IMPORTING ALL NECESSARY LIBRARIES AND FUNCTIONS

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import cv2
import sklearn
from sklearn import model_selection
from sklearn.model_selection import train_test_split, learning_curve, KFold, cross_val_
from sklearn.utils import class_weight
from sklearn.metrics import confusion_matrix

import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten, Conv2D, MaxPool2D, MaxPoo
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.np_utils import to_categorical
from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
from keras.models import model_from_json
from keras import backend as K
from google.colab import files
from glob import glob
import random
import matplotlib.gridspec as gridspec
import seaborn as sns
import zlib
import itertools
from keras.utils.vis_utils import plot_model
from keras.preprocessing import image
from keras.applications.mobilenet import preprocess_input
from keras.models import Model
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import classification_report

from keras import optimizers
from google.colab import files
```

CHECKING IF THE GPU IS WORKING

```
import tensorflow as tf
tf.test.gpu_device_name()
```

```
'/device:GPU:0'
```

IMPORTING THE DATASET FROM GOOGLE DRIVE

```
from google.colab import drive
drive.mount('/content/drive/')

train_path='/content/drive/MyDrive/train_dataset'
test_path='/content/drive/MyDrive/test_dataset'

train_datagen=ImageDataGenerator(rescale=1./255,shear_range=.2, zoom_range=.2,horizonta
test_datagen=ImageDataGenerator(rescale=1./255)

training_set=train_datagen.flow_from_directory(train_path,target_size=(224,224),batch_s
testing_set=test_datagen.flow_from_directory(test_path,target_size=(224,224),batch_size
```

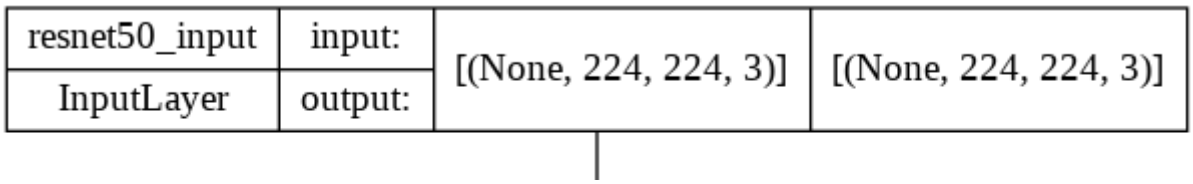
```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive
Found 4596 images belonging to 2 classes.
Found 2159 images belonging to 2 classes.
```

ResNet-50 MODEL

MAKING, COMPILATION, STRUCTURE, FITTING, PROGRESS VISUALIZATION AND HEATMAP

```
from tensorflow.keras.applications import ResNet50

model4=Sequential()

# adding the pre trained ResNet 50 model
model4.add(ResNet50(include_top=False,weights='imagenet',pooling='max',input_shape=(224

# adding the output layer
model4.add(Dense(2,activation='softmax'))

# compilig the model
model4.compile(loss='categorical_crossentropy',optimizer=tf.optimizers.SGD(lr=.01),metr

# printing the blueprint of the model
plot_model(model4, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/re
94773248/94765736 [==============================] - 0s 0us/step
94781440/94765736 [==============================] - 0s 0us/step
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: Us
  super(SGD, self).__init__(name, **kwargs)
```

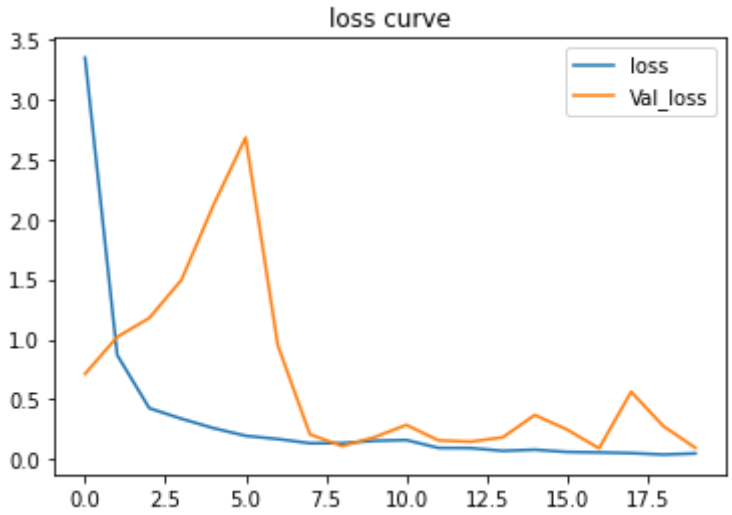



```
# fitting the alexnet model
r4=model4.fit_generator(training_set,validation_data=testing_set,epochs=20,steps_per_ep

# keys present in the history
print(r4.history.keys())
```

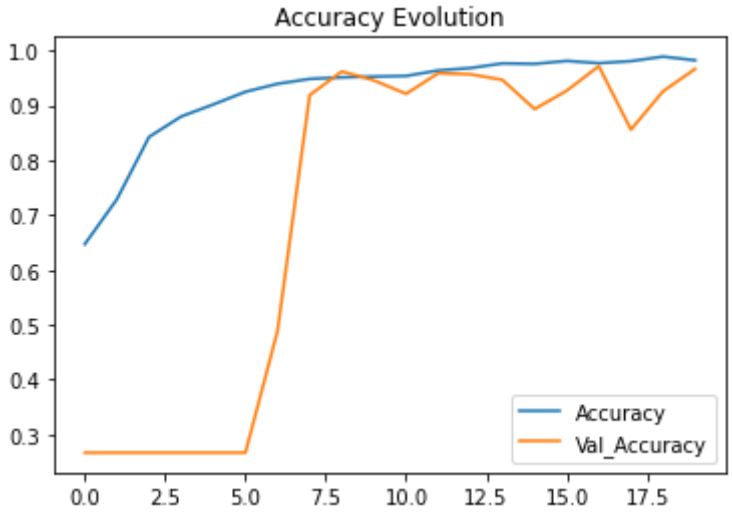
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Model.f

Epoch 1/20
144/144 [==============================] - 1148s 8s/step - loss: 3.3457 - accuracy: 0
Epoch 2/20
144/144 [==============================] - 193s 1s/step - loss: 0.8694 - accuracy: 0
Epoch 3/20
144/144 [==============================] - 185s 1s/step - loss: 0.4263 - accuracy: 0
Epoch 4/20
144/144 [==============================] - 180s 1s/step - loss: 0.3386 - accuracy: 0
Epoch 5/20
144/144 [==============================] - 185s 1s/step - loss: 0.2600 - accuracy: 0
Epoch 6/20
144/144 [==============================] - 184s 1s/step - loss: 0.1962 - accuracy: 0
Epoch 7/20
144/144 [==============================] - 184s 1s/step - loss: 0.1695 - accuracy: 0
Epoch 8/20
144/144 [==============================] - 186s 1s/step - loss: 0.1346 - accuracy: 0
Epoch 9/20
144/144 [==============================] - 184s 1s/step - loss: 0.1353 - accuracy: 0
Epoch 10/20
144/144 [==============================] - 184s 1s/step - loss: 0.1544 - accuracy: 0
Epoch 11/20
144/144 [==============================] - 184s 1s/step - loss: 0.1611 - accuracy: 0
Epoch 12/20
144/144 [==============================] - 185s 1s/step - loss: 0.0954 - accuracy: 0
Epoch 13/20
144/144 [==============================] - 185s 1s/step - loss: 0.0940 - accuracy: 0
Epoch 14/20
144/144 [==============================] - 185s 1s/step - loss: 0.0710 - accuracy: 0
Epoch 15/20
144/144 [==============================] - 184s 1s/step - loss: 0.0803 - accuracy: 0
Epoch 16/20
144/144 [==============================] - 184s 1s/step - loss: 0.0619 - accuracy: 0
Epoch 17/20
144/144 [==============================] - 186s 1s/step - loss: 0.0577 - accuracy: 0
Epoch 18/20
144/144 [==============================] - 185s 1s/step - loss: 0.0523 - accuracy: 0
Epoch 19/20
144/144 [==============================] - 187s 1s/step - loss: 0.0402 - accuracy: 0
Epoch 20/20
```

```
144/144 [==============================] - 185s 1s/step - loss: 0.0494 - accuracy: 0
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
1 import matplotlib.pyplot as plt
2
3 plt.plot(r4.history['loss'],label='loss')
4 plt.plot(r4.history['val_loss'],label='Val_loss')
5 plt.title('loss curve')
6 plt.legend()
7 plt.show()
8
9 plt.plot(r4.history['accuracy'], label='Accuracy')
10 plt.plot(r4.history['val_accuracy'], label='Val_Accuracy')
11 plt.title('accuracy curve')
12 plt.legend()
13 plt.title('Accuracy Evolution')
```




```
Text(0.5, 1.0, 'Accuracy Evolution')
```




```
1 evaluation=model4.evaluate(testing_set)
2 print(f"Test Accuracy: {evaluation[1] * 100:.2f}%")
3
4 evaluation=model4.evaluate(training_set)
5 print(f"Train Accuracy: {evaluation[1] * 100:.2f}%")
```

```
68/68 [==============================] - 55s 807ms/step - loss: 0.0962 - accuracy: 0
Test Accuracy: 96.67%
144/144 [==============================] - 119s 824ms/step - loss: 0.0485 - accuracy
Train Accuracy: 98.13%
```