450/450 [==============================] - 335s 744ms/step - loss: 0.0391 -
accuracy: 0.9890 - val_loss: 0.0289 - val_accuracy: 0.9900
Epoch 20/25
450/450 [==============================] - 336s 746ms/step - loss: 0.0303 -
accuracy: 0.9906 - val_loss: 0.0271 - val_accuracy: 0.9906
Epoch 21/25
450/450 [==============================] - 335s 744ms/step - loss: 0.0364 -
accuracy: 0.9894 - val_loss: 0.0183 - val_accuracy: 0.9937
Epoch 22/25
450/450 [==============================] - 334s 743ms/step - loss: 0.0290 -
accuracy: 0.9910 - val_loss: 0.0243 - val_accuracy: 0.9906
Epoch 23/25
450/450 [==============================] - 345s 766ms/step - loss: 0.0364 -
accuracy: 0.9885 - val_loss: 0.0344 - val_accuracy: 0.9875
Epoch 24/25
450/450 [==============================] - 342s 760ms/step - loss: 0.0294 -
accuracy: 0.9908 - val_loss: 0.0324 - val_accuracy: 0.9925
Epoch 25/25
450/450 [==============================] - 341s 758ms/step - loss: 0.0309 -
accuracy: 0.9896 - val_loss: 0.0321 - val_accuracy: 0.9894

Training and validation time:  2:22:38.071220
```

In [29]:

```
# 5.Model training history

acc = CNN_xcep_history.history['accuracy']
val_acc = CNN_xcep_history.history['val_accuracy']
loss = CNN_xcep_history.history['loss']
val_loss = CNN_xcep_history.history['val_loss']
epochs = range(1, len(acc) + 1)

plt.title('Training and validation accuracy')
plt.plot(epochs, acc, 'red', label='Training acc')
plt.plot(epochs, val_acc, 'blue', label='Validation acc')
plt.legend()

plt.figure()
plt.title('Training and validation loss')
plt.plot(epochs, loss, 'red', label='Training loss')
plt.plot(epochs, val_loss, 'blue', label='Validation loss')

plt.legend()

plt.show()
```





In [30]:
```
# 6.Viewing results and generating forecasts

score_xcep = CNN_xcep.evaluate(test_set)
print("Test Loss:", score_xcep[0])
print("Test Accuracy:", score_xcep[1])
```

```
56/56 [==============================] - 50s 883ms/step - loss: 0.0301 - ac
curacy: 0.9893
Test Loss: 0.030061062425374985
Test Accuracy: 0.9893018007278442
```

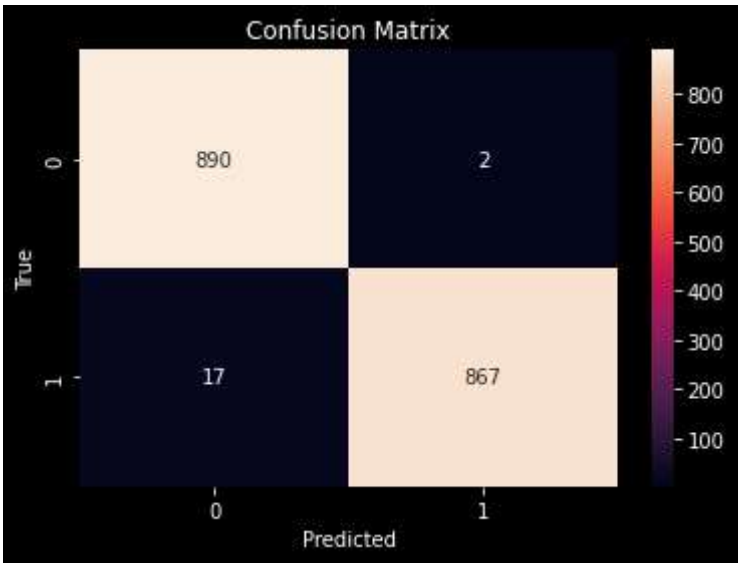
In [31]:
```
y_pred_xcep = CNN_xcep.predict(test_set)
y_pred_xcep = np.round(y_pred_xcep)

recall_xcep = recall_score(y_test, y_pred_xcep)
precision_xcep = precision_score(y_test, y_pred_xcep)
f1_xcep = f1_score(y_test, y_pred_xcep)
roc_xcep = roc_auc_score(y_test, y_pred_xcep)
```

```
56/56 [==============================] - 42s 721ms/step
```

In [32]:
```
print(classification_report(y_test, y_pred_xcep))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       892
           1       1.00      0.98      0.99       884

    accuracy                           0.99      1776
   macro avg       0.99      0.99      0.99      1776
weighted avg       0.99      0.99      0.99      1776
```

In [33]:
```
plt.figure(figsize = (6, 4))

sns.heatmap(confusion_matrix(y_test, y_pred_xcep),annot = True, fmt = 'd')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")

plt.show()
```



In [34]:
```
# 7. Viewing the results of all models

models= [('Inception', time_CNN_inc, np.mean(CNN_inc_history.history['accurac
         ('Xception', time_CNN_xcep, np.mean(CNN_xcep_history.history['accura
        ]

df_all_models = pd.DataFrame(models, columns = ['Model', 'Time', 'Training ac

df_all_models
```

Out[34]:

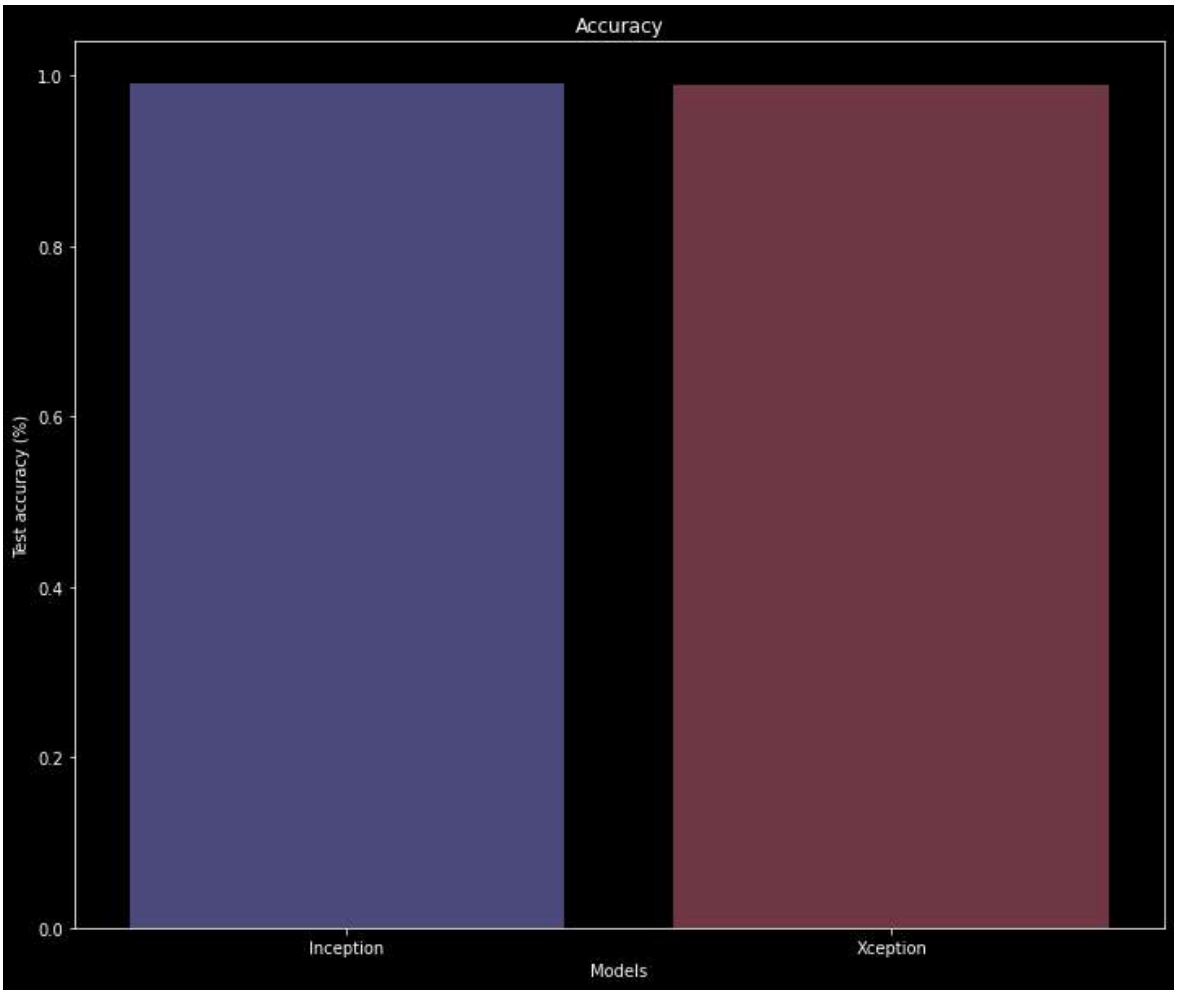
	Model	Time	Training accuracy (%)	Validation Accuracy (%)
0	Inception	0 days 01:17:47.967123	0.979680	0.981402
1	Xception	0 days 02:22:38.071220	0.987755	0.986809

In [35]:
```
models = [('Inception', score_inc[1], recall_inc, precision_inc, f1_inc, roc_
          ('Xception', score_xcep[1], recall_xcep, precision_xcep, f1_xcep, r
          ]

df_all_models_testset = pd.DataFrame(models, columns = ['Model', 'Test accura

df_all_models_testset
```

Out[35]:

	Model	Test accuracy (%)	Recall (%)	Precision (%)	F1 (%)	AUC
0	Inception	0.990991	0.997738	0.984375	0.991011	0.991021
1	Xception	0.989302	0.980769	0.997699	0.989161	0.989264

In [36]:
```
plt.subplots(figsize=(12, 10))
sns.barplot(y = df_all_models_testset['Test accuracy (%)'], x = df_all_models
plt.xlabel("Models")
plt.title('Accuracy')
plt.show()
```



In [ ]:
```

```