Untitled1

October 31, 2019

[1]:
```
import os,sys
import pandas as pd
from sklearn.model_selection import train_test_split
import scipy.io
import numpy as np
from scipy.spatial.distance import pdist
import time
import math
import xgboost as xgb
from xgboost.sklearn import XGBClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
import keras
from keras.utils import to_categorical
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
```

```
Using TensorFlow backend.
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:518: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:519: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
```

```
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:520: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:545: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:550: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

[2]:
```
from keras.layers import Dense, Activation, Flatten, Input, Dropout
from keras.layers import BatchNormalization
from keras.models import Model
from keras import initializers
from keras.optimizers import Adam
```

[3]:
```
# change the root to your own path
root = sys.path[0]
train_dir =  os.path.join(root,  '../data/train_set')
train_image_dir =  os.path.join(train_dir, 'images')
train_pt_dir =  os.path.join(train_dir, 'points' )
train_label_path =  os.path.join(train_dir,  "label.csv")
```

[4]:
```
info = pd.read_csv(train_label_path)
train_idx, test_idx = train_test_split(info['Index'], test_size=0.2,␣
↪random_state=123)

# read mat file and store coordinates in mat
m = []
for idx in info['Index']:
    file = "%04d.mat"%(idx)
    m.append( scipy.io.loadmat( os.path.join( train_pt_dir, file ) ))

mat = [x[[i for i in x.keys() if not i in ['__header__', '__version__',␣
↪'__globals__']][0]] for x in m]
```

0.0.1 train test split

[5]:
```
train_idx, test_idx = train_test_split(info['Index'], test_size=0.2,␣
↪random_state=123)
```

[16]:
```
train_mat = [ mat[i-1] for i in train_idx ]
test_mat = [ mat[i-1] for i in test_idx ]
```

[22]:
```
train_labels = info.emotion_idx[train_idx-1]
test_labels = info.emotion_idx[test_idx-1]
```

[29]:
```
train_label_cat = to_categorical(train_labels)
train_label_cat= train_label_cat[:,1:]
test_label_cat = to_categorical(test_labels)
test_label_cat= test_label_cat[:,1:]
```

0.0.2 feature extraction

[18]:
```
# method 1   pairwise_dist_cal
#def pairwise_dist_cal(xy_cord):
    #p_dist =[]
    #for i in range(xy_cord.shape[0]):
        #for j in range(i+1,xy_cord.shape[0]):
```

```
                # p_dist.append( abs(round(xy_cord[i,0]) - round(xy_cord[j,0] ))␣
 ↪)
                # p_dist.append( abs(round(xy_cord[i,1]) - round(xy_cord[j,1] )␣
 ↪) )
    #return p_dist



#### updated methods with selected fiducial points; this reduce 78 poins to 50
'''
feature selection : 1. remove  points P64 - 70 and P72 - 78
                    2. remove P51,53,55,57,58,60, 61, 63
                    3. calculate midpints between  upper and lower eyebrow lines␣
 ↪and replace P20-22,P24-16 with midpoints
So there is in total 50 points left, which gives 50*49 pairwise distance
'''

def pairwise_dist_cal_updt(mt):
    t0 = time.time()
    p_dist_updt =np.zeros([len(mt),1225,2])
    n = len(mt)
    for k in range(n):
        xy_cord = mt[k]

        xy_cord_cpy  = xy_cord

        #  eye_brow midpoint
        to_add_brl = (xy_cord_cpy[19:22]+ xy_cord_cpy[23:26])/2
        to_add_brr = (xy_cord_cpy[27:30]+ xy_cord_cpy[31:34])/2

        # index to remove
        rm_idx = np.append(np.arange(63,70), np.arange(71,78))
        rm_idx = np.append(rm_idx,np.arange(50,57,2) )
        rm_idx = np.append(rm_idx, [57,59,60,62])
        rm_idx = np.append(rm_idx, np.arange(19,22))
        rm_idx = np.append(rm_idx, np.arange(23,26))
        rm_idx = np.append(rm_idx, np.arange(27,30))
        rm_idx = np.append(rm_idx, np.arange(31,34))
        xy_cord = np.delete(xy_cord, rm_idx, axis = 0)
        xy_cord = np.concatenate((xy_cord,to_add_brl,to_add_brr))


        dist_h = []
        dist_v = []
        for i in range(xy_cord.shape[0]):
            for j in range(i+1,xy_cord.shape[0]):
                dist_h.append( abs(round(xy_cord[i,0]) - round(xy_cord[j,0] )) )
```

```
                dist_v.append( abs(round(xy_cord[i,1]) - round(xy_cord[j,1] ) )␣
↪)
        p_dist_updt[k,:,0]= dist_h
        p_dist_updt[k,:,1]= dist_v


    print("feature constructions takes %s seconds" % (time.time() - t0))
    return p_dist_updt.reshape([n,2450])
```

[19]:
```
print("training: ")
train_data = pairwise_dist_cal_updt(train_mat[0:])
```

```
training:
training and testing feature constructions 18.035425901412964 seconds
```

[21]:
```
print("testing: ")
test_data = pairwise_dist_cal_updt(test_mat[0:])
```

```
testing:
training and testing feature constructions 4.777792930603027 seconds
```

[ ]:
```
accuracy_score(pred_xgb_2,test_labels_xgb )
```

1 XGB model

[55]:
```
from xgboost.sklearn import XGBClassifier
```

[58]:
```
train_labels_xgb = [ x  - 1 for x in train_labels ]
test_labels_xgb = [ x  - 1 for x in test_labels ]
```

[60]:
```
start_time=time.time()

xgb = XGBClassifier(
 learning_rate =0.1,
 n_estimators= 200,
 max_depth=5,
 min_child_weight=1,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'multi:softmax',  # for multi-labels classification
 num_class = 22,
 scale_pos_weight=1,
 seed=123)
```

```
xgb.fit(train_data, train_labels_xgb ,eval_metric='auc')
print("training  model takes %s seconds" % round((time.time() - start_time),3))
```

```
training  model takes 1074.819 seconds
```

[65]:
```
start_time = time.time()
pred_xgb = xgb.predict(test_data)
print("testing model takes %s seconds" % round((time.time() - start_time),3))
```

```
testing model takes 0.71 seconds
```

[68]:
```
acc_xgb = accuracy_score(pred_xgb,test_labels_xgb )
print("Test accuracy is %s percent" %(acc_xgb*100))
```

```
Test accuracy is 47.4 percent
```

Try other leaner rate and n_trees

[96]:
```
xgb_2 = XGBClassifier(
 learning_rate =0.01,
 n_estimators= 500,
 max_depth= 3 ,
 min_child_weight=1,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'multi:softmax',  # for multi-labels classification
 num_class = 22,
 scale_pos_weight=1,
 seed=123)
```

[97]:
```
xgb_2.fit(train_data, train_labels_xgb ,eval_metric='auc')
```

[97]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.8, gamma=0,
              learning_rate=0.01, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=500, n_jobs=1,
              nthread=None, num_class=22, objective='multi:softprob',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
              seed=123, silent=None, subsample=0.8, verbosity=1)
```

[100]:
```
start_time = time.time()
pred_xgb_2 = xgb_2.predict(test_data)
print("testing model takes %s seconds" % round((time.time() - start_time),3))
```

```
testing model takes 1.546 seconds
```

[102]:
```
acc_sc = accuracy_score(pred_xgb_2,test_labels_xgb )
print("Test accuracy is %s percent" %(acc_sc *100))
```

```
Test accuracy is 49.2 percent
```

[ ]:

[ ]:

2 CNN model

[32]:
```
X = train_data
Y = train_label_cat
```

[33]:
```
input_shape = [2450]
input_layer = Input(input_shape)
x = BatchNormalization()(input_layer)
x = Dense(22*20,activation='relu',kernel_initializer=initializers.
 ↪glorot_normal(seed=None))(x)
x = Dropout(0.25)(x)
x = BatchNormalization()(x)
x = Dense(22*10,activation='relu',kernel_initializer=initializers.
 ↪glorot_normal(seed=None))(x)
x = Dropout(0.25)(x)
x = Dense(22*6,activation='relu',kernel_initializer=initializers.
 ↪glorot_normal(seed=None))(x)
x = Dropout(0.25)(x)
x = Dense(22*2,activation='relu',kernel_initializer=initializers.
 ↪glorot_normal(seed=None))(x)
output_layer = Dense(22,activation='softmax',kernel_initializer=initializers.
 ↪glorot_normal(seed=None))(x)
model = Model(input_layer,output_layer)
```

[34]:
```
start_time = time.time()
model.compile(loss='categorical_crossentropy',optimizer = Adam(lr=0.
 ↪001),metrics=['accuracy'])
model_his = model.fit(X,Y,epochs=100)
plt.figure(figsize=[8,6])
plt.plot(model_his.history['accuracy'],'r',linewidth=1.0)
plt.legend(['Training Accuracy'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.title('Accuracy Curves',fontsize=16)
print('lr=0.0001, train complete')
print("training  model takes %s seconds" % round((time.time() - start_time),3))
```

```
WARNING:tensorflow:From /Users/Qiqi/opt/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/100
2000/2000 [==============================] - 3s 2ms/step - loss: 2.9830 -
accuracy: 0.1120
Epoch 2/100
2000/2000 [==============================] - 2s 851us/step - loss: 2.5688 -
accuracy: 0.1970
Epoch 3/100
2000/2000 [==============================] - 2s 850us/step - loss: 2.2683 -
accuracy: 0.2710
Epoch 4/100
2000/2000 [==============================] - 2s 1ms/step - loss: 2.1527 -
accuracy: 0.2925
Epoch 5/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.9536 -
accuracy: 0.3700
Epoch 6/100
2000/2000 [==============================] - 2s 941us/step - loss: 1.8609 -
accuracy: 0.3825
Epoch 7/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.7398 -
accuracy: 0.4190
Epoch 8/100
2000/2000 [==============================] - 3s 1ms/step - loss: 1.6755 -
accuracy: 0.4515
Epoch 9/100
2000/2000 [==============================] - 3s 1ms/step - loss: 1.5764 -
accuracy: 0.4630
Epoch 10/100
2000/2000 [==============================] - 3s 1ms/step - loss: 1.5189 -
accuracy: 0.4905
Epoch 11/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.4929 -
accuracy: 0.4955
Epoch 12/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.4274 -
accuracy: 0.5065
Epoch 13/100
2000/2000 [==============================] - 3s 1ms/step - loss: 1.3726 -
accuracy: 0.5320
Epoch 14/100
2000/2000 [==============================] - 2s 987us/step - loss: 1.3441 -
accuracy: 0.5400
Epoch 15/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.3144 -
```

```
accuracy: 0.5645
Epoch 16/100
2000/2000 [==============================] - 2s 990us/step - loss: 1.2874 -
accuracy: 0.5630
Epoch 17/100
2000/2000 [==============================] - 2s 950us/step - loss: 1.2252 -
accuracy: 0.5710
Epoch 18/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.2199 -
accuracy: 0.5845
Epoch 19/100
2000/2000 [==============================] - 2s 905us/step - loss: 1.2136 -
accuracy: 0.5975
Epoch 20/100
2000/2000 [==============================] - 2s 905us/step - loss: 1.1623 -
accuracy: 0.5965
Epoch 21/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.1260 -
accuracy: 0.6050
Epoch 22/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.1027 -
accuracy: 0.6175
Epoch 23/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.0676 -
accuracy: 0.6295
Epoch 24/100
2000/2000 [==============================] - 2s 986us/step - loss: 1.0240 -
accuracy: 0.6475
Epoch 25/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.0467 -
accuracy: 0.6435
Epoch 26/100
2000/2000 [==============================] - 2s 957us/step - loss: 1.0388 -
accuracy: 0.6475
Epoch 27/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.0205 -
accuracy: 0.6510
Epoch 28/100
2000/2000 [==============================] - 2s 1ms/step - loss: 1.0012 -
accuracy: 0.6585
Epoch 29/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.9623 -
accuracy: 0.6670
Epoch 30/100
2000/2000 [==============================] - 3s 1ms/step - loss: 0.9328 -
accuracy: 0.6755
Epoch 31/100
2000/2000 [==============================] - 3s 1ms/step - loss: 0.9678 -
```

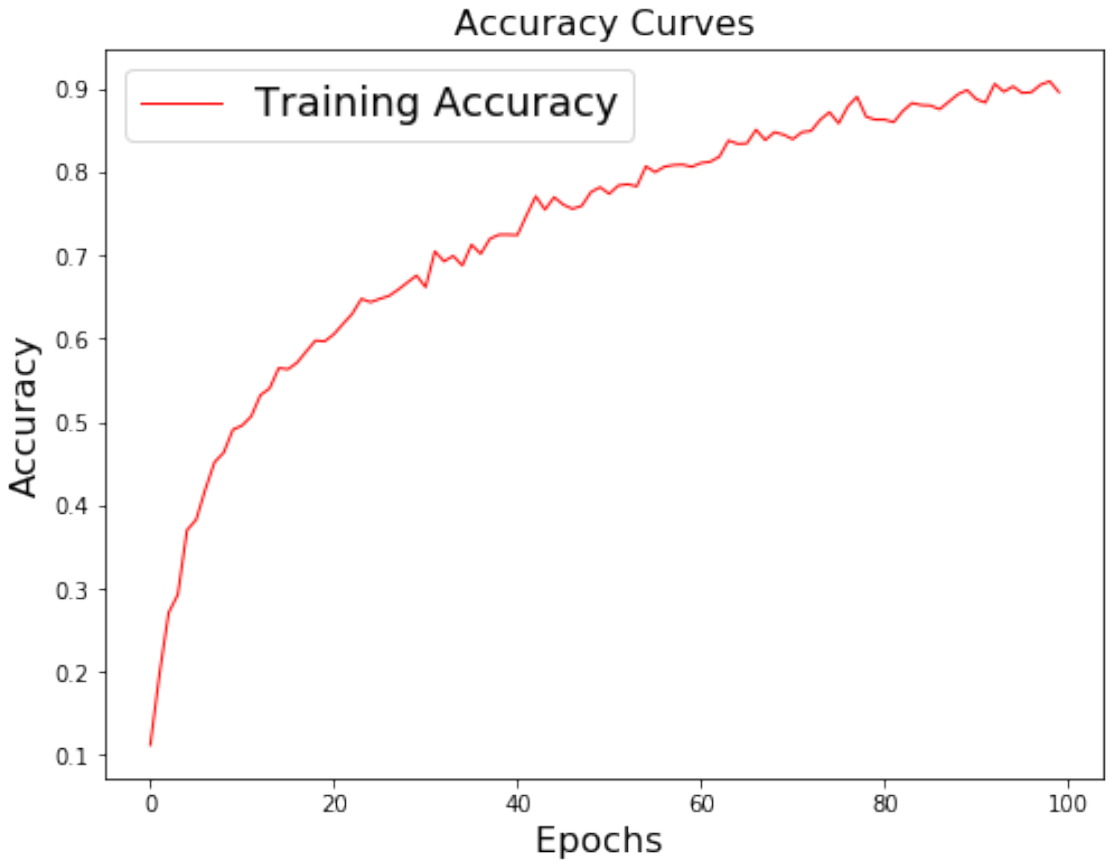
```
accuracy: 0.6615
Epoch 32/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.8481 -
accuracy: 0.7045
Epoch 33/100
2000/2000 [==============================] - 2s 902us/step - loss: 0.8880 -
accuracy: 0.6925
Epoch 34/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.8772 -
accuracy: 0.6990
Epoch 35/100
2000/2000 [==============================] - 2s 952us/step - loss: 0.8944 -
accuracy: 0.6875
Epoch 36/100
2000/2000 [==============================] - 2s 986us/step - loss: 0.8132 -
accuracy: 0.7125
Epoch 37/100
2000/2000 [==============================] - 2s 979us/step - loss: 0.8366 -
accuracy: 0.7015
Epoch 38/100
2000/2000 [==============================] - 2s 825us/step - loss: 0.8271 -
accuracy: 0.7195
Epoch 39/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.8066 -
accuracy: 0.7245
Epoch 40/100
2000/2000 [==============================] - 2s 828us/step - loss: 0.8132 -
accuracy: 0.7245
Epoch 41/100
2000/2000 [==============================] - 2s 902us/step - loss: 0.8100 -
accuracy: 0.7240
Epoch 42/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.7490 -
accuracy: 0.7480
Epoch 43/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.6730 -
accuracy: 0.7705
Epoch 44/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.6979 -
accuracy: 0.7545
Epoch 45/100
2000/2000 [==============================] - 2s 906us/step - loss: 0.6955 -
accuracy: 0.7695
Epoch 46/100
2000/2000 [==============================] - 2s 902us/step - loss: 0.6897 -
accuracy: 0.7605
Epoch 47/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.6854 -
```

```
accuracy: 0.7555
Epoch 48/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.6818 -
accuracy: 0.7590
Epoch 49/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.6678 -
accuracy: 0.7755
Epoch 50/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.6404 -
accuracy: 0.7815
Epoch 51/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.6365 -
accuracy: 0.7735
Epoch 52/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.6262 -
accuracy: 0.7835
Epoch 53/100
2000/2000 [==============================] - 2s 941us/step - loss: 0.6379 -
accuracy: 0.7850
Epoch 54/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.6294 -
accuracy: 0.7825
Epoch 55/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.5886 -
accuracy: 0.8065
Epoch 56/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.5665 -
accuracy: 0.7995
Epoch 57/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.5644 -
accuracy: 0.8060
Epoch 58/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.5541 -
accuracy: 0.8080
Epoch 59/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.5672 -
accuracy: 0.8085
Epoch 60/100
2000/2000 [==============================] - 2s 837us/step - loss: 0.5300 -
accuracy: 0.8060
Epoch 61/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.5212 -
accuracy: 0.8105
Epoch 62/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.5411 -
accuracy: 0.8120
Epoch 63/100
2000/2000 [==============================] - 2s 989us/step - loss: 0.5242 -
```

```
accuracy: 0.8180
Epoch 64/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.4955 -
accuracy: 0.8375
Epoch 65/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.5001 -
accuracy: 0.8335
Epoch 66/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.4576 -
accuracy: 0.8340
Epoch 67/100
2000/2000 [==============================] - 2s 950us/step - loss: 0.4601 -
accuracy: 0.8505
Epoch 68/100
2000/2000 [==============================] - 2s 882us/step - loss: 0.4774 -
accuracy: 0.8380
Epoch 69/100
2000/2000 [==============================] - 2s 973us/step - loss: 0.4717 -
accuracy: 0.8475
Epoch 70/100
2000/2000 [==============================] - 2s 868us/step - loss: 0.4693 -
accuracy: 0.8445
Epoch 71/100
2000/2000 [==============================] - 2s 778us/step - loss: 0.4726 -
accuracy: 0.8390
Epoch 72/100
2000/2000 [==============================] - 1s 737us/step - loss: 0.4146 -
accuracy: 0.8475
Epoch 73/100
2000/2000 [==============================] - 2s 759us/step - loss: 0.4651 -
accuracy: 0.8490
Epoch 74/100
2000/2000 [==============================] - 2s 817us/step - loss: 0.4290 -
accuracy: 0.8625
Epoch 75/100
2000/2000 [==============================] - 1s 711us/step - loss: 0.3979 -
accuracy: 0.8715
Epoch 76/100
2000/2000 [==============================] - 2s 925us/step - loss: 0.4131 -
accuracy: 0.8580
Epoch 77/100
2000/2000 [==============================] - 2s 865us/step - loss: 0.3591 -
accuracy: 0.8785
Epoch 78/100
2000/2000 [==============================] - 2s 906us/step - loss: 0.3487 -
accuracy: 0.8900
Epoch 79/100
2000/2000 [==============================] - 2s 818us/step - loss: 0.3973 -
```

```
accuracy: 0.8660
Epoch 80/100
2000/2000 [==============================] - 2s 831us/step - loss: 0.4145 -
accuracy: 0.8625
Epoch 81/100
2000/2000 [==============================] - 2s 861us/step - loss: 0.3955 -
accuracy: 0.8625
Epoch 82/100
2000/2000 [==============================] - 2s 840us/step - loss: 0.4186 -
accuracy: 0.8595
Epoch 83/100
2000/2000 [==============================] - 2s 806us/step - loss: 0.3804 -
accuracy: 0.8730
Epoch 84/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.3654 -
accuracy: 0.8825
Epoch 85/100
2000/2000 [==============================] - 2s 972us/step - loss: 0.3567 -
accuracy: 0.8800
Epoch 86/100
2000/2000 [==============================] - 2s 932us/step - loss: 0.3916 -
accuracy: 0.8795
Epoch 87/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.3771 -
accuracy: 0.8750
Epoch 88/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.3285 -
accuracy: 0.8840
Epoch 89/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.3277 -
accuracy: 0.8930
Epoch 90/100
2000/2000 [==============================] - 2s 990us/step - loss: 0.3099 -
accuracy: 0.8980
Epoch 91/100
2000/2000 [==============================] - 3s 1ms/step - loss: 0.3361 -
accuracy: 0.8870
Epoch 92/100
2000/2000 [==============================] - 2s 1ms/step - loss: 0.3289 -
accuracy: 0.8830
Epoch 93/100
2000/2000 [==============================] - 2s 933us/step - loss: 0.2941 -
accuracy: 0.9055
Epoch 94/100
2000/2000 [==============================] - 2s 944us/step - loss: 0.3401 -
accuracy: 0.8960
Epoch 95/100
2000/2000 [==============================] - 2s 889us/step - loss: 0.3117 -
```

```
accuracy: 0.9025
Epoch 96/100
2000/2000 [==============================] - 2s 941us/step - loss: 0.3184 -
accuracy: 0.8945
Epoch 97/100
2000/2000 [==============================] - 2s 842us/step - loss: 0.3205 -
accuracy: 0.8955
Epoch 98/100
2000/2000 [==============================] - 2s 877us/step - loss: 0.2833 -
accuracy: 0.9045
Epoch 99/100
2000/2000 [==============================] - 2s 829us/step - loss: 0.2892 -
accuracy: 0.9085
Epoch 100/100
2000/2000 [==============================] - 2s 807us/step - loss: 0.3087 -
accuracy: 0.8955
lr=0.0001, train complete
training  model takes 205.04509115219116 seconds
```



[56]:
```
t0 = time.time()
pred = model.predict(test_data)
pred_lst = []
for i in range(len(pred)):
    arr = pred[i]
    idx = np.argwhere(arr == np.max(arr))
    pred_lst.append(idx[0][0])
tst_labl = np.argmax(test_label_cat, axis=-1)
acc = accuracy_score(pred_lst, tst_labl)
print("Test accuracy is %s percent" %(acc*100))
print("testing model takes %s seconds" % round((time.time() - t0),3))
```

```
Test accuracy is 54.0 percent
testing model takes 0.089 seconds
```

[47]:
```

```