Assignment2_model (4)

May 22, 2022

[ ]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

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

[ ]:
```
import numpy as np
import csv
import pandas as pd
from keras import backend as K
import keras
from keras.models import Sequential, Model,load_model
from keras.callbacks import EarlyStopping,ModelCheckpoint
from google.colab.patches import cv2_imshow
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D,␣
↪BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D,␣
↪GlobalMaxPooling2D,MaxPool2D,Dropout
from keras.preprocessing import image
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
import os
```

Showing if there are GPU device available

[ ]:
```
%tensorflow_version 2.x

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

display if the file is correctly placed in datasets folder in your mounted google drive

[ ]:
```
file_path="/content/gdrive/My Drive/datasets"
file_names=os.listdir(file_path)
print(file_names)
```

```
['emnist-byclass-test.csv', 'emnist-byclass-train.csv']
```

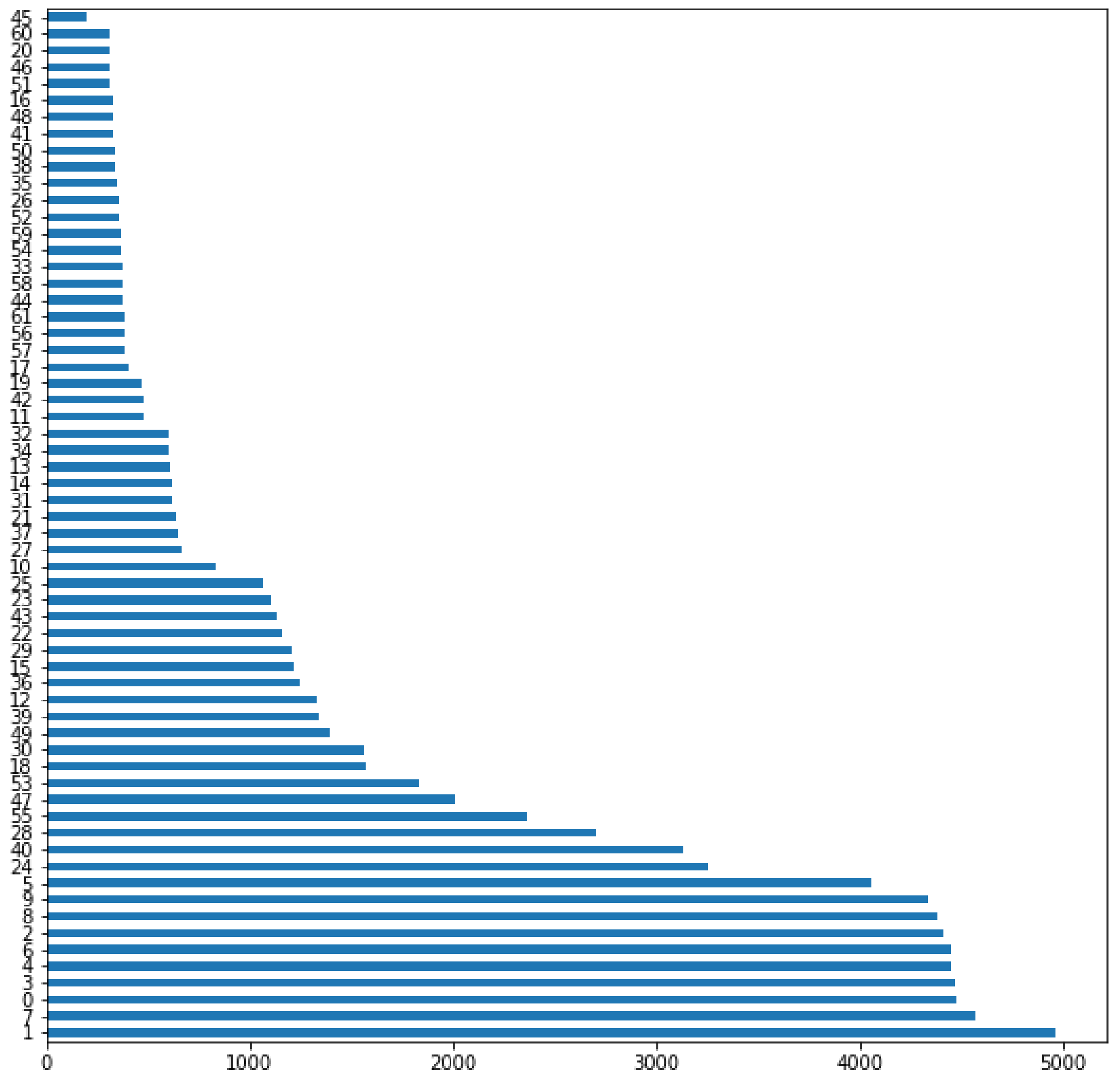
Loading the data first time for the balanced dataset approach

[ ]:
```
train_pd = pd.read_csv('/content/gdrive/My Drive/datasets/emnist-byclass-train.
 ↪csv',delimiter=",", nrows=90000,header=None)
test_pd = pd.read_csv('/content/gdrive/My Drive/datasets/emnist-byclass-test.
 ↪csv',delimiter=",", nrows=70000,header=None)
```

The train dataset distribution overdifferent classes

[ ]:
```
train_pd.iloc[:,0].value_counts().plot(kind='barh',figsize=(10,10))
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f61a7e01b50>
```



Choose the features and the label

[ ]:
```
data_train_feature = train_pd.loc[:, "1":"784"].to_numpy()
# Selecting output lable
data_train_label = train_pd.iloc[:, 0].to_numpy()

data_test_feature = test_pd.loc[:, "1":"784"].to_numpy()
# Selecting output lable
data_test_label = test_pd.iloc[:, 0].to_numpy()
```

Oversampling technique to balance the dataset

[ ]:
```
from imblearn.over_sampling import SMOTE
over_samples = SMOTE(random_state = 1234)
```

```
data_train_f_balance, data_train_l_balance= over_samples.
  ↪fit_resample(data_train_feature,data_train_label)
```

Reshape the data to a 28*28 image like ndarray

[ ]:
```
data_train_f_balance = data_train_f_balance.reshape((data_train_f_balance.
  ↪shape[0], 28, 28))
data_test_feature = data_test_feature.reshape((data_test_feature.shape[0], 28,␣
  ↪28))
```

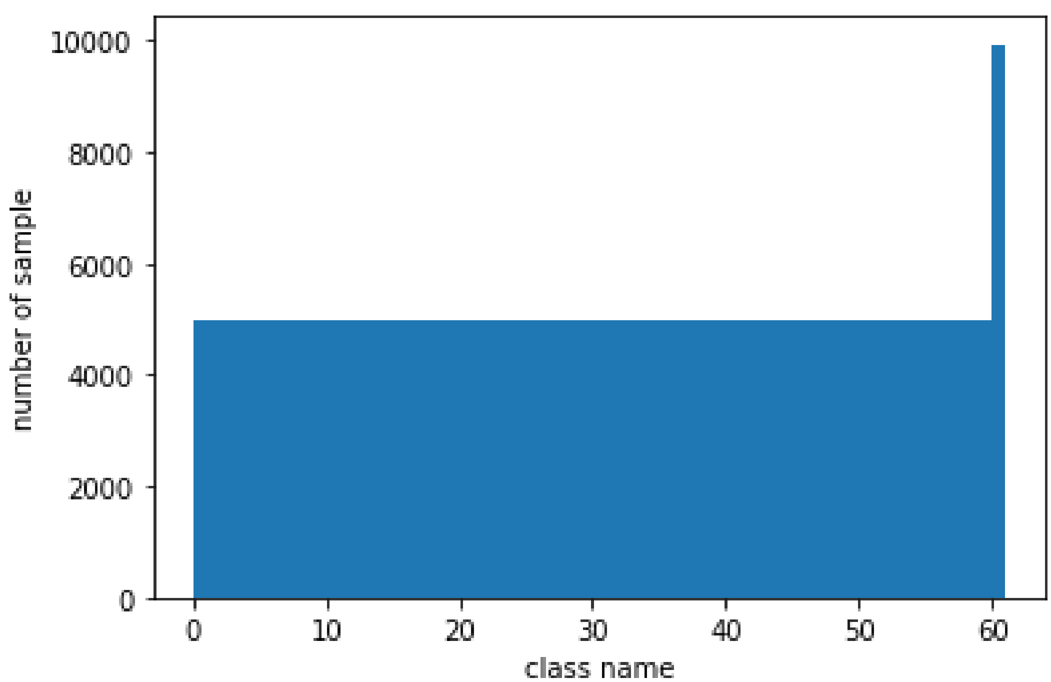
[ ]:
```
print(data_train_f_balance.shape,
data_train_l_balance.shape)
```

```
(308016, 28, 28) (308016,)
```

The train dataset distribution overdifferent classes after oversampling

[ ]:
```
plt.hist(data_train_l_balance, bins=np.arange(data_train_l_balance.min(),␣
  ↪data_train_l_balance.max()+1))
plt.xlabel("class name")
plt.ylabel("number of sample")
```

[ ]:
```
Text(0, 0.5, 'number of sample')
```



CNN model with balanced triaining dataset expanding the dimension to fit the resnet50 model's requirement and normalise the data.

```
[ ]: data_train_f_balance = np.expand_dims(data_train_f_balance, axis=-1)
     # we need 3 channel
     data_train_f_balance = np.repeat(data_train_f_balance, 3, axis=-1)
     # it's always better to normalize
     data_train_f_balance = data_train_f_balance.astype('float32') / 255

     data_test_feature = np.expand_dims(data_test_feature, axis=-1)
     # we need 3 channel
     data_test_feature = np.repeat(data_test_feature, 3, axis=-1)
     # it's always better to normalize
     data_test_feature = data_test_feature.astype('float32') / 255
```

```
[ ]: print(data_train_f_balance.shape,
     data_train_l_balance.shape,
     )
```

```
(308016, 28, 28, 3) (308016,)
```

train test spliting and convert the ndarray to tensor. apply the resize function from tensorflow and make it to fit the minimum requirement for resnet50 model's input image size.

```
[ ]: X_train_b, X_valid_b, y_train_b, y_valid_b =␣
      ↪train_test_split(data_train_f_balance,data_train_l_balance,random_state=0)

     #convert x_train to tensor then resize
     X_train_b = tf.convert_to_tensor(X_train_b)
     X_train_b= tf.image.resize(X_train_b, [32,32])

     #convert x_valid to tensor then resize
     X_valid_b = tf.convert_to_tensor(X_valid_b)
     X_valid_b= tf.image.resize(X_valid_b, [32,32])

     data_test_feature = tf.convert_to_tensor(data_test_feature)
     data_test_feature=tf.image.resize(data_test_feature, [32,32])
```

```
[ ]: print(tf.shape(X_train_b),tf.shape(X_valid_b))
```

```
tf.Tensor([231012     32     32      3], shape=(4,), dtype=int32)
tf.Tensor([77004    32    32     3], shape=(4,), dtype=int32)
```

defining the input and earlystopping of the model.

```
[ ]: from tensorflow.keras.applications import ResNet50
     input_t = Input(shape=(32,32,3))
     callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=2)
```

resenet50 model creation with no pre-train weight

```
[ ]: def creat_model(input_t):
       res_model=ResNet50(include_top=False,weights=None, input_tensor = input_t)
       base_model = Sequential()
       base_model.add(res_model)
       base_model.add(Flatten())
       base_model.add(BatchNormalization())
       base_model.add(Dense(256, activation='relu'))
       base_model.add(Dropout(0.5))
       base_model.add(BatchNormalization())
       base_model.add(Dense(128, activation='relu'))
       base_model.add(Dropout(0.5))
       base_model.add(Dense(62, activation='softmax'))
       return base_model
```

use the same parameter as the imblanaced dataset's turning down below in hyperparameter turning of the ResNet50 on imbalanced dataset,since we want to compare the model on balanced dataset and imbalanced dataset on same parameter.

```
[ ]: base_model_b=creat_model(input_t)
     base_model_b.compile(loss='sparse_categorical_crossentropy',
                     optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                     metrics=['accuracy'])
       #fit the model and train
     history = base_model_b.
       ↪fit(X_train_b,y_train_b,batch_size=128,epochs=15,validation_data=(X_valid_b,y_valid_b),call
     loss, accuracy = base_model_b.evaluate(data_test_feature, data_test_label)
     print(f"Accuracy on test data: {accuracy:.4f}")
```

```
Epoch 1/15
1805/1805 [==============================] - 104s 54ms/step - loss: 1.1590 -
accuracy: 0.6137 - val_loss: 0.7169 - val_accuracy: 0.7199
Epoch 2/15
1805/1805 [==============================] - 98s 54ms/step - loss: 0.6246 -
accuracy: 0.7531 - val_loss: 0.6438 - val_accuracy: 0.7534
Epoch 3/15
1805/1805 [==============================] - 99s 55ms/step - loss: 0.5537 -
accuracy: 0.7811 - val_loss: 0.5149 - val_accuracy: 0.7835
Epoch 4/15
1805/1805 [==============================] - 98s 54ms/step - loss: 0.5144 -
accuracy: 0.7956 - val_loss: 0.6522 - val_accuracy: 0.7545
Epoch 5/15
1805/1805 [==============================] - 98s 54ms/step - loss: 0.4698 -
accuracy: 0.8164 - val_loss: 0.4968 - val_accuracy: 0.8050
Epoch 6/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.4222 -
accuracy: 0.8371 - val_loss: 0.4100 - val_accuracy: 0.8383
Epoch 7/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.3724 -
```

```
accuracy: 0.8588 - val_loss: 0.3420 - val_accuracy: 0.8651
Epoch 8/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.3336 -
accuracy: 0.8763 - val_loss: 0.3735 - val_accuracy: 0.8536
Epoch 9/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.2886 -
accuracy: 0.8955 - val_loss: 0.3531 - val_accuracy: 0.8685
Epoch 10/15
1805/1805 [==============================] - 98s 54ms/step - loss: 0.2556 -
accuracy: 0.9097 - val_loss: 0.2701 - val_accuracy: 0.8976
Epoch 11/15
1805/1805 [==============================] - 98s 54ms/step - loss: 0.2279 -
accuracy: 0.9197 - val_loss: 0.2526 - val_accuracy: 0.9079
Epoch 12/15
1805/1805 [==============================] - 99s 55ms/step - loss: 0.2000 -
accuracy: 0.9297 - val_loss: 0.2515 - val_accuracy: 0.9167
Epoch 13/15
1805/1805 [==============================] - 99s 55ms/step - loss: 0.1853 -
accuracy: 0.9349 - val_loss: 0.2281 - val_accuracy: 0.9224
Epoch 14/15
1805/1805 [==============================] - 98s 54ms/step - loss: 0.1715 -
accuracy: 0.9398 - val_loss: 0.2234 - val_accuracy: 0.9229
Epoch 15/15
1805/1805 [==============================] - 98s 54ms/step - loss: 0.1592 -
accuracy: 0.9447 - val_loss: 0.1914 - val_accuracy: 0.9358
2188/2188 [==============================] - 29s 13ms/step - loss: 0.6807 -
accuracy: 0.8365
Accuracy on test data: 0.8365
```

[ ]:
```
from sklearn.metrics import classification_report
y_pred = base_model_b.predict(data_test_feature, batch_size=128)
y_pred_bool = np.argmax(y_pred, axis=1)
print(classification_report(data_test_label, y_pred_bool))
```

```
              precision    recall  f1-score   support

           0       0.69      0.74      0.71      3461
           1       0.70      0.73      0.71      3860
           2       0.96      0.95      0.95      3559
           3       0.99      0.99      0.99      3577
           4       0.97      0.95      0.96      3411
           5       0.94      0.92      0.93      3115
           6       0.97      0.96      0.96      3426
           7       0.98      0.99      0.98      3768
           8       0.95      0.98      0.97      3375
           9       0.91      0.96      0.94      3390
          10       0.93      0.94      0.93       647
          11       0.94      0.92      0.93       395
```

12	0.80	0.78	0.79	1080
13	0.79	0.91	0.84	446
14	0.93	0.96	0.94	521
15	0.79	0.67	0.72	874
16	0.84	0.87	0.86	254
17	0.88	0.92	0.90	331
18	0.52	0.52	0.52	1237
19	0.85	0.85	0.85	391
20	0.63	0.75	0.69	232
21	0.88	0.86	0.87	476
22	0.79	0.76	0.78	872
23	0.90	0.98	0.94	804
24	0.62	0.58	0.60	2470
25	0.81	0.86	0.83	828
26	0.83	0.89	0.86	257
27	0.89	0.95	0.92	493
28	0.81	0.87	0.84	2099
29	0.90	0.94	0.92	965
30	0.80	0.81	0.81	1209
31	0.64	0.58	0.61	489
32	0.77	0.81	0.79	467
33	0.63	0.72	0.67	266
34	0.68	0.80	0.74	462
35	0.63	0.68	0.65	274
36	0.88	0.91	0.90	955
37	0.79	0.87	0.83	505
38	0.28	0.28	0.28	243
39	0.97	0.96	0.97	1036
40	0.97	0.97	0.97	2462
41	0.32	0.49	0.39	259
42	0.63	0.43	0.51	375
43	0.89	0.96	0.93	900
44	0.40	0.46	0.43	263
45	0.63	0.64	0.63	188
46	0.72	0.63	0.67	283
47	0.37	0.32	0.34	1533
48	0.35	0.40	0.37	269
49	0.95	0.90	0.92	1116
50	0.07	0.02	0.04	283
51	0.46	0.33	0.39	224
52	0.62	0.36	0.45	295
53	0.95	0.94	0.95	1352
54	0.17	0.11	0.13	272
55	0.95	0.92	0.93	1773
56	0.32	0.30	0.31	274
57	0.48	0.57	0.52	302
58	0.65	0.59	0.62	283
59	0.61	0.58	0.59	279

```
          60       0.49      0.35      0.41       231
          61       0.52      0.51      0.52       264

    accuracy                           0.84     70000
   macro avg       0.73      0.73      0.72     70000
weighted avg       0.83      0.84      0.83     70000
```

CNN model with inbalanced training dataset

Load same amount of data as the oversampling dataset numbers

```
[ ]: train_pd_in = pd.read_csv('/content/gdrive/My Drive/datasets/
      ↪emnist-byclass-train.csv',delimiter=",", nrows=308016,header=None)
```

Select features and label

```
[ ]: data_train_feature = train_pd_in.loc[:, "1":"784"].to_numpy()
     # Selecting output lable
     data_train_label = train_pd_in.iloc[:, 0].to_numpy()
```

```
[ ]: data_train_feature = data_train_feature.reshape((data_train_feature.shape[0],␣
      ↪28, 28))
     data_train_feature.shape
```

```
[ ]: (308016, 28, 28)
```

expanding the dimension

```
[ ]: data_train_feature = np.expand_dims(data_train_feature, axis=-1)
     # we need 3 channel
     data_train_feature = np.repeat(data_train_feature, 3, axis=-1)
     # it's always better to normalize
     data_train_feature = data_train_feature.astype('float32') / 255
```

Convert image to tensor then resize them to 32*32

```
[ ]: X_train, X_valid, y_train, y_valid =␣
      ↪train_test_split(data_train_feature,data_train_label,random_state=0)

     #convert x_train to tensor then resize
     X_train = tf.convert_to_tensor(X_train)
     X_train= tf.image.resize(X_train, [32,32])

     #convert x_valid to tensor then resize
     X_valid = tf.convert_to_tensor(X_valid)
     X_valid= tf.image.resize(X_valid, [32,32])
```

```
[ ]: print(tf.shape(X_train),tf.shape(X_valid))
```

```
tf.Tensor([231012     32     32      3], shape=(4,), dtype=int32)
tf.Tensor([77004    32    32     3], shape=(4,), dtype=int32)
```

[ ]:
```
from tensorflow.keras.applications import ResNet50
input_t = Input(shape=(32,32,3))
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
```

Hyperparameter turning on learning rate of Adam optimizer and the batch size

[ ]:
```
learning_rate=[1e-2,1e-3]
batch=[128,256]
result=[]
result_history=[]

def model_turning(lr,bs,base_model):
  base_model.compile(loss='sparse_categorical_crossentropy',
                optimizer=tf.keras.optimizers.Adam(learning_rate=lr),
                metrics=['accuracy'])
  #fit the model and train
  history = base_model.
 ↪fit(X_train,y_train,batch_size=bs,epochs=10,validation_data=(X_valid,y_valid),callbacks=[ca
  loss, accuracy = base_model.evaluate(data_test_feature, data_test_label)
  print(f"Accuracy on test data: {accuracy:.4f}")
  return accuracy,lr,bs,history

for i in range(len(learning_rate)):
  for j in range(len(batch)):
    model=creat_model(input_t)
    print('learning rate:{} batch size: {}'.format(learning_rate[i],batch[j]))
    model_result= model_turning(learning_rate[i],batch[j],model)
    result_history.append(model_result[3])
    result.append((model_result[0],model_result[1],model_result[2]))
```

```
learning rate:0.01 batch size: 128
Epoch 1/10
1805/1805 [==============================] - 108s 57ms/step - loss: 1.1358 -
accuracy: 0.6690 - val_loss: 1.0500 - val_accuracy: 0.6985
Epoch 2/10
1805/1805 [==============================] - 101s 56ms/step - loss: 0.7017 -
accuracy: 0.7835 - val_loss: 0.5910 - val_accuracy: 0.8118
Epoch 3/10
1805/1805 [==============================] - 102s 56ms/step - loss: 0.6558 -
accuracy: 0.7979 - val_loss: 0.9398 - val_accuracy: 0.7469
Epoch 4/10
1805/1805 [==============================] - 101s 56ms/step - loss: 0.6324 -
accuracy: 0.8040 - val_loss: 0.5620 - val_accuracy: 0.8148
Epoch 5/10
1805/1805 [==============================] - 102s 56ms/step - loss: 0.6240 -
```

```
accuracy: 0.8087 - val_loss: 0.5889 - val_accuracy: 0.8216
Epoch 6/10
1805/1805 [==============================] - 102s 56ms/step - loss: 0.6036 -
accuracy: 0.8124 - val_loss: 0.5974 - val_accuracy: 0.8119
Epoch 7/10
1805/1805 [==============================] - 101s 56ms/step - loss: 0.5840 -
accuracy: 0.8179 - val_loss: 0.5113 - val_accuracy: 0.8350
Epoch 8/10
1805/1805 [==============================] - 101s 56ms/step - loss: 0.5743 -
accuracy: 0.8200 - val_loss: 0.4940 - val_accuracy: 0.8337
Epoch 9/10
1805/1805 [==============================] - 101s 56ms/step - loss: 0.5679 -
accuracy: 0.8214 - val_loss: 0.4808 - val_accuracy: 0.8420
Epoch 10/10
1805/1805 [==============================] - 101s 56ms/step - loss: 0.5619 -
accuracy: 0.8236 - val_loss: 0.4901 - val_accuracy: 0.8427
2188/2188 [==============================] - 29s 13ms/step - loss: 0.4882 -
accuracy: 0.8443
Accuracy on test data: 0.8443
learning rate:0.01 batch size: 256
Epoch 1/10
903/903 [==============================] - 75s 77ms/step - loss: 1.1299 -
accuracy: 0.6708 - val_loss: 1.0208 - val_accuracy: 0.7317
Epoch 2/10
903/903 [==============================] - 68s 75ms/step - loss: 0.6362 -
accuracy: 0.7981 - val_loss: 0.5807 - val_accuracy: 0.8131
Epoch 3/10
903/903 [==============================] - 68s 75ms/step - loss: 0.5761 -
accuracy: 0.8164 - val_loss: 0.4917 - val_accuracy: 0.8336
Epoch 4/10
903/903 [==============================] - 68s 75ms/step - loss: 0.5492 -
accuracy: 0.8239 - val_loss: 0.5831 - val_accuracy: 0.8062
Epoch 5/10
903/903 [==============================] - 68s 75ms/step - loss: 0.5363 -
accuracy: 0.8273 - val_loss: 5.7276 - val_accuracy: 0.6807
Epoch 6/10
903/903 [==============================] - 68s 75ms/step - loss: 0.5534 -
accuracy: 0.8224 - val_loss: 0.5351 - val_accuracy: 0.8229
Epoch 7/10
903/903 [==============================] - 68s 75ms/step - loss: 0.5160 -
accuracy: 0.8319 - val_loss: 0.6106 - val_accuracy: 0.8103
Epoch 8/10
903/903 [==============================] - 68s 75ms/step - loss: 0.6048 -
accuracy: 0.8113 - val_loss: 0.5018 - val_accuracy: 0.8329
Epoch 9/10
903/903 [==============================] - 68s 75ms/step - loss: 0.5169 -
accuracy: 0.8319 - val_loss: 0.4708 - val_accuracy: 0.8408
Epoch 10/10
```

```
903/903 [==============================] - 68s 75ms/step - loss: 0.5002 -
accuracy: 0.8361 - val_loss: 0.4972 - val_accuracy: 0.8406
2188/2188 [==============================] - 29s 13ms/step - loss: 0.5015 -
accuracy: 0.8400
Accuracy on test data: 0.8400
learning rate:0.001 batch size: 128
Epoch 1/10
1805/1805 [==============================] - 107s 56ms/step - loss: 1.0305 -
accuracy: 0.7161 - val_loss: 0.6130 - val_accuracy: 0.7978
Epoch 2/10
1805/1805 [==============================] - 100s 56ms/step - loss: 0.5740 -
accuracy: 0.8181 - val_loss: 0.5374 - val_accuracy: 0.8224
Epoch 3/10
1805/1805 [==============================] - 98s 54ms/step - loss: 0.5274 -
accuracy: 0.8286 - val_loss: 0.5101 - val_accuracy: 0.8273
Epoch 4/10
1805/1805 [==============================] - 98s 54ms/step - loss: 0.5035 -
accuracy: 0.8356 - val_loss: 0.4939 - val_accuracy: 0.8246
Epoch 5/10
1805/1805 [==============================] - 98s 54ms/step - loss: 0.4828 -
accuracy: 0.8401 - val_loss: 0.4577 - val_accuracy: 0.8431
Epoch 6/10
1805/1805 [==============================] - 98s 54ms/step - loss: 0.4630 -
accuracy: 0.8451 - val_loss: 0.4388 - val_accuracy: 0.8497
Epoch 7/10
1805/1805 [==============================] - 98s 54ms/step - loss: 0.4420 -
accuracy: 0.8502 - val_loss: 0.4264 - val_accuracy: 0.8476
Epoch 8/10
1805/1805 [==============================] - 98s 54ms/step - loss: 0.4309 -
accuracy: 0.8534 - val_loss: 0.4017 - val_accuracy: 0.8601
Epoch 9/10
1805/1805 [==============================] - 99s 55ms/step - loss: 0.4155 -
accuracy: 0.8568 - val_loss: 0.4002 - val_accuracy: 0.8591
Epoch 10/10
1805/1805 [==============================] - 101s 56ms/step - loss: 0.4042 -
accuracy: 0.8598 - val_loss: 0.3927 - val_accuracy: 0.8603
2188/2188 [==============================] - 32s 14ms/step - loss: 0.3922 -
accuracy: 0.8606
Accuracy on test data: 0.8606
learning rate:0.001 batch size: 256
Epoch 1/10
903/903 [==============================] - 76s 77ms/step - loss: 1.0967 -
accuracy: 0.7037 - val_loss: 0.5844 - val_accuracy: 0.8022
Epoch 2/10
903/903 [==============================] - 69s 76ms/step - loss: 0.5652 -
accuracy: 0.8208 - val_loss: 0.5037 - val_accuracy: 0.8312
Epoch 3/10
903/903 [==============================] - 69s 76ms/step - loss: 0.5059 -
```

```
accuracy: 0.8349 - val_loss: 0.5039 - val_accuracy: 0.8359
Epoch 4/10
903/903 [==============================] - 69s 76ms/step - loss: 0.4816 -
accuracy: 0.8412 - val_loss: 0.4928 - val_accuracy: 0.8349
Epoch 5/10
903/903 [==============================] - 68s 75ms/step - loss: 0.4654 -
accuracy: 0.8446 - val_loss: 0.5131 - val_accuracy: 0.8325
Epoch 6/10
903/903 [==============================] - 68s 75ms/step - loss: 0.4543 -
accuracy: 0.8482 - val_loss: 0.4538 - val_accuracy: 0.8444
Epoch 7/10
903/903 [==============================] - 67s 75ms/step - loss: 0.4403 -
accuracy: 0.8510 - val_loss: 0.5364 - val_accuracy: 0.8301
Epoch 8/10
903/903 [==============================] - 67s 75ms/step - loss: 0.4326 -
accuracy: 0.8529 - val_loss: 0.4478 - val_accuracy: 0.8435
Epoch 9/10
903/903 [==============================] - 67s 75ms/step - loss: 0.4231 -
accuracy: 0.8551 - val_loss: 0.4077 - val_accuracy: 0.8562
Epoch 10/10
903/903 [==============================] - 67s 74ms/step - loss: 0.4120 -
accuracy: 0.8579 - val_loss: 0.4157 - val_accuracy: 0.8518
2188/2188 [==============================] - 29s 13ms/step - loss: 0.4155 -
accuracy: 0.8533
Accuracy on test data: 0.8533
```

showing all the result and find best parameter

[ ]:
```
for item in result:
  print(item[0],item[1],item[2])
```

```
0.8442714214324951 0.01 128
0.8399571180343628 0.01 256
0.8605571389198303 0.001 128
0.8532857298851013 0.001 256
```

Model with best hyperparameter

[ ]:
```
base_model=creat_model(input_t)
base_model.compile(loss='sparse_categorical_crossentropy',
                optimizer=tf.keras.optimizers.Adam(learning_rate=0.001 ),
                metrics=['accuracy'])
  #fit the model and train
history = base_model.fit(X_train,y_train,batch_size=128,epochs=␣
 ↪15,validation_data=(X_valid,y_valid),callbacks=[callback])
loss, accuracy = base_model.evaluate(data_test_feature, data_test_label)
print(f"Accuracy on test data: {accuracy:.4f}")
```

```
Epoch 1/15
```

```
1805/1805 [==============================] - 105s 55ms/step - loss: 1.0338 -
accuracy: 0.7140 - val_loss: 0.6012 - val_accuracy: 0.8028
Epoch 2/15
1805/1805 [==============================] - 98s 54ms/step - loss: 0.5678 -
accuracy: 0.8189 - val_loss: 0.5038 - val_accuracy: 0.8215
Epoch 3/15
1805/1805 [==============================] - 98s 55ms/step - loss: 0.5256 -
accuracy: 0.8293 - val_loss: 0.4980 - val_accuracy: 0.8356
Epoch 4/15
1805/1805 [==============================] - 98s 54ms/step - loss: 0.4998 -
accuracy: 0.8354 - val_loss: 0.4699 - val_accuracy: 0.8381
Epoch 5/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.4806 -
accuracy: 0.8402 - val_loss: 0.4788 - val_accuracy: 0.8334
Epoch 6/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.4616 -
accuracy: 0.8454 - val_loss: 0.4223 - val_accuracy: 0.8511
Epoch 7/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.4444 -
accuracy: 0.8495 - val_loss: 0.4638 - val_accuracy: 0.8423
Epoch 8/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.4278 -
accuracy: 0.8535 - val_loss: 0.4264 - val_accuracy: 0.8505
Epoch 9/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.4135 -
accuracy: 0.8573 - val_loss: 0.4378 - val_accuracy: 0.8422
Epoch 10/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.4044 -
accuracy: 0.8597 - val_loss: 0.4021 - val_accuracy: 0.8576
Epoch 11/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.3920 -
accuracy: 0.8623 - val_loss: 0.3973 - val_accuracy: 0.8592
Epoch 12/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.3832 -
accuracy: 0.8647 - val_loss: 0.3973 - val_accuracy: 0.8604
Epoch 13/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.3751 -
accuracy: 0.8670 - val_loss: 0.3905 - val_accuracy: 0.8613
Epoch 14/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.3686 -
accuracy: 0.8681 - val_loss: 0.3809 - val_accuracy: 0.8637
Epoch 15/15
1805/1805 [==============================] - 97s 54ms/step - loss: 0.3593 -
accuracy: 0.8712 - val_loss: 0.3901 - val_accuracy: 0.8637
2188/2188 [==============================] - 28s 13ms/step - loss: 0.3855 -
accuracy: 0.8646
Accuracy on test data: 0.8646
```

[ ]:
```python
from sklearn.metrics import classification_report
y_pred = base_model.predict(data_test_feature, batch_size=128)
y_pred_bool = np.argmax(y_pred, axis=1)
print(classification_report(data_test_label, y_pred_bool))
```

```
              precision    recall  f1-score   support

           0       0.69      0.80      0.74      3461
           1       0.65      0.96      0.77      3860
           2       0.96      0.98      0.97      3559
           3       0.99      0.99      0.99      3577
           4       0.97      0.97      0.97      3411
           5       0.96      0.92      0.94      3115
           6       0.97      0.98      0.97      3426
           7       0.99      0.99      0.99      3768
           8       0.98      0.99      0.98      3375
           9       0.93      0.98      0.96      3390
          10       0.92      0.98      0.95       647
          11       0.94      0.93      0.94       395
          12       0.77      0.97      0.86      1080
          13       0.96      0.83      0.89       446
          14       0.95      0.97      0.96       521
          15       0.75      0.98      0.85       874
          16       0.85      0.95      0.90       254
          17       0.94      0.92      0.93       331
          18       0.67      0.46      0.55      1237
          19       0.95      0.77      0.85       391
          20       0.67      0.72      0.69       232
          21       0.88      0.91      0.89       476
          22       0.75      0.99      0.86       872
          23       0.92      0.99      0.95       804
          24       0.63      0.57      0.60      2470
          25       0.82      0.94      0.88       828
          26       0.93      0.86      0.90       257
          27       0.92      0.97      0.95       493
          28       0.80      0.96      0.87      2099
          29       0.93      0.95      0.94       965
          30       0.76      0.98      0.85      1209
          31       0.74      0.40      0.52       489
          32       0.95      0.55      0.69       467
          33       0.78      0.60      0.68       266
          34       0.83      0.71      0.77       462
          35       0.76      0.68      0.72       274
          36       0.93      0.93      0.93       955
          37       0.97      0.77      0.86       505
          38       0.00      0.00      0.00       243
          39       0.99      0.98      0.98      1036
```

```
          40       0.98      0.98      0.98      2462
          41       0.00      0.00      0.00       259
          42       0.77      0.55      0.64       375
          43       0.96      0.94      0.95       900
          44       0.73      0.41      0.52       263
          45       0.73      0.73      0.73       188
          46       0.74      0.68      0.71       283
          47       0.57      0.06      0.11      1533
          48       0.00      0.00      0.00       269
          49       0.93      0.94      0.94      1116
          50       0.00      0.00      0.00       283
          51       0.63      0.23      0.34       224
          52       0.71      0.42      0.52       295
          53       0.94      0.97      0.95      1352
          54       0.00      0.00      0.00       272
          55       0.97      0.93      0.95      1773
          56       0.17      0.00      0.01       274
          57       0.47      0.72      0.57       302
          58       0.56      0.93      0.70       283
          59       0.65      0.82      0.73       279
          60       0.57      0.51      0.54       231
          61       0.69      0.52      0.59       264

    accuracy                           0.86     70000
   macro avg       0.75      0.73      0.73     70000
weighted avg       0.85      0.86      0.85     70000

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Save the model

[ ]:
```
base_model.save('my_model.h5')
```

[ ]:
```python
from google.colab import files
files.download('/content/my_model.h5')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

System information

[ ]:
```python
!cat /proc/cpuinfo
!cat /proc/meminfo
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2199.998
cache size      : 56320 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 2
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni
pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx
f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single ssbd ibrs ibpb
stibp fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx
smap xsaveopt arat md_clear arch_capabilities
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds
swapgs taa
bogomips        : 4399.99
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 1
```

```
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2199.998
cache size      : 56320 KB
physical id     : 0
siblings        : 4
core id         : 1
cpu cores       : 2
apicid          : 2
initial apicid  : 2
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni
pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx
f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single ssbd ibrs ibpb
stibp fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx
smap xsaveopt arat md_clear arch_capabilities
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds
swapgs taa
bogomips        : 4399.99
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 2
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2199.998
cache size      : 56320 KB
physical id     : 0
siblings        : 4
core id         : 0
cpu cores       : 2
apicid          : 1
initial apicid  : 1
```

```
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni
pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx
f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single ssbd ibrs ibpb
stibp fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx
smap xsaveopt arat md_clear arch_capabilities
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds
swapgs taa
bogomips        : 4399.99
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 3
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2199.998
cache size      : 56320 KB
physical id     : 0
siblings        : 4
core id         : 1
cpu cores       : 2
apicid          : 3
initial apicid  : 3
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni
pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx
f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single ssbd ibrs ibpb
stibp fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx
smap xsaveopt arat md_clear arch_capabilities
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds
swapgs taa
bogomips        : 4399.99
```

```
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

MemTotal:       26692024 kB
MemFree:          281868 kB
MemAvailable:    7822852 kB
Buffers:           65956 kB
Cached:          3650204 kB
SwapCached:            0 kB
Active:         21863000 kB
Inactive:        3984276 kB
Active(anon):   18047084 kB
Inactive(anon):    12800 kB
Active(file):    3815916 kB
Inactive(file):  3971476 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:               732 kB
Writeback:             0 kB
AnonPages:      22131140 kB
Mapped:          1491160 kB
Shmem:             13516 kB
KReclaimable:     153532 kB
Slab:             238824 kB
SReclaimable:     153532 kB
SUnreclaim:        85292 kB
KernelStack:        7264 kB
PageTables:        77280 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:    13346012 kB
Committed_AS:   26903804 kB
VmallocTotal:   34359738367 kB
VmallocUsed:       50140 kB
VmallocChunk:          0 kB
Percpu:             3008 kB
AnonHugePages:   9654272 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
```

```
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:      441152 kB
DirectMap2M:    15284224 kB
DirectMap1G:    13631488 kB
```

[ ]:
```
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 8467891350589682677
 xla_global_id: -1, name: "/device:GPU:0"
 device_type: "GPU"
 memory_limit: 16154099712
 locality {
   bus_id: 1
   links {
   }
 }
 incarnation: 17322349674500832609
 physical_device_desc: "device: 0, name: Tesla P100-PCIE-16GB, pci bus id:
0000:00:04.0, compute capability: 6.0"
 xla_global_id: 416903419]
```