In [1]:
```
import numpy as np
import PIL.Image
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import os
import time
```

In [26]:
```
model=keras.models.Sequential([
    #CONV1
    keras.layers.ZeroPadding2D(padding=(2, 2)),
    keras.layers.Conv2D(filters=128, kernel_size=(3,3), strides=(1,1), activation='relu',padding='valid', input
    keras.layers.BatchNormalization(),
    #CONV2
    keras.layers.ZeroPadding2D(padding=(2, 2)),
    keras.layers.Conv2D(filters=128, kernel_size=(3,3), strides=(1,1), activation='relu', padding="valid"),
    keras.layers.BatchNormalization(),
    #MaxPool1
    keras.layers.MaxPool2D(pool_size=(2,2)),
    #CONV3
    keras.layers.Conv2D(filters=63, kernel_size=(3,3), strides=(2,2), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    #CONV4
    keras.layers.Conv2D(filters=74, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    #MaxPool2
    keras.layers.MaxPool2D(pool_size=(2,2)),
    #CONV5
    keras.layers.Conv2D(filters=32, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    #CONV6
    keras.layers.Conv2D(filters=32, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Flatten(),
    #FC1
    keras.layers.Dense(1024,activation='relu'),
    keras.layers.Dropout(0.5),
    #FC2
    keras.layers.Dense(1024,activation='relu'),
    keras.layers.Dropout(0.5),
    #Output
    keras.layers.Dense(10,activation='softmax')
    ])
```

In [27]:
```
import cv2

im = np.float32(cv2.imread('lena_new.jpeg'))
print(im.shape)
```

```
(225, 225, 3)
```

In [37]:
```
type(im)
```

Out[37]:
```
numpy.ndarray
```

In [28]:
```
im=im.reshape((1,225,225,3))
```

In [21]:
```
im.shape
```

Out[21]:
```
(1, 225, 225, 3)
```

In [29]:
```
model.build(im.shape)
```

In [30]:
```
model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer=tf.optimizers.SGD(lr=0.001),
    metrics=['accuracy']
)
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 zero_padding2d_5 (ZeroPaddi  (1, 229, 229, 3)         0
 ng2D)

 conv2d_13 (Conv2D)          (1, 227, 227, 128)        3584

 batch_normalization_12 (Bat  (1, 227, 227, 128)       512
 chNormalization)

 zero_padding2d_6 (ZeroPaddi  (1, 231, 231, 128)       0
 ng2D)

 conv2d_14 (Conv2D)          (1, 229, 229, 128)        147584

 batch_normalization_13 (Bat  (1, 229, 229, 128)       512
 chNormalization)

 max_pooling2d_4 (MaxPooling  (1, 114, 114, 128)       0
 2D)

 conv2d_15 (Conv2D)          (1, 57, 57, 63)           72639

 batch_normalization_14 (Bat  (1, 57, 57, 63)          252
 chNormalization)

 conv2d_16 (Conv2D)          (1, 57, 57, 74)           42032

 batch_normalization_15 (Bat  (1, 57, 57, 74)          296
 chNormalization)

 max_pooling2d_5 (MaxPooling  (1, 28, 28, 74)          0
 2D)

 conv2d_17 (Conv2D)          (1, 28, 28, 32)           21344

 batch_normalization_16 (Bat  (1, 28, 28, 32)          128
 chNormalization)

 conv2d_18 (Conv2D)          (1, 28, 28, 32)           9248

 batch_normalization_17 (Bat  (1, 28, 28, 32)          128
 chNormalization)

 flatten_2 (Flatten)         (1, 25088)                0

 dense_6 (Dense)             (1, 1024)                 25691136

 dropout_4 (Dropout)         (1, 1024)                 0

 dense_7 (Dense)             (1, 1024)                 1049600

 dropout_5 (Dropout)         (1, 1024)                 0

 dense_8 (Dense)             (1, 10)                   10250

=================================================================
Total params: 27,049,245
Trainable params: 27,048,331
Non-trainable params: 914
_________________________________________________________________
```

In [31]:
```
pred=model.predict(im)
```

```
1/1 [==============================] - 0s 460ms/step
```

In [32]:
```
pred.shape
```

Out[32]:
```
(1, 10)
```

In [ ]: