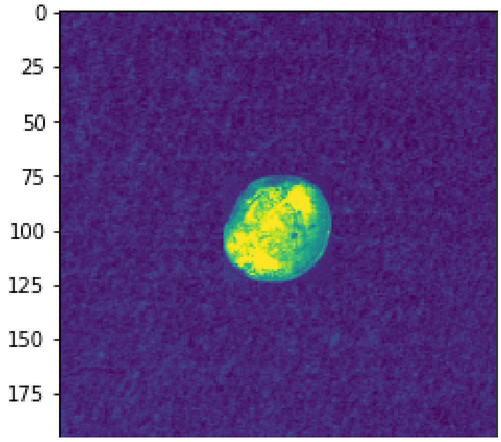
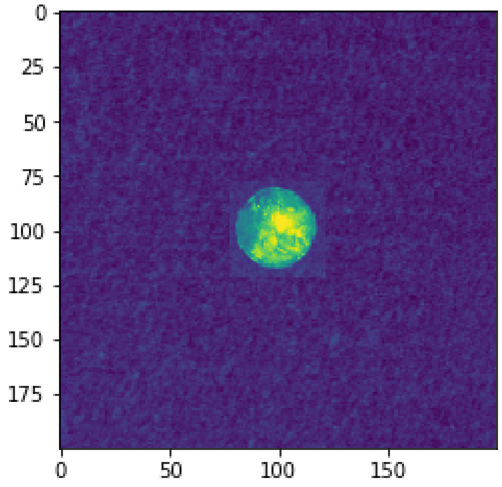
```
import tensorflow as tf
import random as rn
import os,cv2
import numpy as np

os.environ['PYTHONHASHSEED'] = '0'
# Setting the seed for numpy-generated random numbers
np.random.seed(37)
# Setting the seed for python random numbers
rn.seed(1254)
# Setting the graph-level random seed.
tf.set_random_seed(89)
from keras import backend as K
session_conf = tf.ConfigProto(intra_op_parallelism_threads=1,inter_op_parallelism_threads=

#Force Tensorflow to use a single thread
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)

import glob

from sklearn.utils import shuffle

from sklearn.model_selection import train_test_split

import re

from keras.utils import np_utils

import matplotlib.pyplot as plt
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from keras.preprocessing.image import ImageDataGenerator



def gen_image(arr):
    two_d = (np.reshape(arr, (200, 200)) * 255).astype(np.uint8)
    plt.imshow(two_d, interpolation='nearest')
    return plt

def unique(list1):

    # insert the list to the set
    list_set = set(list1)
    # convert the set to the list
    unique_list = (list(list_set))
    for x in unique_list:
        print(x)

#from sklearn.cross_validation import train_test_split

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

PATH = os.getcwd()
# Define data path
data_path = '/content/drive/My Drive/Colab Notebooks/Coin_Recognition_Assignment_Dataset_f

data_dir_list = (os.listdir(data_path)) # os.listdir(data_path)



img_rows=128
img_cols=128
num_channel=1
num_epoch=20

# Define the number of classes
num_classes = 2

labels_name={'COIN':0, 'SCRAP':1}
img_data_list=[]
labels_list = []

for dataset in data_dir_list:
    img_list = glob.glob(data_path+'/'+ dataset +'/*.jpg')

    label = labels_name[dataset] # label is generated as the library updated above
    for img in img_list:
        input_img=cv2.imread(img,1 )
        input_img=cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
        input_img_resize=cv2.resize(input_img,(200,200))
        img_data_list.append(input_img_resize)
        labels_list.append(label)

#print(unique(labels_list))
img_data = np.array(img_data_list)
img_data = img_data.astype('float32')

labels = np.array(labels_list)

#print(unique(labels))
print(np.unique(labels,return_counts=True))
Y = np_utils.to_categorical(labels, num_classes)



#Shuffle the dataset
x,y = shuffle(img_data,Y, random_state=2)



X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2) #

#Normalization of the data
X_train = X_train / 255
X_test = X_test / 255

Nv = X_train.shape[0]
Nv_test = X_test.shape[0]
```

```
#reshape data to fit model
X_train = X_train.reshape(int(Nv),200,200,1)
X_test = X_test.reshape(int(Nv_test),200,200,1)

#create model
model = Sequential()
#add model layers

model.add(Conv2D(64, kernel_size=3,strides=(2,2), activation='relu', input_shape=(200,200,
        # 64 are the number of filters, kernel size is the size of the filters example 3*3
#model.add(Conv2D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

# 8. Compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# 9. Fit model on training data
model.fit(X_train, y_train, batch_size=32, nb_epoch=20, verbose=1, shuffle=False, validati
'''data_generator = ImageDataGenerator(vertical_flip=True,horizontal_flip=True)
data_generator.fit(X_train)
model.fit_generator(data_generator.flow(X_train, y_train, batch_size=32),steps_per_epoch=l


#TESTING

# Define data path
data_path = '/content/drive/My Drive/Colab Notebooks/Coin_Recognition_Assignment_Dataset_f
data_dir_list = (os.listdir(data_path)) # os.listdir(data_path)

# Define the number of classes
num_classes = 2

labels_name={'COIN':0, 'SCRAP':1}
img_data_list=[]
labels_list = []

for dataset in data_dir_list:
    img_list = glob.glob(data_path+'/'+ dataset +'/*.jpg')

    label = labels_name[dataset] # label is generated as the library updated above
    for img in img_list:
        input_img=cv2.imread(img,1 )
        input_img=cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
        input_img_resize=cv2.resize(input_img,(200,200))
        img_data_list.append(input_img_resize)
        labels_list.append(label)

        #print(unique(labels_list))
img_data = np.array(img_data_list)
img_data = img_data.astype('float32')
img_data = img_data.astype('float32')

labels = np.array(labels_list)

#print(unique(labels))
print(np.unique(labels,return_counts=True))
Y = np_utils.to_categorical(labels, num_classes)


#Shuffle the dataset
x,y = shuffle(img_data,Y, random_state=2)



#Normalization of the data
X_t = x / 255
y_t=y

Nv_test = X_t.shape[0]

#reshape data to fit model
X_t = X_t.reshape(int(Nv_test),200,200,1)




# 10. Evaluate model on test data
score = model.evaluate(X_t, y_t, verbose=1)

print('Testing accuracy - > ',score[1] * 100)

ytested = model.predict_classes(X_t)
for i in range(10):
  print("The Predicted Testing image is =%s verify below" % ((list(labels_name.keys())[lis
  gen_image(X_t[i]).show() # printing image vs the predicted image below
```

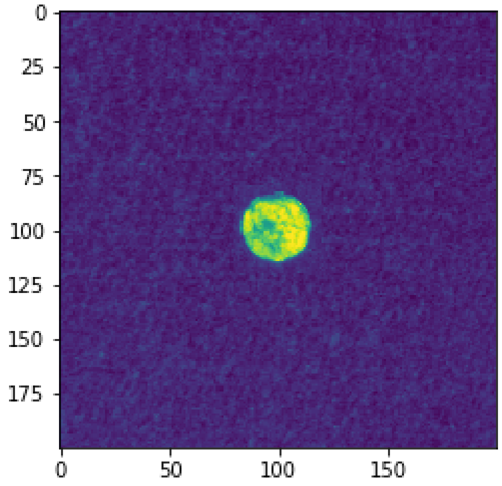
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
(array([0, 1]), array([921, 400]))
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:132: UserWarning: The `n
Train on 1056 samples, validate on 265 samples
Epoch 1/20
1056/1056 [==============================] - 5s 5ms/step - loss: 0.8547 - acc: 0.7140
Epoch 2/20
1056/1056 [==============================] - 1s 608us/step - loss: 0.3931 - acc: 0.84
Epoch 3/20
1056/1056 [==============================] - 1s 605us/step - loss: 0.3614 - acc: 0.86
Epoch 4/20
1056/1056 [==============================] - 1s 599us/step - loss: 0.3330 - acc: 0.86
Epoch 5/20
1056/1056 [==============================] - 1s 615us/step - loss: 0.3235 - acc: 0.87
Epoch 6/20
1056/1056 [==============================] - 1s 599us/step - loss: 0.3002 - acc: 0.88
Epoch 7/20
1056/1056 [==============================] - 1s 602us/step - loss: 0.2998 - acc: 0.88
Epoch 8/20
1056/1056 [==============================] - 1s 611us/step - loss: 0.2970 - acc: 0.88
Epoch 9/20
1056/1056 [==============================] - 1s 593us/step - loss: 0.2738 - acc: 0.89
Epoch 10/20
1056/1056 [==============================] - 1s 620us/step - loss: 0.2597 - acc: 0.89
Epoch 11/20
1056/1056 [==============================] - 1s 603us/step - loss: 0.2497 - acc: 0.90
Epoch 12/20
1056/1056 [==============================] - 1s 602us/step - loss: 0.2287 - acc: 0.91
Epoch 13/20
1056/1056 [==============================] - 1s 592us/step - loss: 0.2058 - acc: 0.91
Epoch 14/20
1056/1056 [==============================] - 1s 623us/step - loss: 0.1909 - acc: 0.91
Epoch 15/20
1056/1056 [==============================] - 1s 590us/step - loss: 0.1890 - acc: 0.92
Epoch 16/20
1056/1056 [==============================] - 1s 604us/step - loss: 0.1746 - acc: 0.92
Epoch 17/20
1056/1056 [==============================] - 1s 617us/step - loss: 0.1584 - acc: 0.93
Epoch 18/20
1056/1056 [==============================] - 1s 611us/step - loss: 0.1475 - acc: 0.94
Epoch 19/20
1056/1056 [==============================] - 1s 611us/step - loss: 0.1476 - acc: 0.94
Epoch 20/20
1056/1056 [==============================] - 1s 592us/step - loss: 0.1338 - acc: 0.95
(array([0, 1]), array([276, 120]))
396/396 [==============================] - 0s 199us/step
Testing accuracy - >  97.72727278747944
The Predicted Testing image is =COIN verify below
```



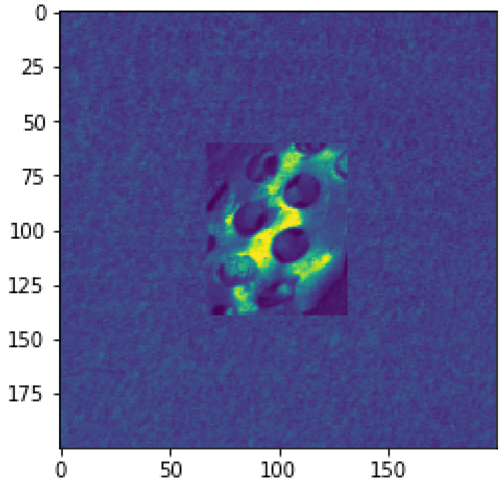
```
The Predicted Testing image is =COIN verify below
```



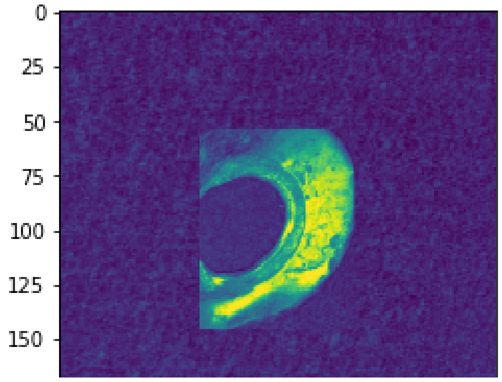
```
The Predicted Testing image is =COIN verify below
```

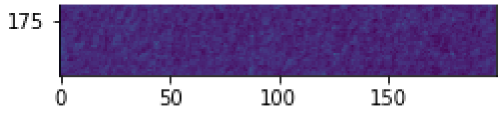


```
The Predicted Testing image is =SCRAP verify below
```

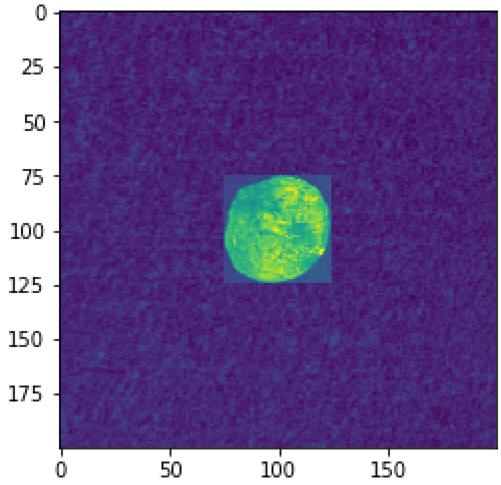


```
The Predicted Testing image is =SCRAP verify below
```

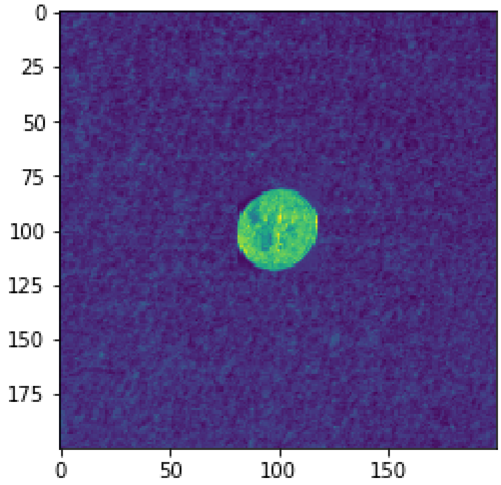




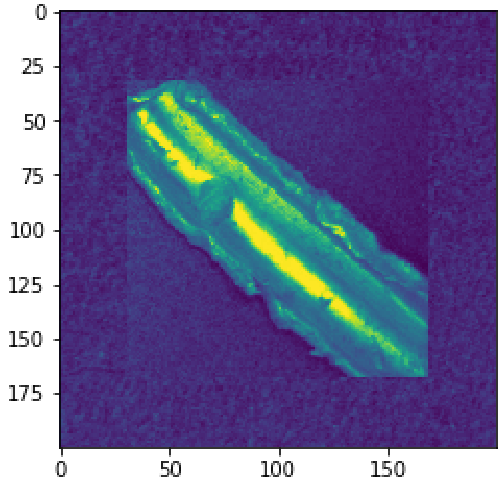
```
The Predicted Testing image is =COIN verify below
```



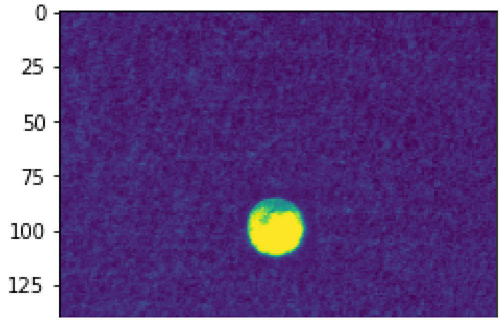
```
The Predicted Testing image is =COIN verify below
```

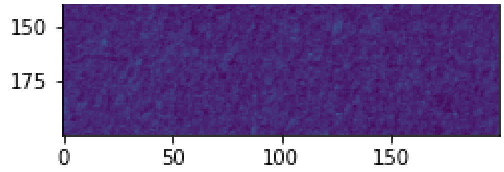


```
The Predicted Testing image is =SCRAP verify below
```

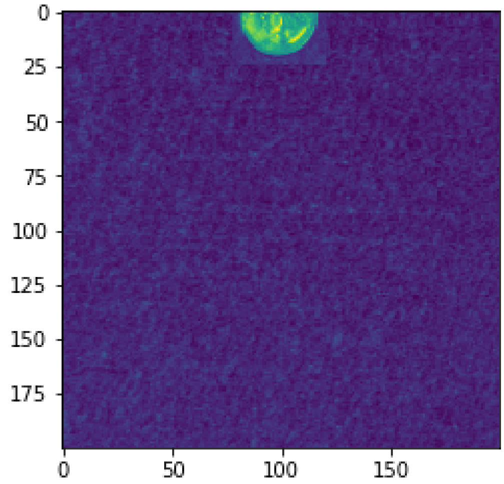


```
The Predicted Testing image is =COIN verify below
```





```
The Predicted Testing image is =COIN verify below
```


0
25
50
75
100
125
150
175
0
50
100
150