Assignment 2

The Question provides us with a set of images and csv file giving description of those images. The images are of seven categories out of which we are using 3 categories: Network, Spheroidite , Pearilite. The first 100 images of each category are taken as Training set and rest of the images are taken as test set.

We use a pre-trained VGG16 to extract features from the images and use Non Linear SVM to classify those images. We create four classifiers in total

- SVM on Network vs Pearlite
- SVM on Pearlite vs Spheroidite
- SVM on Spheroidite vs Network
- Multi-label Classifier (One vs One)

Summary of Code:

- Using model.add layer functionality ,I have created 5 models with model1 having one layer , model2 having two layers and so on
- Then use each model to find the cross validation error which will give me estimate of which model(Max_Pool layer) to work with
- Using the best layer I have predicted test error for four classifiers mentioned above
- For multi classifier I have used OnevsOne classifier which will automatically classify multiple classes

Output

	Cross Validation Error				
Classifier	**Max_Pool Block 1**	**Max_Pool Block 2**	**Max_Pool Block 3**	**Max_Pool Block 4**	**Max_Pool Block 5**
SVMNP(Network & Pearlite)	0.5	0.5	0.485	0.485	0.025
SVMPS(Pearlite & Spheroidite)	0.5	0.5	0.49	0.49	0.035
SVMNS(Network & Spheroidite)	0.495	0.495	0.485	0.485	0.03
Multi-Label Classifier	0.66	0.646	0.65	0.6599	0.0433

Observation:

Max_Pool Layers Cross Validation : The best convolution layer out of the 5 is **"Block-5_Max pool layer"** giving the lowest cross-validation errors on each pairwise classifier as well as in Multi-label classifier

This is because of the following reasons:

- It is clear that the classification can be improved by increasing the depth of CNN increases the predictive capabilities of the model. This is understandable as we increase the depth the max_pool layer output dimensions increases hence providing more features. More features lead to more information for classifying and hence reduces error.
- While the architecture of VGG16 is made to classify more parameters of images as we move from "layer 1" to "layer 5" and hence it is not surprising that layer 5 provides the best cross validation results

Test Error Observations: I use layer 5 max pool output to find test errors for each pairwise classifiers as well as for the multi-label classifier. The observations are as follows

Classifier	Test Error
SVM(Network & Pearlite)	0.0735
SVM(Pearlite & Spheroidite)	0
SVM(Network & Spheroidite)	0.0336
Multi-Label Classifier	0.031

Observation:

- It is clear from the test errors that model is not over-fitting even if we are using more features to train our models
- Secondly it is clearly visible that the pairwise classifier using Pearlite and Spheroidite is performing better with zero test error
- We also see that Multi label classifier has better results that the other two pairwise classifiers

Cose For Assignment 2

```
import keras
from keras.applications.vgg16 import preprocess_input
from keras.models import Sequential
import pandas as pd
from keras.preprocessing import image
import numpy as np
import os
from sklearn import svm
from sklearn.model_selection import cross_val_score


def err(x, y, z):
    E = 0
    for i in range(len(y)):
        if (x[i] != y[i]):
            E = E + 1
        else:
            continue

    R = E / len(x)
    if (z == 0):
        print('Resubstitution Error: ', R)
    else:
        print('Test Error: ', R)

vgg16_model=keras.applications.vgg16.VGG16(weights='imagenet',include_top=False)

#The following code makes the 5 models with model one having one layer and model 5 having 5 layers
model1 = Sequential()
i = 0
for layer in vgg16_model.layers:

    X1 = "block1" in layer.name
    X2 = "block2" in layer.name
    X3 = "block3" in layer.name
    X4 = "block4" in layer.name
    X5 = "block5" in layer.name
    Xf = "flatten" in layer.name
    Xf1 = "fc1" in layer.name
    Xf2 = "fc2" in layer.name
    if (i == 0 or X1 == True):
        model1.add(layer)#model with one layer
        i = i + 1

model2 = Sequential()
i = 0
for layer in vgg16_model.layers:

    X1 = "block1" in layer.name
    X2 = "block2" in layer.name
    X3 = "block3" in layer.name
    X4 = "block4" in layer.name
    X5 = "block5" in layer.name
    Xf = "flatten" in layer.name
    Xf1 = "fc1" in layer.name
    Xf2 = "fc2" in layer.name
    if (i == 0 or X1 == True or X2 == True):
        model2.add(layer)#model with two layers
        i = i + 1

model3 = Sequential()
i = 0
for layer in vgg16_model.layers:

    X1 = "block1" in layer.name
    X2 = "block2" in layer.name
    X3 = "block3" in layer.name
    X4 = "block4" in layer.name
    X5 = "block5" in layer.name
    Xf = "flatten" in layer.name
    Xf1 = "fc1" in layer.name
    Xf2 = "fc2" in layer.name
    if (i == 0 or X1 == True or X2 == True or X3 == True):
        model3.add(layer)#model with three layer
        i = i + 1

model4 = Sequential()
```

```
i = 0
for layer in vgg16_model.layers:

    X1 = "block1" in layer.name
    X2 = "block2" in layer.name
    X3 = "block3" in layer.name
    X4 = "block4" in layer.name
    X5 = "block5" in layer.name
    Xf = "flatten" in layer.name
    Xf1 = "fc1" in layer.name
    Xf2 = "fc2" in layer.name
    if (i == 0 or X1 == True or X2 == True or X3 == True or X4 == True):
        model4.add(layer)#model with four layers
        i = i + 1

model5 = Sequential()
i = 0
for layer in vgg16_model.layers:

    X1 = "block1" in layer.name
    X2 = "block2" in layer.name
    X3 = "block3" in layer.name
    X4 = "block4" in layer.name
    X5 = "block5" in layer.name
    Xf = "flatten" in layer.name
    Xf1 = "fc1" in layer.name
    Xf2 = "fc2" in layer.name
    if (i == 0 or X1 == True or X2 == True or X3 == True or X4 == True or X5 == True):
        model5.add(layer)#model with five layer
        i = i + 1

os.chdir('C:/Users/ACER/PycharmProjects/Next Steps/micrograph')#specify the Image folder path
PATH=os.getcwd()
img_list=os.listdir(PATH)
os.chdir('C:/Users/ACER/PycharmProjects/Next Steps')#specify the csv file path
label_micrograph=pd.read_csv("micrograph.csv")
label_micrograph1=label_micrograph.micrograph_id
label micrograph3=label micrograph.path
result=pd.concat([label_micrograph1, label_micrograph3], axis=1, sort=False)
label micrograph2=label micrograph.primary microconstituent
result=pd.concat([result, label_micrograph2], axis=1, sort=False)
os.chdir('C:/Users/ACER/PycharmProjects/Next Steps/micrograph')#specify the image folder path again

result2 = result.primary_microconstituent

img_list_pearlite = []
img_list_spheroidite = []
img_list_network = []
#Following code will biforcate network ,pearlite,spheroidite images and  load it in VGG16 and extract features
for i in range(len(img_list)):

    if (result2[i] == 'pearlite'):
        x = image.load img(result.path[i])
        x = image.img_to_array(x)
        x = x[0:484, :, :]  # crop the bottom subtitles
        x = np.expand_dims(x, axis=0)
        x = preprocess input(x)
        x = model5.predict(x)
        x = np.mean(x, axis=(0, 1, 2))
        img_list_pearlite.append(x)

    elif (result2[i] == 'network'):
        x = image.load_img(result.path[i])

        x = image.img to array(x)
        x = x[0:484, :, :]  # crop the bottom subtitles
        x = np.expand dims(x, axis=0)
        x = preprocess_input(x)
        x = model5.predict(x)
        x = np.mean(x, axis=(0, 1, 2))
        img list network.append(x)

    elif (result2[i] == 'spheroidite'):
        x = image.load_img(result.path[i])

        x = image.img_to_array(x)
        x = x[0:484, :, :]  # crop the bottom subtitles
        x = np.expand_dims(x, axis=0)
        x = preprocess input(x)
        x = model5.predict(x)
        x = np.mean(x, axis=(0, 1, 2))
```

```
        img_list_spheroidite.append(x)

    else:
        continue


img_list_network_train=img_list_network[0:100]
img_list_network_test=img_list_network[100:]
img_list_pearlite_train=img_list_pearlite[0:100]
img_list_pearlite_test=img_list_pearlite[100:]
img list spheroidite train=img list spheroidite[0:100]
img list spheroidite test=img list spheroidite[100:]

data_trainNP=img_list_network_train+img_list_pearlite_train
data_trainNP=np.array(data_trainNP)
data_trainNP=pd.DataFrame(data_trainNP)
label_train=np.ones((data_trainNP.shape[0]),dtype="int64")
a=len(img_list_network_train)
label_train[0:a]=0#network
label_train[a:]=1#pearlite
label_train=pd.DataFrame({'label':label_train})
data_trainNP=pd.concat([data_trainNP,label_train], axis=1, sort=False)

SVMNP=svm.SVC(kernel='rbf',C=1)
data_train=data_trainNP.drop(columns="label")
SVMNP.fit(data train,data trainNP.label)
# score=cross_val_score(SVMNP,data_train,data_trainNP.label,cv=10)
# 1-score.mean()
#The commented part is used to find cross validation errors

data_test=img_list_network_test+img_list_pearlite_test
data test=np.array(data test)
data_test=pd.DataFrame(data_test)
label_test=np.ones((data_test.shape[0]),dtype="int64")
a=len(img_list_network_test)
label_test[0:a]=0#network
label_test[a:]=1#pearlite
label_test=pd.DataFrame({'label':label_test})
data_test1=pd.concat([data_test,label_test], axis=1, sort=False)

#Training Error
label_pred_re=SVMNP.predict(data_train)
err(data_trainNP.label,label_pred_re,0)
#Testing Data
data_test=data_test1.drop(columns="label")
label_pred_te=SVMNP.predict(data_test)
err(data_test1.label,label_pred_te,1)

data_trainPS=img_list_pearlite_train+img_list_spheroidite_train
data_trainPS=np.array(data_trainPS)
data_trainPS=pd.DataFrame(data_trainPS)

label_train=np.ones((data_trainPS.shape[0]),dtype="int64")
a=len(img_list_pearlite_train)
label_train[0:a]=0#pearlite
label_train[a:]=1#Spheroidite
label_train=pd.DataFrame({'label':label_train})
data_trainPS=pd.concat([data_trainPS,label_train], axis=1, sort=False)

SVMPS=svm.SVC(kernel='rbf',C=1)
data_train=data_trainPS.drop(columns="label")
SVMPS.fit(data_train,data_trainPS.label)
# score=cross_val_score(SVMNP,data_train,data_trainNP.label,cv=10)
# 1-score.mean()
#The commented part is used to find cross validation errors

data test=img list pearlite test+img list spheroidite test
data_test=np.array(data_test)
data_test=pd.DataFrame(data_test)
label_test=np.ones((data_test.shape[0]),dtype="int64")
a=len(img list pearlite test)
print(a)
label_test[0:a]=0#pearlite
label_test[a:]=1#spheroidite
label_test=pd.DataFrame({'label':label_test})
data_test1=pd.concat([data_test,label_test], axis=1, sort=False)

#Testing Data
data_test=data_test1.drop(columns="label")
label pred te=SVMPS.predict(data test)
err(data_test1.label,label_pred_te,1)
```

```
data_trainNS=img_list_network_train+img_list_spheroidite_train
data_trainNS=np.array(data_trainNS)
data_trainNS=pd.DataFrame(data_trainNS)

label_train=np.ones((data_trainNS.shape[0]),dtype="int64")
a=len(img_list_network_train)
label_train[0:a]=0#network
label_train[a:]=1#Spheroidite
label_train=pd.DataFrame({'label':label_train})
data_trainNS=pd.concat([data_trainNS,label_train], axis=1, sort=False)

SVMNS=svm.SVC(kernel='rbf',C=1)
data_train=data_trainNS.drop(columns="label")
SVMNS.fit(data_train,data_trainNS.label)
# score=cross_val_score(SVMNP,data_train,data_trainNP.label,cv=10)
# 1-score.mean()
#The commented part is used to find cross validation errors

data_test=img_list_network_test+img_list_spheroidite_test
data_test=np.array(data_test)
data_test=pd.DataFrame(data_test)
label_test=np.ones((data_test.shape[0]),dtype="int64")
a=len(img list network test)
print(a)
label test[0:a]=0#network
label_test[a:]=1#spheroidite
label_test=pd.DataFrame({'label':label_test})
data_test1=pd.concat([data_test,label_test], axis=1, sort=False)

#Training Error
label pred re=SVMNS.predict(data train)
err(data_trainNS.label,label_pred_re,0)

#Testing Data
data_test=data_test1.drop(columns="label")
label_pred_te=SVMNS.predict(data_test)
err(data test1.label,label pred te,1)

from sklearn.multiclass import OneVsOneClassifier

data_trainNSP=img_list_network_train+img_list_pearlite_train+img_list_spheroidite_train
data_trainNSP=np.array(data_trainNSP)
data_trainNSP=pd.DataFrame(data_trainNSP)
label_train=np.ones((data_trainNSP.shape[0]),dtype="int64")
a=len(img_list_network_train)
b=len(img_list_spheroidite_train)
label_train[0:a]=0#network
label_train[a:a+b]=1#Pearilite
label_train[a+b:]=2#Spheroidite
label_train=pd.DataFrame({'label':label_train})
data_trainNSP=pd.concat([data_trainNSP,label_train], axis=1, sort=False)
#Testing data consisting of 3 types of images
data_test=img_list_network_test+img_list_pearlite_test+img_list_spheroidite_test
data_test=np.array(data_test)
data_test=pd.DataFrame(data_test)
label_test=np.ones((data_test.shape[0]),dtype="int64")
a=len(img list network test)
b=len(img list pearlite test)
c=len(img_list_spheroidite_test)
label_test[0:a]=0#Network
label_test[a:b+a]=1#Pearlite
label_test[a+b:]=2#Spheroidite
label_test=pd.DataFrame({'label':label_test})
data_test1=pd.concat([data_test,label_test], axis=1, sort=False)
#Using One vs One Multi class classifier
data_train=data_trainNSP.drop(columns="label")
MultiNSP=OneVsOneClassifier(svm.SVC(kernel='rbf',C=1,gamma='auto'))
MultiNSP.fit(data_train,data_trainNSP.label)
score=cross_val_score(MultiNSP,data_train,data_trainNSP.label,cv=10)
print('cross validation error is' ,1-score)
score=MultiNSP.score(data test,data test1.label)
print('Error for Multilabel classifier',1-score)
```