DEEP LEARNING MICRO PROJECT

Real time object detection using pre-trained imagenet models

DEFINITION:

A convolutional neural network (CNN or convnet) is a subset of machine learning. It is one of the various types of artificial neural networks which are used for different applications and data types. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data.

There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. This makes them highly suitable for computer vision (CV) tasks and for applications where object recognition is vital, such as self-driving cars and facial recognition.

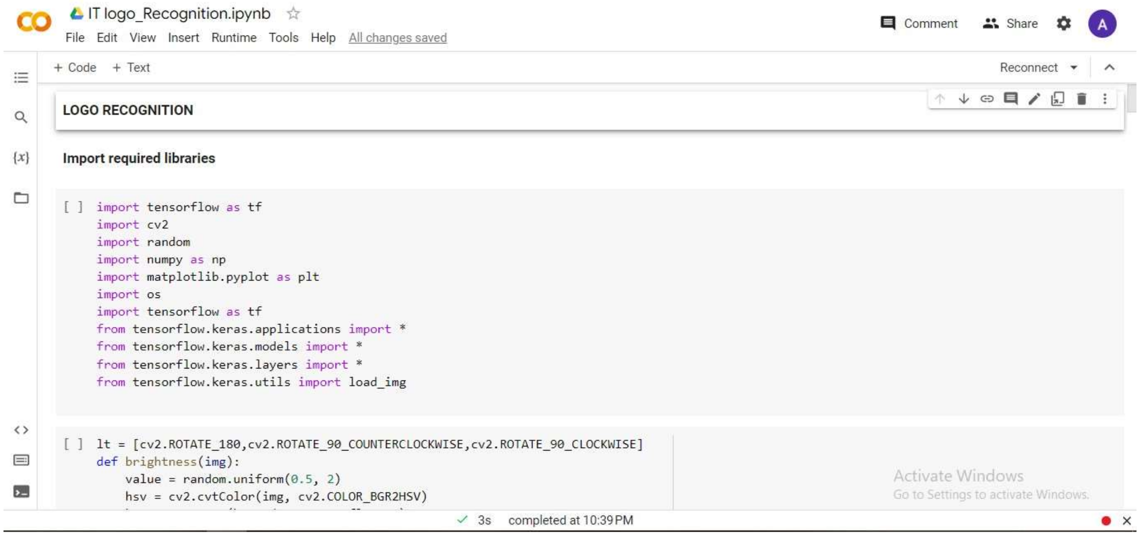
DATASET DESCRIPTION:

Data that we are using here is images. Images of Multinational company and IT company logos, which is in the format of jpg, jpeg, png.

For example:

DELL
accenture
IBM®
Infosys
wipro

PROGRAM CODE:

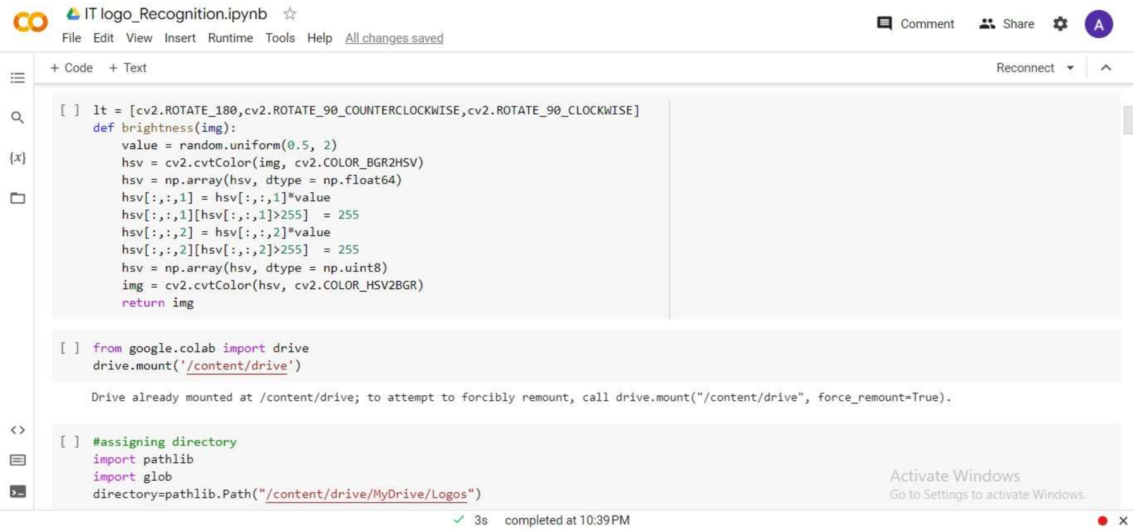

IT logo_Recognition.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
A
+ Code + Text
Reconnect
Activate Windows
Go to Settings to activate Windows.
✓ 3s completed at 10:39 PM

LOGO RECOGNITION

Import required libraries

```
import tensorflow as tf
import cv2
import random
import numpy as np
import matplotlib.pyplot as plt
import os
import tensorflow as tf
from tensorflow.keras.applications import *
from tensorflow.keras.models import *
from tensorflow.keras.layers import *
from tensorflow.keras.utils import load_img
```

```
lt = [cv2.ROTATE_180,cv2.ROTATE_90_COUNTERCLOCKWISE,cv2.ROTATE_90_CLOCKWISE]
def brightness(img):
    value = random.uniform(0.5, 2)
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

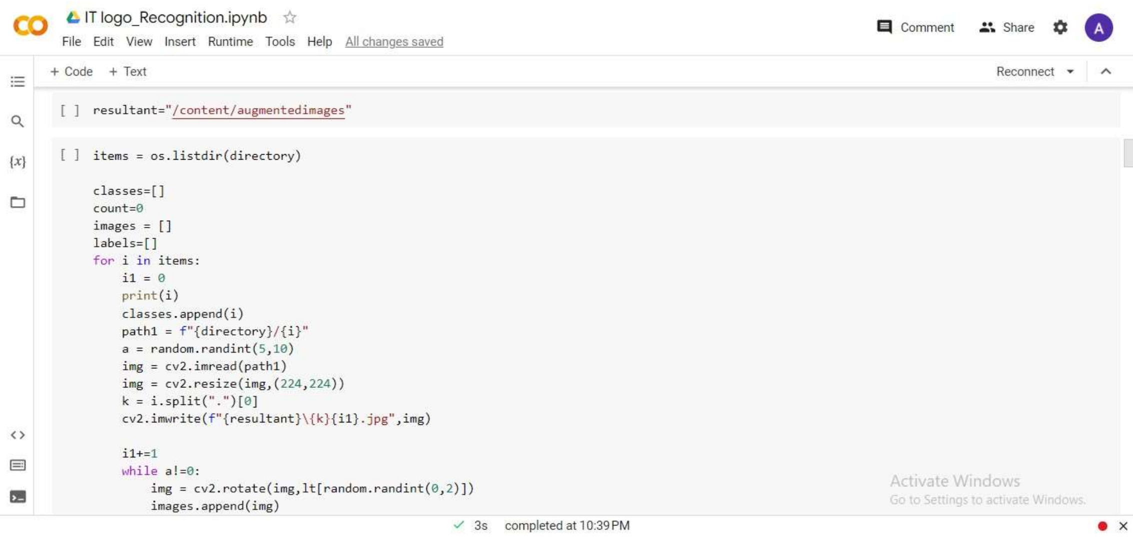

IT logo_Recognition.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
A
+ Code + Text
Reconnect
Activate Windows
Go to Settings to activate Windows.
✓ 3s completed at 10:39 PM

```
lt = [cv2.ROTATE_180,cv2.ROTATE_90_COUNTERCLOCKWISE,cv2.ROTATE_90_CLOCKWISE]
def brightness(img):
    value = random.uniform(0.5, 2)
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    hsv = np.array(hsv, dtype = np.float64)
    hsv[:,:,1] = hsv[:,:,1]*value
    hsv[:,:,1][hsv[:,:,1]>255]  = 255
    hsv[:,:,2] = hsv[:,:,2]*value
    hsv[:,:,2][hsv[:,:,2]>255]  = 255
    hsv = np.array(hsv, dtype = np.uint8)
    img = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
    return img
```

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

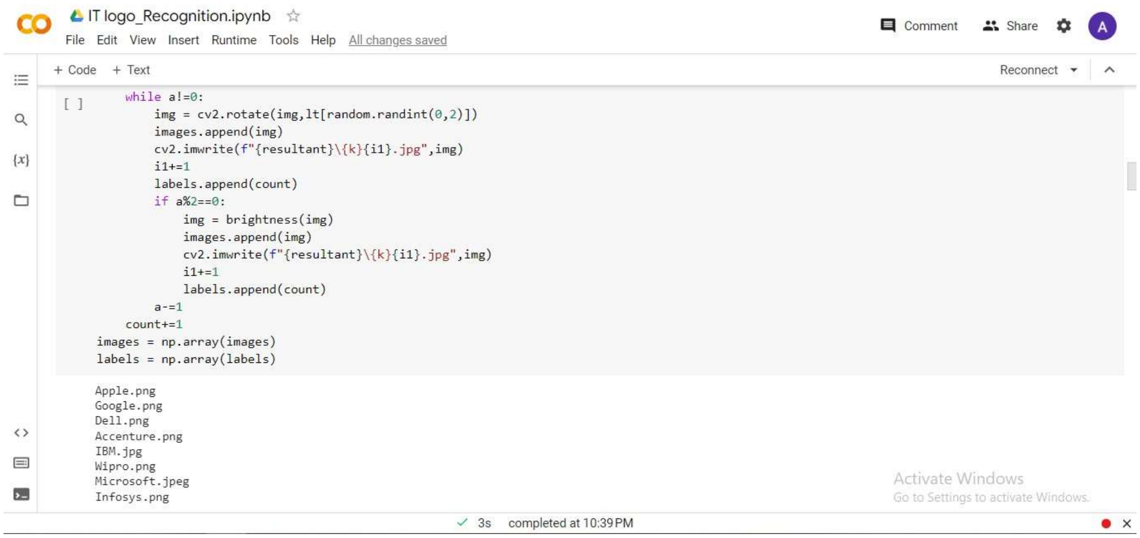
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#assigning directory
import pathlib
import glob
directory=pathlib.Path("/content/drive/MyDrive/Logos")
```


IT logo_Recognition.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Activate Windows
Go to Settings to activate Windows.
✓ 3s completed at 10:39 PM

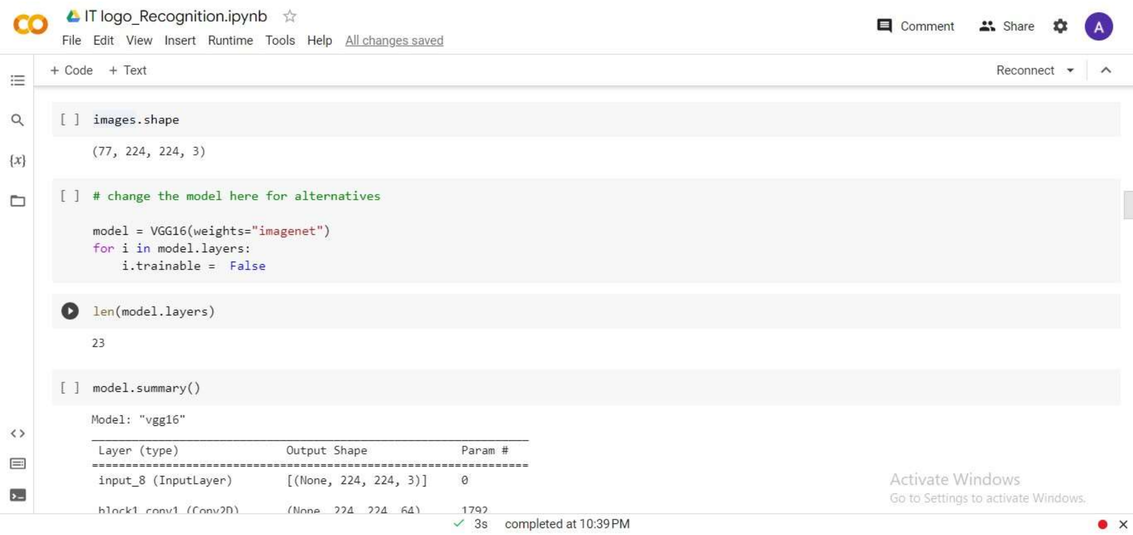
```
[ ] resultant="/content/augmentedimages"
```

```
[ ] items = os.listdir(directory)

    classes=[]
    count=0
    images = []
    labels=[]
    for i in items:
        i1 = 0
        print(i)
        classes.append(i)
        path1 = f"{directory}/{i}"
        a = random.randint(5,10)
        img = cv2.imread(path1)
        img = cv2.resize(img,(224,224))
        k = i.split(".")[0]
        cv2.imwrite(f"{resultant}\{k}{i1}.jpg",img)

        i1+=1
        while a!=0:
            img = cv2.rotate(img,lt[random.randint(0,2)])
            images.append(img)
```


IT logo_Recognition.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Activate Windows
Go to Settings to activate Windows.
✓ 3s completed at 10:39 PM

```
[ ]     while a!=0:
            img = cv2.rotate(img,lt[random.randint(0,2)])
            images.append(img)
            cv2.imwrite(f"{resultant}\{k}{i1}.jpg",img)
            i1+=1
            labels.append(count)
            if a%2==0:
                img = brightness(img)
                images.append(img)
                cv2.imwrite(f"{resultant}\{k}{i1}.jpg",img)
                i1+=1
                labels.append(count)
            a-=1
        count+=1
    images = np.array(images)
    labels = np.array(labels)
```

```
Apple.png
Google.png
Dell.png
Accenture.png
IBM.jpg
Wipro.png
Microsoft.jpeg
Infosys.png
```


IT logo_Recognition.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Activate Windows
Go to Settings to activate Windows.
✓ 3s completed at 10:39 PM

```
[ ] images.shape
```

```
(77, 224, 224, 3)
```

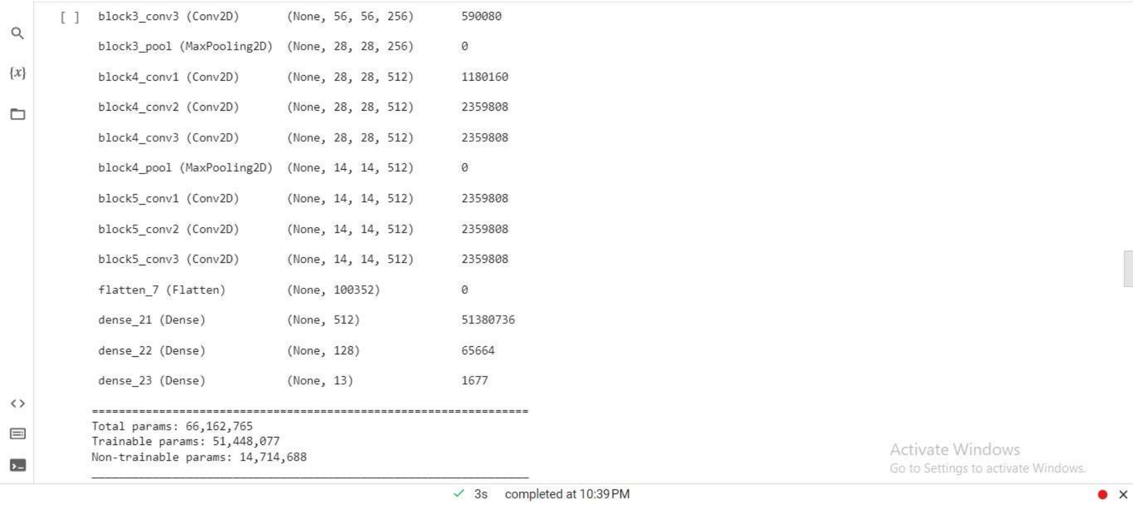
```
[ ] # change the model here for alternatives

    model = VGG16(weights="imagenet")
    for i in model.layers:
        i.trainable =  False
```

```
len(model.layers)
```

```
23
```

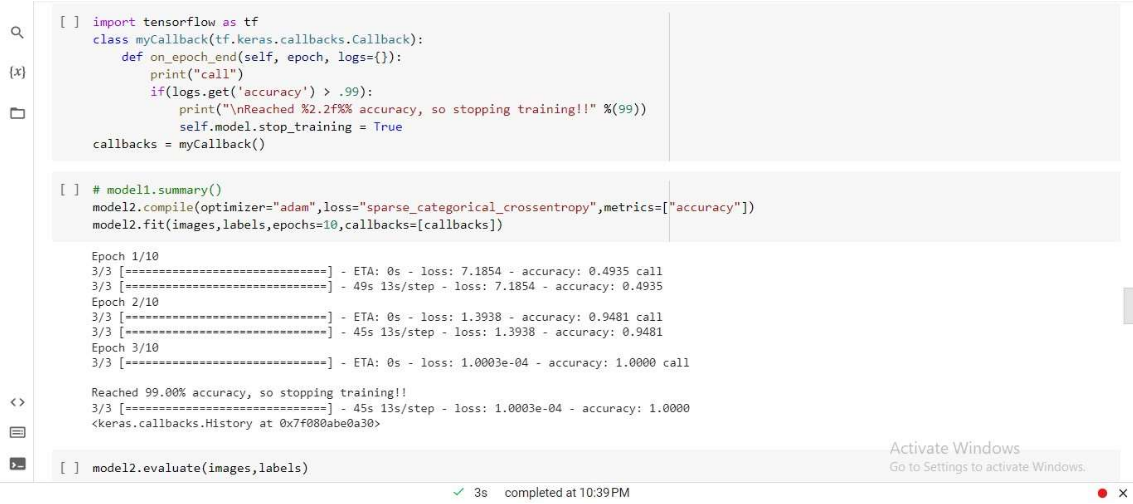
```
[ ] model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_8 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
```



```
block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

flatten_7 (Flatten)         (None, 100352)            0

dense_21 (Dense)            (None, 512)               51380736

dense_22 (Dense)            (None, 128)               65664

dense_23 (Dense)            (None, 13)                1677

=================================================================
Total params: 66,162,765
Trainable params: 51,448,077
Non-trainable params: 14,714,688
_________________________________________________________________
```

[ ]
Activate Windows
Go to Settings to activate Windows.
✓ 3s completed at 10:39 PM



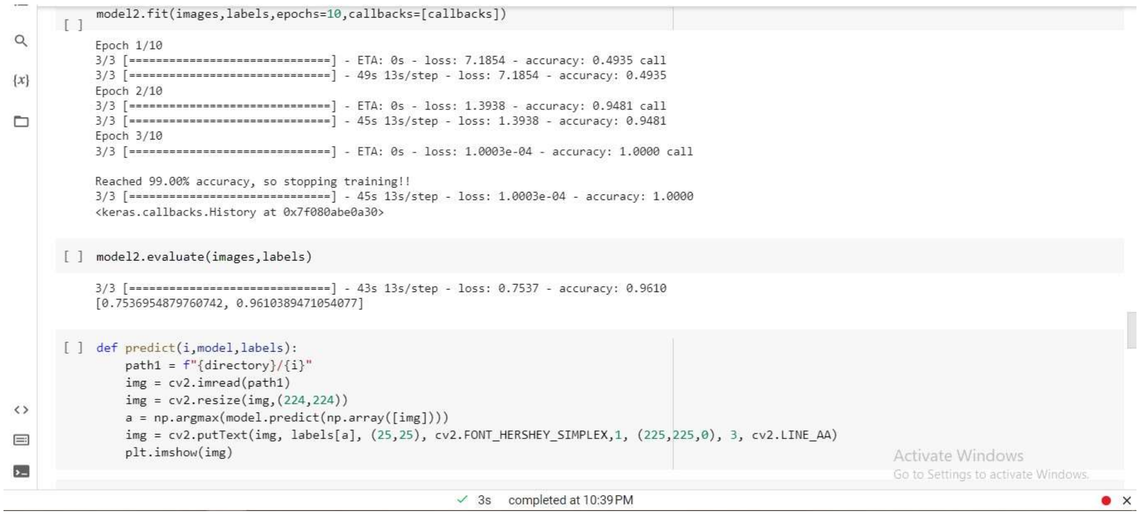
```
import tensorflow as tf
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        print("call")
        if(logs.get('accuracy') > .99):
            print("\nReached %2.2f%% accuracy, so stopping training!!" %(99))
            self.model.stop_training = True
callbacks = myCallback()
```

```
# model1.summary()
model2.compile(optimizer="adam",loss="sparse_categorical_crossentropy",metrics=["accuracy"])
model2.fit(images,labels,epochs=10,callbacks=[callbacks])
```

```
Epoch 1/10
3/3 [==============================] - ETA: 0s - loss: 7.1854 - accuracy: 0.4935 call
3/3 [==============================] - 49s 13s/step - loss: 7.1854 - accuracy: 0.4935
Epoch 2/10
3/3 [==============================] - ETA: 0s - loss: 1.3938 - accuracy: 0.9481 call
3/3 [==============================] - 45s 13s/step - loss: 1.3938 - accuracy: 0.9481
Epoch 3/10
3/3 [==============================] - ETA: 0s - loss: 1.0003e-04 - accuracy: 1.0000 call

Reached 99.00% accuracy, so stopping training!!
3/3 [==============================] - 45s 13s/step - loss: 1.0003e-04 - accuracy: 1.0000
<keras.callbacks.History at 0x7f080abe0a30>
```

```
model2.evaluate(images,labels)
```

[ ]
[ ]
[ ]
Activate Windows
Go to Settings to activate Windows.
✓ 3s completed at 10:39 PM



```
model2.fit(images,labels,epochs=10,callbacks=[callbacks])
```

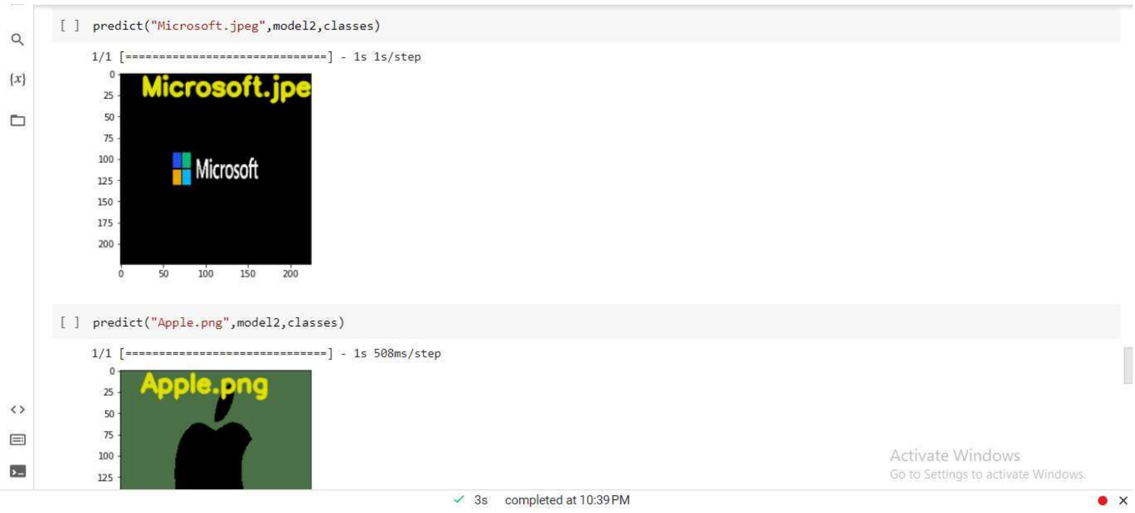
```
Epoch 1/10
3/3 [==============================] - ETA: 0s - loss: 7.1854 - accuracy: 0.4935 call
3/3 [==============================] - 49s 13s/step - loss: 7.1854 - accuracy: 0.4935
Epoch 2/10
3/3 [==============================] - ETA: 0s - loss: 1.3938 - accuracy: 0.9481 call
3/3 [==============================] - 45s 13s/step - loss: 1.3938 - accuracy: 0.9481
Epoch 3/10
3/3 [==============================] - ETA: 0s - loss: 1.0003e-04 - accuracy: 1.0000 call

Reached 99.00% accuracy, so stopping training!!
3/3 [==============================] - 45s 13s/step - loss: 1.0003e-04 - accuracy: 1.0000
<keras.callbacks.History at 0x7f080abe0a30>
```

```
model2.evaluate(images,labels)
```

```
3/3 [==============================] - 43s 13s/step - loss: 0.7537 - accuracy: 0.9610
[0.7536954879760742, 0.9610389471054077]
```

```
def predict(i,model,labels):
    path1 = f"{directory}/{i}"
    img = cv2.imread(path1)
    img = cv2.resize(img,(224,224))
    a = np.argmax(model.predict(np.array([img])))
    img = cv2.putText(img, labels[a], (25,25), cv2.FONT_HERSHEY_SIMPLEX,1, (225,225,0), 3, cv2.LINE_AA)
    plt.imshow(img)
```

[ ]
[ ]
[ ]
Activate Windows
Go to Settings to activate Windows.
✓ 3s completed at 10:39 PM



```
[ ] predict("Microsoft.jpeg",model2,classes)
```

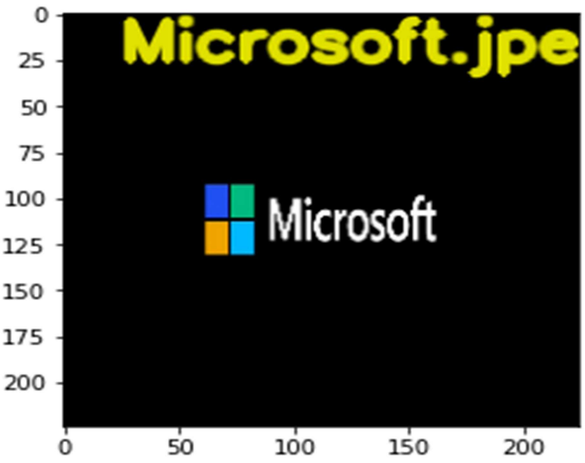
```
1/1 [==============================] - 1s 1s/step
```

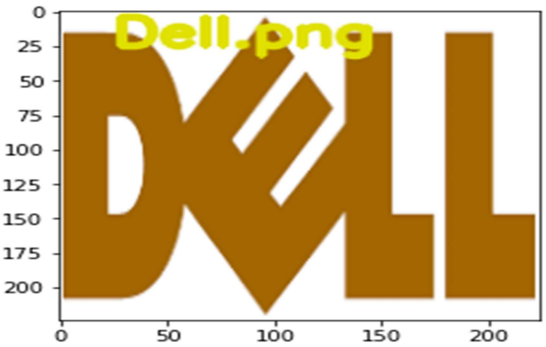
```
[ ] predict("Apple.png",model2,classes)
```

```
1/1 [==============================] - 1s 508ms/step
```

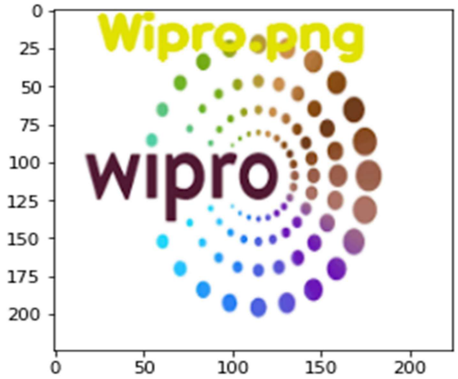
Microsoft.jpe
Microsoft
Apple.png
Activate Windows
Go to Settings to activate Windows.
✓ 3s completed at 10:39 PM

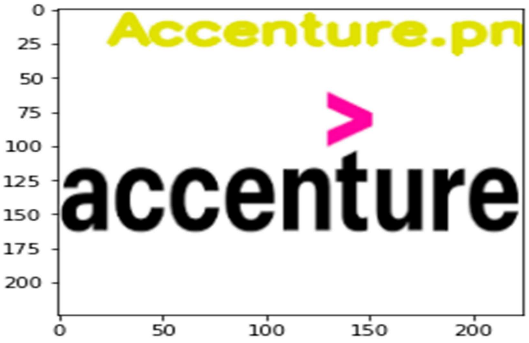
OUTPUT:





Apple.png
0
25
50
75
100
125
150
175
200
0
50
100
150
200





RESULTS:

Thus, the Images (LOGO) of the IT or MNCs are successfully recognized using convolution neural network.

Therefore, Accuracy = 0.96