ELC 2021(August)

Handwritten: Optical Character Recognition

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Submitted By

Prateek Bansal 101917080
Akshat Gupta 101917081
Bhavesh Sareen 101917085
Lovish Bansal 101917087

```
import io
import os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from scipy import fftpack
from scipy import ndimage
from sklearn import svm, metrics

from sklearn.metrics import classification_report, confusion_matrix
import imageio
import matplotlib.image as mpimage
import cv2
import glob
import h5py
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
import datetime as dt
from six.moves import range


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

     Mounted at /content/drive


import zipfile
import os
zip_ref=zipfile.ZipFile("/content/drive/MyDrive/train_EC.zip")
zip_ref.extractall("/content/drive/MyDrive")
zip_ref.close()
train_path="/content/drive/MyDrive/train_EC"



train_labels=os.listdir(train_path)

train_labels.sort()

print(train_labels)
nb_classes= 5
global_features_train=[]
train_classes=[]



i,j=0,0
k=0
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

```
from matplotlib.pyplot import imread
for training_name in train_labels:
    path=os.path.join(train_path,training_name,'*')
    files=glob.glob(path)
    current_label=training_name
    k=1
    for fl in files:
        image=imread(fl)
        global_feature=np.hstack([image])
        train_classes.append(current_label)
        global_features_train.append(global_feature)

        i+=1
        k+1
    print("[status] processed folder: {}".format(current_label))
    j+=1
print("[status] completed global feature extraction..")
print("[status] feature vector size {}".format(np.array(global_features_train).shape))
print("[status] training labels {}".format(np.array(train_classes).shape))
```

```
[status] processed folder: a
[status] processed folder: b
[status] processed folder: c
[status] processed folder: d
[status] processed folder: e
[status] processed folder: f
[status] processed folder: g
[status] processed folder: h
[status] processed folder: i
[status] processed folder: j
[status] completed global feature extraction..
[status] feature vector size (835, 32, 32)
[status] training labels (835,)
```

```
#labels = (np.arange(nb_classes) == labels[:,None]).astype(np.float32)
targetNames=np.unique(train_classes)
le=LabelEncoder()
target=le.fit_transform(train_classes)
print("[status] training labels encoded...")
```

```
[status] training labels encoded...
```

```
n_samples, nx,ch=np.array(global_features_train).shape
d2_global_features=np.array(global_features_train).reshape((n_samples, nx*ch))
#scaler=MinMaxScaler(feature_range=(0,1))
#rescaled_features=scaler.fit_transform(d2_global_features)
```

```
print("[Status] feature vector normalized...")

print("[Status] target label{}".format(target))
print("[Status] target label shape {}".format(target.shape))
```

```
[Status] feature vector normalized...
[Status] target label[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9]
[Status] target label shape (835,)
```

```
h5f_data=h5py.File('/content/drive/MyDrive/train_EC/dataelc.h5','w')
h5f_data.create_dataset('dataset_1',data=np.array(d2_global_features))

h5f_label=h5py.File('/content/drive/MyDrive/train_EC/labelselc.h5','w')
h5f_label.create_dataset('dataset_1',data=np.array(target))
```

```
<HDF5 dataset "dataset_1": shape (835,), type "<i8">
```

```
h5f_data.close()
h5f_label.close()
```

```
h5f_data=h5py.File('/content/drive/MyDrive/train_EC/dataelc.h5','r')
h5f_label=h5py.File('/content/drive/MyDrive/train_EC/labelselc.h5','r')
```

```
global_features_string_train= h5f_data['dataset_1']
global_labels_string_train=h5f_label['dataset_1']
```

```
global_features_train=np.array(global_features_string_train)
global_labels_train=np.array(global_labels_string_train)



h5f_data.close()
h5f_label.close()


print("[Status] training feature shape: {}".format(global_features_train.shape))

print("[Status] labels shape: {}".format(global_labels_train.shape))
```

```
[Status] training feature shape: (835, 1024)
[Status] labels shape: (835,)
```

```
clf = svm.SVC(kernel='linear')


from sklearn.model_selection import KFold


cv = KFold(n_splits=4, random_state=1, shuffle=True)


scores = cross_val_score(clf,global_features_train,global_labels_train , scoring='accuracy',


from numpy import mean
from numpy import std
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Accuracy: 0.796 (0.014)
```

```
from sklearn.model_selection import cross_val_predict
y_pred = cross_val_predict(clf, global_features_train,global_labels_train , cv=10)
conf_mat = confusion_matrix(global_labels_train, y_pred)



print("Confusion matrix:\n%s" % conf_mat)
```

```
Confusion matrix:
[[81  2  0  6  1  1  3  6  0  0]
 [11 78  0  5  0  2  1  2  1  0]
 [ 0  1 93  2  2  1  0  0  0  1]
 [ 4  7  4 80  0  0  1  0  3  0]
 [ 0  0  5  1 92  1  0  1  0  0]
 [ 2  3  1  1  1 40  0  0  4  2]
 [ 2  4  2  2  1  3 84  0  2  0]
 [25  6  2  1  6  0  2 58  0  0]
 [ 0  0  3  1  0  3  1  0 14  6]
 [ 1  0  3  4  1  3  0  0  7 35]]
```

```
for i in range(2,11):
  cv = KFold(n_splits=i, random_state=1, shuffle=True)
  scores = cross_val_score(clf,global_features_train,global_labels_train , scoring='accuracy'
  print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
  y_pred = cross_val_predict(clf, global_features_train,global_labels_train , cv=10)
  conf_mat = confusion_matrix(global_labels_train, y_pred)
  print("Confusion matrix:\n%s" % conf_mat)
```

```
Accuracy: 0.780 (0.015)
Confusion matrix:
[[81  2  0  6  1  1  3  6  0  0]
 [11 78  0  5  0  2  1  2  1  0]
 [ 0  1 93  2  2  1  0  0  0  1]
 [ 4  7  4 80  0  0  1  0  3  0]
 [ 0  0  5  1 92  1  0  1  0  0]
 [ 2  3  1  1  1 40  0  0  4  2]
 [ 2  4  2  2  1  3 84  0  2  0]
 [25  6  2  1  6  0  2 58  0  0]
 [ 0  0  3  1  0  3  1  0 14  6]
 [ 1  0  3  4  1  3  0  0  7 35]]
Accuracy: 0.804 (0.023)
Confusion matrix:
[[81  2  0  6  1  1  3  6  0  0]
 [11 78  0  5  0  2  1  2  1  0]
 [ 0  1 93  2  2  1  0  0  0  1]
 [ 4  7  4 80  0  0  1  0  3  0]
 [ 0  0  5  1 92  1  0  1  0  0]
 [ 2  3  1  1  1 40  0  0  4  2]
 [ 2  4  2  2  1  3 84  0  2  0]
 [25  6  2  1  6  0  2 58  0  0]
 [ 0  0  3  1  0  3  1  0 14  6]
 [ 1  0  3  4  1  3  0  0  7 35]]
Accuracy: 0.796 (0.014)
Confusion matrix:
[[81  2  0  6  1  1  3  6  0  0]
 [11 78  0  5  0  2  1  2  1  0]
 [ 0  1 93  2  2  1  0  0  0  1]
 [ 4  7  4 80  0  0  1  0  3  0]
 [ 0  0  5  1 92  1  0  1  0  0]
 [ 2  3  1  1  1 40  0  0  4  2]
 [ 2  4  2  2  1  3 84  0  2  0]
 [25  6  2  1  6  0  2 58  0  0]
 [ 0  0  3  1  0  3  1  0 14  6]
 [ 1  0  3  4  1  3  0  0  7 35]]
Accuracy: 0.810 (0.040)
Confusion matrix:
[[81  2  0  6  1  1  3  6  0  0]
 [11 78  0  5  0  2  1  2  1  0]
 [ 0  1 93  2  2  1  0  0  0  1]
 [ 4  7  4 80  0  0  1  0  3  0]
 [ 0  0  5  1 92  1  0  1  0  0]
 [ 2  3  1  1  1 40  0  0  4  2]
 [ 2  4  2  2  1  3 84  0  2  0]
 [25  6  2  1  6  0  2 58  0  0]
 [ 0  0  3  1  0  3  1  0 14  6]
```

```
 [ 1  0  3  4  1  3  0  0  7 35]]
Accuracy: 0.802 (0.042)
Confusion matrix:
[[81  2  0  6  1  1  3  6  0  0]
 [11 78  0  5  0  2  1  2  1  0]
 [ 0  1 93  2  2  1  0  0  0  1]
 [ 4  7  4 80  0  0  1  0  3  0]
 [ 0  0  5  1 92  1  0  1  0  0]
 [ 2  3  1  1  1 40  0  0  4  2]
 [ 2  4  2  2  1  3 84  0  2  0]
 [25  6  2  1  6  0  2 58  0  0]
 [ 0  0  3  1  0  3  1  0 14  6]
```