

▼ IMPORTING ALL NECESSARY LIBRARIES AND FUNCTIONS

```
1 import pandas as pd
2 import numpy as np
3 import os
4 import matplotlib.pyplot as plt
5 import cv2
6 import sklearn
7 from sklearn import model_selection
8 from sklearn.model_selection import train_test_split, learning_curve, KFold, cross_val_
9 from sklearn.utils import class_weight
10 from sklearn.metrics import confusion_matrix
11
12 import keras
13 from keras.models import Sequential
14 from keras.layers import Dense, Activation, Dropout, Flatten, Conv2D, MaxPool2D, MaxPoo
15 from keras.preprocessing.image import ImageDataGenerator
16 from keras.utils.np_utils import to_categorical
17 from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
18 from keras.models import model_from_json
19 from keras import backend as K
20 from google.colab import files
21 from glob import glob
22 import random
23 import matplotlib.gridspec as gridspec
24 import seaborn as sns
25 import zlib
26 import itertools
27 from keras.utils.vis_utils import plot_model
28 from keras.preprocessing import image
29 from keras.applications.mobilenet import preprocess_input
30 from keras.models import Model
31 from mlxtend.plotting import plot_confusion_matrix
32 from sklearn.metrics import classification_report
33
34 from keras import optimizers
35 from google.colab import files
```

▼ CHECKING IF THE GPU IS WORKING

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

'/device:GPU:0'
```


▼ IMPORTING THE DATASET FROM GOOGLE DRIVE

```

1 from google.colab import drive
2 drive.mount('/content/drive/')
3
4 train_path='/content/drive/MyDrive/image_classification/train_dataset'
5 test_path='/content/drive/MyDrive/image_classification/test_dataset'
6
7 train_datagen=ImageDataGenerator(rescale=1./255,shear_range=.2, zoom_range=.2,horizonta
8 test_datagen=ImageDataGenerator(rescale=1./255)
9
10 training_set=train_datagen.flow_from_directory(train_path,target_size=(224,224),batch_s
11 testing_set=test_datagen.flow_from_directory(test_path,target_size=(224,224),batch_size
12
13 training_set_320=train_datagen.flow_from_directory(train_path,target_size=(320,320),bat
14 testing_set_320=test_datagen.flow_from_directory(test_path,target_size=(320,320),batch_

    Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive
    Found 4593 images belonging to 2 classes.
    Found 2166 images belonging to 2 classes.
    Found 4593 images belonging to 2 classes.
    Found 2166 images belonging to 2 classes.

```



▼ GoogleNet MODEL

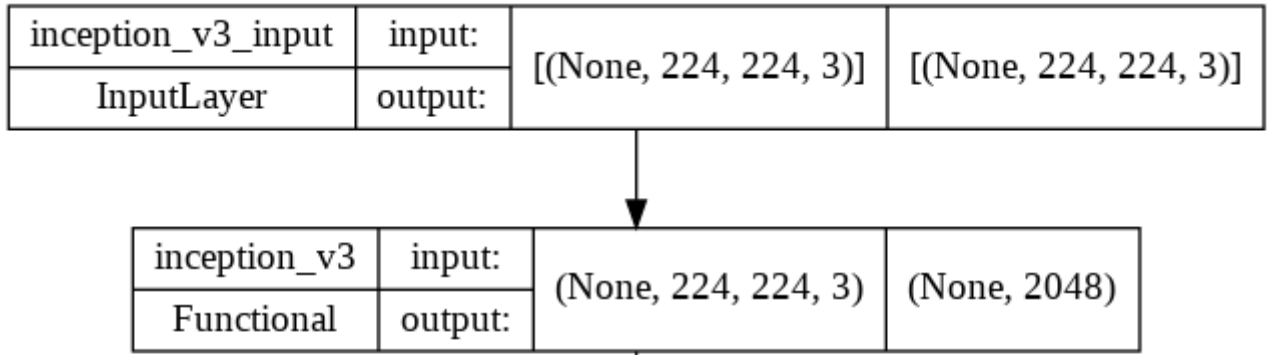
MAKING, COMPILATION, STRUCTURE, FITTING, PROGRESS VISUALIZATION AND HEATMAP

```

1 from tensorflow.keras.applications.inception_v3 import InceptionV3
2
3 model3=Sequential()
4
5 # adding the pretrained InceptionNet model
6 model3.add(InceptionV3(include_top=False,weights='imagenet',pooling='max',input_shape=(
7
8 # adding the output layer
9 model3.add(Dense(2,activation='softmax')))
10
11 # compiling the model
12 model3.compile(loss='categorical_crossentropy',optimizer=tf.optimizers.SGD(lr=.01),metr
13
14 # printing the blueprint of the model
15 plot_model(model3, to_file='model_plot.png', show_shapes=True, show_layer_names=True)

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3_weights_tf_dim_ordering_tf_data_format.h5 [=====] - 0s 0us/step
 87924736/87910968 [=====] - 0s 0us/step
 /usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: UserWarning: Using the SGD optimizer may be slower than Adam
 super(SGD, self).__init__(name, **kwargs)



```
1 # fitting the alexnet model
2 r3=model3.fit_generator(training_set,validation_data=testing_set,epochs=10,steps_per_ep
3
4 # keys present in the history
5 print(r3.history.keys())
```

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

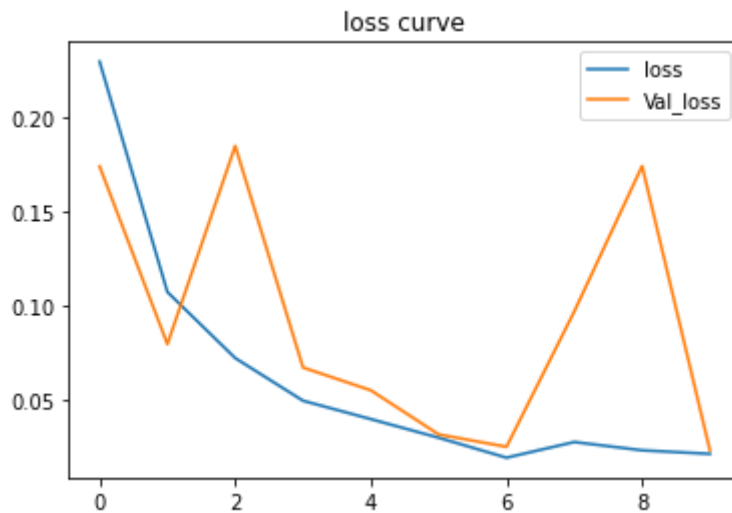
```
Epoch 1/10
144/144 [=====] - 1166s 8s/step - loss: 0.2300 - accuracy: 0.0000
Epoch 2/10
144/144 [=====] - 196s 1s/step - loss: 0.1074 - accuracy: 0.0000
Epoch 3/10
144/144 [=====] - 190s 1s/step - loss: 0.0722 - accuracy: 0.0000
Epoch 4/10
144/144 [=====] - 188s 1s/step - loss: 0.0495 - accuracy: 0.0000
Epoch 5/10
144/144 [=====] - 187s 1s/step - loss: 0.0398 - accuracy: 0.0000
Epoch 6/10
144/144 [=====] - 187s 1s/step - loss: 0.0299 - accuracy: 0.0000
Epoch 7/10
144/144 [=====] - 185s 1s/step - loss: 0.0192 - accuracy: 0.0000
Epoch 8/10
144/144 [=====] - 185s 1s/step - loss: 0.0276 - accuracy: 0.0000
Epoch 9/10
144/144 [=====] - 185s 1s/step - loss: 0.0232 - accuracy: 0.0000
Epoch 10/10
144/144 [=====] - 185s 1s/step - loss: 0.0213 - accuracy: 0.0000
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
1 import matplotlib.pyplot as plt
2
3 plt.plot(r3.history['loss'],label='loss')
4 plt.plot(r3.history['val_loss'],label='Val_loss')
5 plt.title('loss curve')
6 plt.legend()
7 plt.show()
8
9 plt.plot(r3.history['accuracy'], label='Accuracy')
10 plt.plot(r3.history['val_accuracy'], label='Val_Accuracy')
```

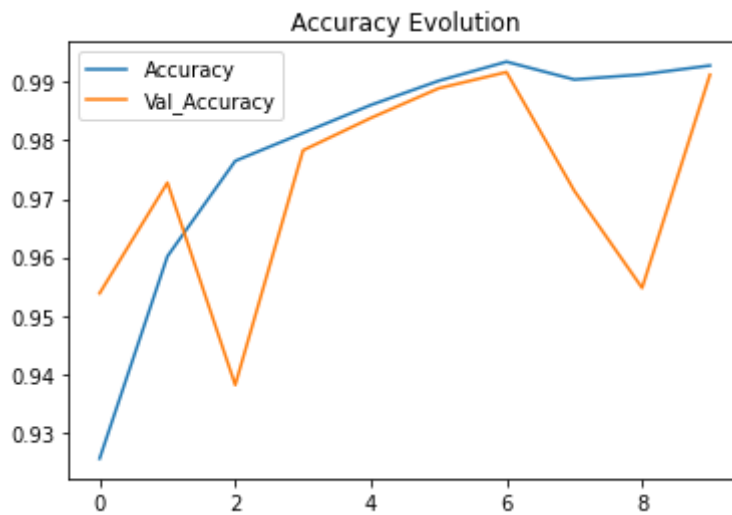
```

11 plt.title('accuracy curve')
12 plt.legend()
13 plt.title('Accuracy Evolution')

```



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



```

1 evaluation=model3.evaluate(testing_set)
2 print(f"Test Accuracy: {evaluation[1] * 100:.2f}%")
3
4 evaluation=model3.evaluate(training_set)
5 print(f"Train Accuracy: {evaluation[1] * 100:.2f}%")

```

```

68/68 [=====] - 57s 832ms/step - loss: 0.0231 - accuracy: 0
Test Accuracy: 99.12%
144/144 [=====] - 124s 863ms/step - loss: 0.0192 - accuracy
Train Accuracy: 99.26%

```



[Colab paid products](#) - [Cancel contracts here](#)

