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 4591 images belonging to 2 classes.
Found 2159 images belonging to 2 classes.
```

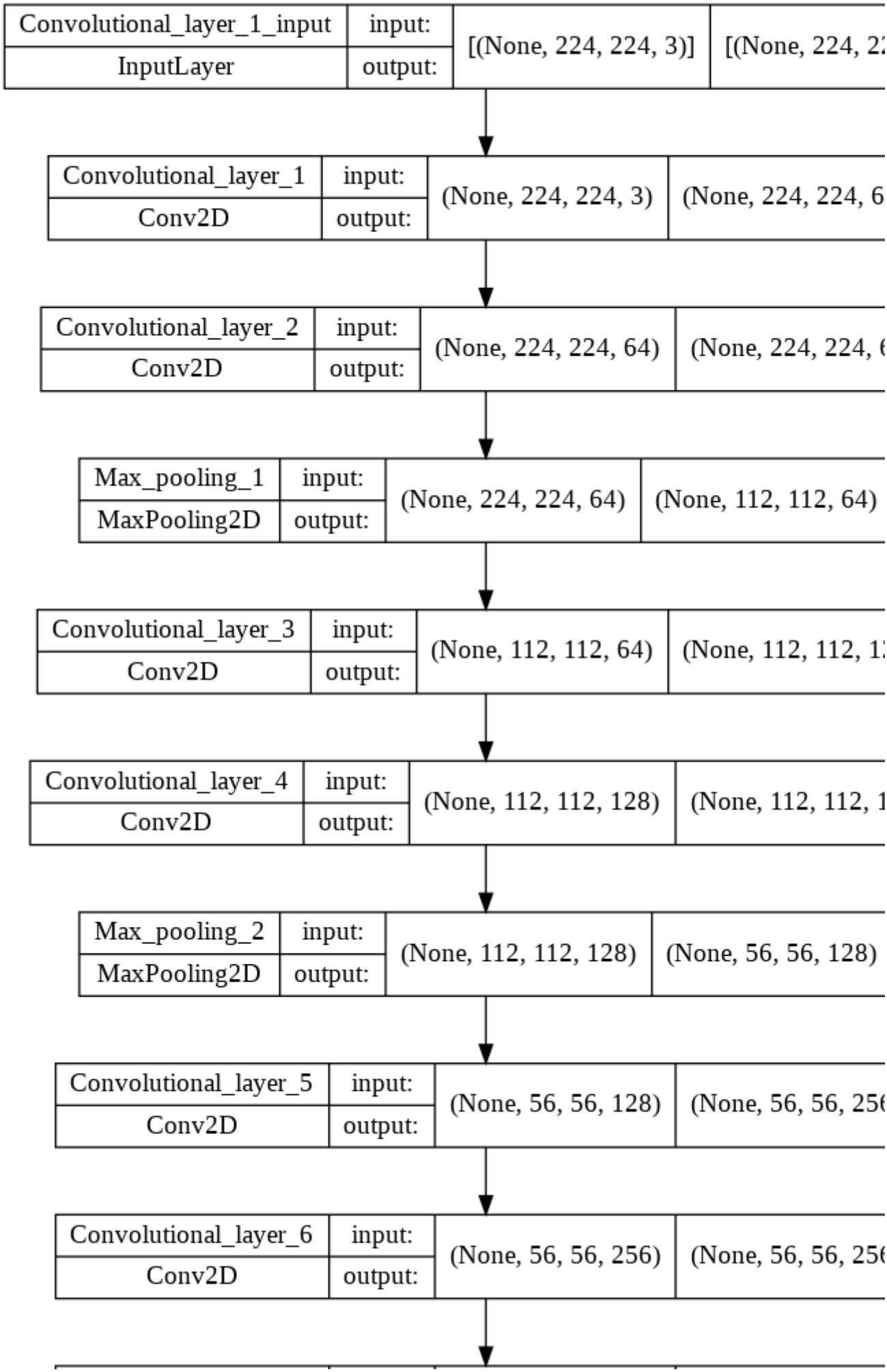
VGG-19 MODEL

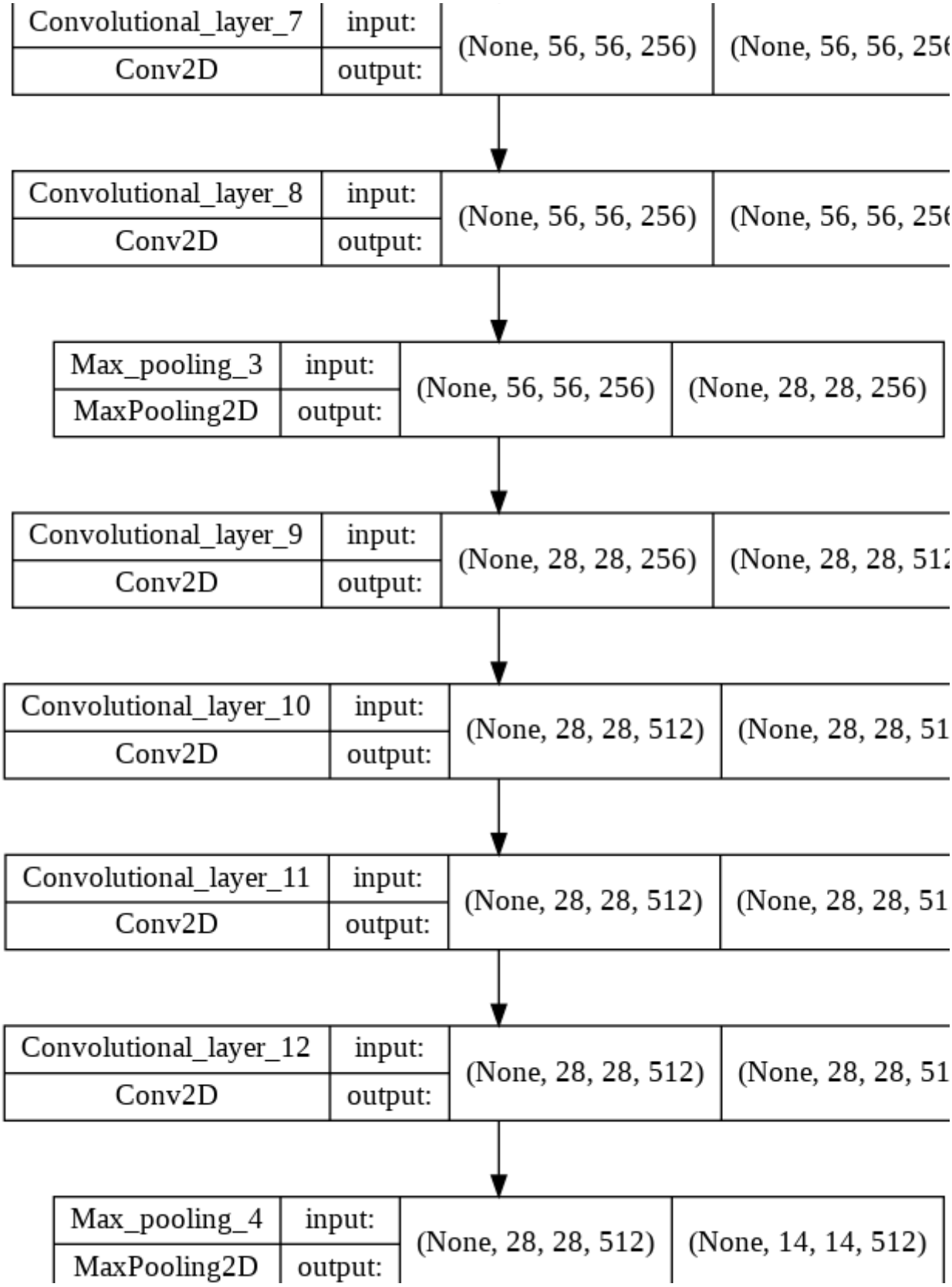
MAKING, COMPILATION, STRUCTURE, FITTING, PROGRESS VISUALIZATION AND HEATMAP

```
model2=Sequential()

# Layer 1: Convolutional
model2.add(Conv2D(filters=64,kernel_size=3,activation='relu',padding='same',name='Convo

# Layer 2: Convolutional
model2.add(Conv2D(filters=64,kernel_size=3,activation='relu',padding='same',name='Convo

# MaxPool 1
model2.add(MaxPool2D(pool_size=2,strides=2,name='Max_pooling_1'))

# Layer 3: Convolutional
model2.add(Conv2D(filters=128,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 4: Convolutional
model2.add(Conv2D(filters=128,kernel_size=3,activation='relu',padding='same',name='Conv

# MaxPool 2
model2.add(MaxPool2D(pool_size=2,strides=2,name='Max_pooling_2'))

# Layer 5: Convolutional
model2.add(Conv2D(filters=256,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 6: Convolutional
model2.add(Conv2D(filters=256,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 7: Convolutional
```

```
model2.add(Conv2D(filters=256,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 8: Convolutional
model2.add(Conv2D(filters=256,kernel_size=3,activation='relu',padding='same',name='Conv

# MaxPool 3
model2.add(MaxPool2D(pool_size=2,strides=2,name='Max_pooling_3'))

# Layer 9: Convolutional
model2.add(Conv2D(filters=512,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 10: Convolutional
model2.add(Conv2D(filters=512,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 11: Convolutional
model2.add(Conv2D(filters=512,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 12: Convolutional
model2.add(Conv2D(filters=512,kernel_size=3,activation='relu',padding='same',name='Conv

# MaxPool 4
model2.add(MaxPool2D(pool_size=2,strides=2,name='Max_pooling_4'))

# Layer 13: Convolutional
model2.add(Conv2D(filters=512,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 14: Convolutional
model2.add(Conv2D(filters=512,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 15: Convolutional
model2.add(Conv2D(filters=512,kernel_size=3,activation='relu',padding='same',name='Conv

# Layer 16: Convolutional
model2.add(Conv2D(filters=512,kernel_size=3,activation='relu',padding='same',name='Conv

# MaxPool 5
model2.add(MaxPool2D(pool_size=2,strides=2,name='Max_pooling_5'))

# flatten
model2.add(Flatten(input_shape=[227,227,3]))
/
# Layer 17: Fully Connected
model2.add(Dense(units=4096,activation='relu',name='Dense_layer_1'))

# Layer 18: Fully Connected
model2.add(Dense(units=4096,activation='relu',name='Dense_layer_2'))

# Layer 19: Fully Connected
model2.add(Dense(units=2,activation='softmax',name='Dense_layer_3'))

# compiling the model
model2.compile(loss='categorical_crossentropy',optimizer=tf.optimizers.SGD(lr=.01),metr

# printing the blueprint of the model
plot_model(model2, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/gradient_descent.py:102: Us
  super(SGD, self).__init__(name, **kwargs)
```





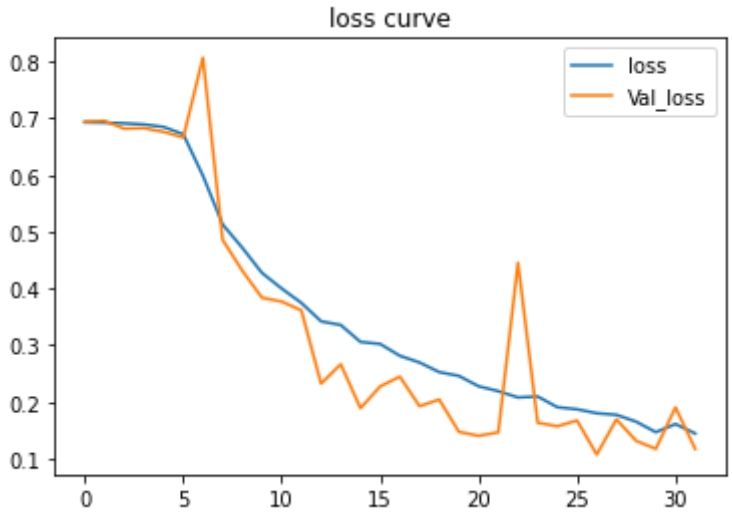

```
# fitting the alexnet model
r2=model2.fit_generator(training_set,validation_data=testing_set,epochs=32,steps_per_ep

# keys present in the history
print(r2.history.keys())
```

```
Epoch 5/32
144/144 [==============================] - 190s 1s/step - loss: 0.6848 - accuracy:
Epoch 6/32
144/144 [==============================] - 190s 1s/step - loss: 0.6719 - accuracy:
Epoch 7/32
144/144 [==============================] - 190s 1s/step - loss: 0.5988 - accuracy:
```

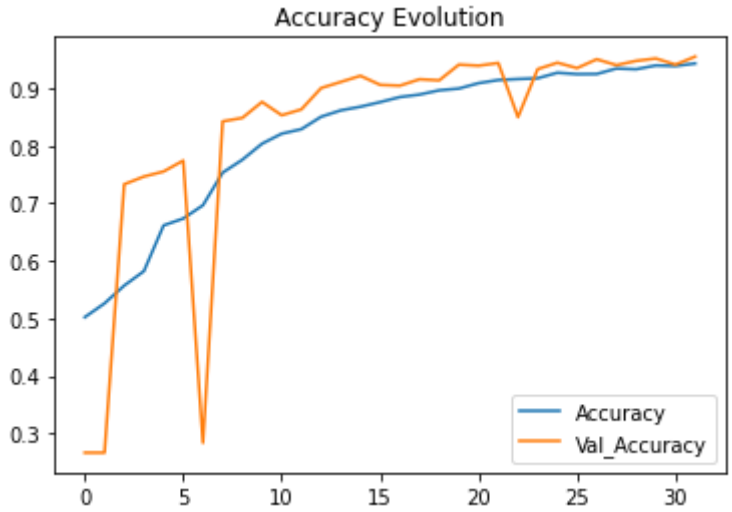
```
Epoch 8/32
144/144 [==============================] - 189s 1s/step - loss: 0.5128 - accuracy:
Epoch 9/32
144/144 [==============================] - 189s 1s/step - loss: 0.4717 - accuracy:
Epoch 10/32
144/144 [==============================] - 189s 1s/step - loss: 0.4273 - accuracy:
Epoch 11/32
144/144 [==============================] - 190s 1s/step - loss: 0.3998 - accuracy:
Epoch 12/32
144/144 [==============================] - 189s 1s/step - loss: 0.3745 - accuracy:
Epoch 13/32
144/144 [==============================] - 189s 1s/step - loss: 0.3420 - accuracy:
Epoch 14/32
144/144 [==============================] - 188s 1s/step - loss: 0.3355 - accuracy:
Epoch 15/32
144/144 [==============================] - 190s 1s/step - loss: 0.3057 - accuracy:
Epoch 16/32
144/144 [==============================] - 189s 1s/step - loss: 0.3021 - accuracy:
Epoch 17/32
144/144 [==============================] - 189s 1s/step - loss: 0.2813 - accuracy:
Epoch 18/32
144/144 [==============================] - 188s 1s/step - loss: 0.2694 - accuracy:
Epoch 19/32
144/144 [==============================] - 189s 1s/step - loss: 0.2525 - accuracy:
Epoch 20/32
144/144 [==============================] - 188s 1s/step - loss: 0.2458 - accuracy:
Epoch 21/32
144/144 [==============================] - 188s 1s/step - loss: 0.2275 - accuracy:
Epoch 22/32
144/144 [==============================] - 189s 1s/step - loss: 0.2188 - accuracy:
Epoch 23/32
144/144 [==============================] - 193s 1s/step - loss: 0.2078 - accuracy:
Epoch 24/32
144/144 [==============================] - 211s 1s/step - loss: 0.2094 - accuracy:
Epoch 25/32
144/144 [==============================] - 223s 2s/step - loss: 0.1906 - accuracy:
Epoch 26/32
144/144 [==============================] - 202s 1s/step - loss: 0.1869 - accuracy:
Epoch 27/32
144/144 [==============================] - 197s 1s/step - loss: 0.1800 - accuracy:
Epoch 28/32
144/144 [==============================] - 189s 1s/step - loss: 0.1771 - accuracy:
Epoch 29/32
144/144 [==============================] - 191s 1s/step - loss: 0.1650 - accuracy:
Epoch 30/32
144/144 [==============================] - 190s 1s/step - loss: 0.1465 - accuracy:
Epoch 31/32
144/144 [==============================] - 190s 1s/step - loss: 0.1609 - accuracy:
Epoch 32/32
144/144 [==============================] - 191s 1s/step - loss: 0.1442 - accuracy:
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
1 import matplotlib.pyplot as plt
2
3 plt.plot(r2.history['loss'],label='loss')
4 plt.plot(r2.history['val_loss'],label='Val_loss')
5 plt.title('loss curve')
```

```
plt.legend()
plt.show()

plt.plot(r2.history['accuracy'], label='Accuracy')
plt.plot(r2.history['val_accuracy'], label='Val_Accuracy')
plt.title('accuracy curve')
plt.legend()
plt.title('Accuracy Evolution')
```




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




```
evaluation=model2.evaluate(testing_set)
print(f"Test Accuracy: {evaluation[1] * 100:.2f}%")

evaluation=model2.evaluate(training_set)
print(f"Train Accuracy: {evaluation[1] * 100:.2f}%")
```

```
68/68 [==============================] - 55s 805ms/step - loss: 0.1171 - accuracy: 0
Test Accuracy: 95.51%
144/144 [==============================] - 121s 840ms/step - loss: 0.1201 - accuracy
Train Accuracy: 95.51%
```