IBM Machine Learning

Course 5: Deep Learning

Topic: English Handwritten Characters

1. Introduction

We will work on the handwritten English dataset, which consists of 3410 images. The characters include 0-9, A-Z and a-z, 62 classes in total and each image is map to only one label. The objective is to classify the images to the correct label. The dataset can be found in this link: https://www.kaggle.com/dhruvildave/english-handwritten-characters-dataset?select=Img

2. Exploratory Data Analysis

2.1 import the data

In [141]:
```python
## import the library
import numpy as np
import pandas as pd

import matplotlib.image as mpimg
import matplotlib.pyplot as plt

from PIL import Image
import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
import warnings

warnings.filterwarnings('ignore')
```

In [142]:
```python
## load the data label
df=pd.read_csv("data/english.csv")
```

In [143]:
```python
## the first 5 rows in the label file
df.head()
```

Out[143]:

	image	label
0	Img/img001-001.png	0
1	Img/img001-002.png	0
2	Img/img001-003.png	0
3	Img/img001-004.png	0
4	Img/img001-005.png	0

In [144]:
```python
## As describe in introduction, there 3410 records.
df.info()
```

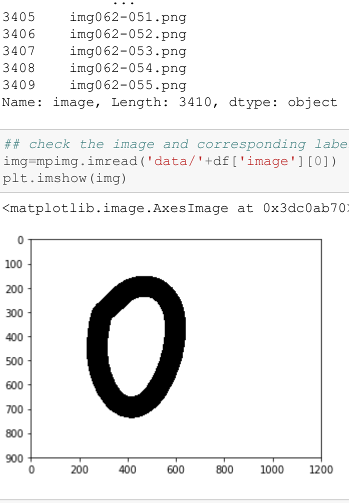
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3410 entries, 0 to 3409
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   image   3410 non-null   object
 1   label   3410 non-null   object
dtypes: object(2)
memory usage: 53.4+ KB
```

In [145]:
```python
## Process the image column
df['image']=df['image'].str.split('/').str[1]
df['image']
```

Out[145]:
```
0       img001-001.png
1       img001-002.png
2       img001-003.png
3       img001-004.png
4       img001-005.png
             ...
3405    img062-051.png
3406    img062-052.png
3407    img062-053.png
3408    img062-054.png
3409    img062-055.png
Name: image, Length: 3410, dtype: object
```

In [146]:
```python
## check the image and corresponding label
img=mpimg.imread('data/'+df['image'][0])
plt.imshow(img)
```

Out[146]: `<matplotlib.image.AxesImage at 0x3dc0ab70>`



In [147]:
```python
df['label'][0]
```

Out[147]: `'0'`

2.2 convert each image to 2D array

From the image we can see that each image have 900 pixels in height and 1200 pixels in width. We have 3409 iamges. If we use the original dimension, it will need a few GB memory. we will compress and convert each image to an array of size(30, 40)

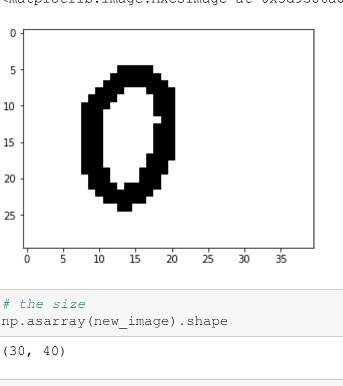
In [148]:
```python
from sklearn.preprocessing import LabelEncoder

# before resize
image=Image.open('data/'+df['image'][0]).convert('L')
```

In [149]:
```python
print(image.format)
print(image.size)
print(image.mode)
```

```
None
(1200, 900)
L
```

In [150]:
```python
# after resize
new_image=image.resize((40,30))
plt.imshow(new_image)
```

Out[150]: `<matplotlib.image.AxesImage at 0x3d9586a0>`



In [151]:
```python
# the size
np.asarray(new_image).shape
```

Out[151]: `(30, 40)`

In [152]:
```python
# create the dataset for the pixel value of each image
L=[]
for image in df['image']:
    rawData=np.asarray(Image.open('data/'+image).convert('L').resize((40,30)))
    L.append(rawData)
```

In [153]:
```python
# convert to numpy array
len(L)
X=np.array(L)
X.shape
```

Out[153]: `(3410, 30, 40)`

In [154]:
```python
# create the data for label
y=df['label']
```

In [155]:
```python
# creating instance of labelencoder
labelencoder = LabelEncoder()

# Assigning numerical values
y=labelencoder.fit_transform(y)
```

2.3 Data preprocess

In [156]:
```python
# Let's make everything float and scale
X=X.astype('float32')
X/=255
```

2.4 Data splitted into training and test

In [157]:
```python
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, y,
                                                    stratify=y,
                                                    test_size=0.25)
```

In [158]:
```python
x_train.shape
```

Out[158]: `(2557, 30, 40)`

3. Build Model

3.1 Get a baseline performance using Random Forest

In [159]:
```python
## flatten our data
x_train_flat=x_train.reshape(len(x_train), -1)
x_test_flat=x_test.reshape(len(x_test), -1)
```

In [160]:
```python
## the size of training
print('x_train_flat shape:', x_train_flat.shape)
```

```
x_train_flat shape: (2557, 1200)
```

In [161]:
```python
## the size of testing
print('x_test_flat shape:', x_test_flat.shape)
```

```
x_test_flat shape: (853, 1200)
```

In [162]:
```python
import datetime
#used to help some of the timing functions
now = datetime.datetime.now
```

In [163]:
```python
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_auc_score, roc_curve, accura
cy_score
from sklearn.ensemble import RandomForestClassifier

## Train the RF Model
t = now()
rf_model = RandomForestClassifier(n_estimators=200)
rf_model.fit(x_train_flat, y_train)
print('Training time: %s' % (now() - t))
```

```
Training time: 0:00:02.186125
```

In [164]:
```python
# Make predictions on the test set - both "hard" predictions, and the scores (percent of trees voting
 yes)
y_pred_class_rf = rf_model.predict(x_test_flat)
y_pred_prob_rf = rf_model.predict_proba(x_test_flat)

print('accuracy is {:.3f}'.format(accuracy_score(y_test,y_pred_class_rf)))
```

```
accuracy is 0.449
```

In [165]:
```python
confusion_matrix(y_test,y_pred_class_rf)
```

Out[165]:
```
array([[ 4,  0,  0, ...,  0,  0,  0],
       [ 0, 10,  0, ...,  0,  0,  0],
       [ 0,  0, 10, ...,  0,  0,  0],
       ...,
       [ 0,  0,  0, ...,  4,  0,  1],
       [ 0,  0,  0, ...,  1,  5,  0],
       [ 0,  0,  1, ...,  1,  0,  5]], dtype=int64)
```

As we can see above, the training time is long and the accuracy is quite low(less than 50%)

3.2 Build a full connected neural network

In [166]:
```python
num_classes = len(set(y))

y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

In [167]:
```python
model_1=Sequential()
model_1.add(Dense(100,input_shape=(1200,), activation='sigmoid'))
model_1.add(Dense(62,activation='softmax'))
```

In [168]:
```python
model_1.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_10 (Dense)             (None, 100)               120100
_________________________________________________________________
dense_11 (Dense)             (None, 62)                6262
=================================================================
Total params: 126,362
Trainable params: 126,362
Non-trainable params: 0
_________________________________________________________________
```

```
In [169]: t = now()

          # Compile the model with Optimizer, Loss Function and Metrics
          opt_1 = keras.optimizers.RMSprop(lr=0.0005)

          model_1.compile(optimizer=opt_1, loss="categorical_crossentropy", metrics=["accuracy"])
          run_hist_1 = model_1.fit(x_train_flat, y_train, validation_data=(x_test_flat, y_test), epochs=200)

          print('Training time: %s' % (now() - t))
```

```
Epoch 1/200
80/80 [==============================] - 0s 3ms/step - loss: 4.1787 - accuracy: 0.0160 - val_loss: 4.
1254 - val_accuracy: 0.0211
Epoch 2/200
80/80 [==============================] - 0s 2ms/step - loss: 4.1243 - accuracy: 0.0235 - val_loss: 4.
1042 - val_accuracy: 0.0457
Epoch 3/200
80/80 [==============================] - 0s 1ms/step - loss: 4.0929 - accuracy: 0.0305 - val_loss: 4.
0581 - val_accuracy: 0.0352
Epoch 4/200
80/80 [==============================] - 0s 1ms/step - loss: 4.0302 - accuracy: 0.0372 - val_loss: 4.
0225 - val_accuracy: 0.0399
Epoch 5/200
80/80 [==============================] - 0s 1ms/step - loss: 3.9689 - accuracy: 0.0571 - val_loss: 3.
9600 - val_accuracy: 0.0715
Epoch 6/200
80/80 [==============================] - 0s 1ms/step - loss: 3.9033 - accuracy: 0.0755 - val_loss: 3.
9012 - val_accuracy: 0.0680
Epoch 7/200
80/80 [==============================] - 0s 1ms/step - loss: 3.8296 - accuracy: 0.1060 - val_loss: 3.
8459 - val_accuracy: 0.0797
Epoch 8/200
80/80 [==============================] - 0s 2ms/step - loss: 3.7624 - accuracy: 0.1279 - val_loss: 3.
7920 - val_accuracy: 0.1020
Epoch 9/200
80/80 [==============================] - 0s 2ms/step - loss: 3.6869 - accuracy: 0.1486 - val_loss: 3.
7252 - val_accuracy: 0.1313
Epoch 10/200
80/80 [==============================] - 0s 1ms/step - loss: 3.6252 - accuracy: 0.1658 - val_loss: 3.
6671 - val_accuracy: 0.1348
Epoch 11/200
80/80 [==============================] - 0s 1ms/step - loss: 3.5523 - accuracy: 0.1979 - val_loss: 3.
6011 - val_accuracy: 0.1676
Epoch 12/200
80/80 [==============================] - 0s 1ms/step - loss: 3.4790 - accuracy: 0.2081 - val_loss: 3.
5393 - val_accuracy: 0.1524
Epoch 13/200
80/80 [==============================] - 0s 1ms/step - loss: 3.4214 - accuracy: 0.2276 - val_loss: 3.
4947 - val_accuracy: 0.1981
Epoch 14/200
80/80 [==============================] - 0s 1ms/step - loss: 3.3563 - accuracy: 0.2464 - val_loss: 3.
4636 - val_accuracy: 0.1676
Epoch 15/200
80/80 [==============================] - 0s 1ms/step - loss: 3.2975 - accuracy: 0.2538 - val_loss: 3.
4078 - val_accuracy: 0.2286
Epoch 16/200
80/80 [==============================] - 0s 1ms/step - loss: 3.2486 - accuracy: 0.2663 - val_loss: 3.
3633 - val_accuracy: 0.2005
Epoch 17/200
80/80 [==============================] - 0s 1ms/step - loss: 3.1863 - accuracy: 0.2914 - val_loss: 3.
3248 - val_accuracy: 0.2169
Epoch 18/200
80/80 [==============================] - 0s 1ms/step - loss: 3.1430 - accuracy: 0.2804 - val_loss: 3.
2714 - val_accuracy: 0.2450
Epoch 19/200
80/80 [==============================] - 0s 1ms/step - loss: 3.0859 - accuracy: 0.3125 - val_loss: 3.
2537 - val_accuracy: 0.2392
Epoch 20/200
80/80 [==============================] - 0s 1ms/step - loss: 3.0384 - accuracy: 0.3269 - val_loss: 3.
2062 - val_accuracy: 0.2579
Epoch 21/200
80/80 [==============================] - 0s 1ms/step - loss: 3.0062 - accuracy: 0.3273 - val_loss: 3.
1845 - val_accuracy: 0.2567
Epoch 22/200
80/80 [==============================] - 0s 1ms/step - loss: 2.9514 - accuracy: 0.3359 - val_loss: 3.
1596 - val_accuracy: 0.2521
Epoch 23/200
80/80 [==============================] - 0s 1ms/step - loss: 2.9149 - accuracy: 0.3465 - val_loss: 3.
1335 - val_accuracy: 0.2403
Epoch 24/200
80/80 [==============================] - 0s 1ms/step - loss: 2.8736 - accuracy: 0.3531 - val_loss: 3.
0901 - val_accuracy: 0.2755
Epoch 25/200
80/80 [==============================] - 0s 1ms/step - loss: 2.8418 - accuracy: 0.3559 - val_loss: 3.
0683 - val_accuracy: 0.2755
Epoch 26/200
80/80 [==============================] - 0s 1ms/step - loss: 2.8027 - accuracy: 0.3582 - val_loss: 3.
0699 - val_accuracy: 0.2767
Epoch 27/200
80/80 [==============================] - 0s 1ms/step - loss: 2.7672 - accuracy: 0.3696 - val_loss: 3.
0457 - val_accuracy: 0.2614
Epoch 28/200
80/80 [==============================] - 0s 1ms/step - loss: 2.7379 - accuracy: 0.3766 - val_loss: 3.
0259 - val_accuracy: 0.2532
Epoch 29/200
80/80 [==============================] - 0s 1ms/step - loss: 2.7097 - accuracy: 0.3817 - val_loss: 2.
9494 - val_accuracy: 0.3118
Epoch 30/200
80/80 [==============================] - 0s 1ms/step - loss: 2.6711 - accuracy: 0.3817 - val_loss: 2.
9578 - val_accuracy: 0.2884
Epoch 31/200
80/80 [==============================] - 0s 1ms/step - loss: 2.6403 - accuracy: 0.3954 - val_loss: 2.
9805 - val_accuracy: 0.2755
Epoch 32/200
80/80 [==============================] - 0s 1ms/step - loss: 2.6155 - accuracy: 0.3989 - val_loss: 2.
9317 - val_accuracy: 0.3107
Epoch 33/200
80/80 [==============================] - 0s 1ms/step - loss: 2.5904 - accuracy: 0.4048 - val_loss: 2.
8948 - val_accuracy: 0.3025
Epoch 34/200
80/80 [==============================] - 0s 1ms/step - loss: 2.5607 - accuracy: 0.4095 - val_loss: 2.
8858 - val_accuracy: 0.3036
Epoch 35/200
80/80 [==============================] - 0s 1ms/step - loss: 2.5370 - accuracy: 0.3989 - val_loss: 2.
8850 - val_accuracy: 0.3154
Epoch 36/200
80/80 [==============================] - 0s 1ms/step - loss: 2.5103 - accuracy: 0.4247 - val_loss: 2.
8791 - val_accuracy: 0.3025
Epoch 37/200
80/80 [==============================] - 0s 1ms/step - loss: 2.4762 - accuracy: 0.4204 - val_loss: 2.
8483 - val_accuracy: 0.3200
Epoch 38/200
80/80 [==============================] - 0s 1ms/step - loss: 2.4574 - accuracy: 0.4251 - val_loss: 2.
8944 - val_accuracy: 0.2814
Epoch 39/200
80/80 [==============================] - 0s 1ms/step - loss: 2.4368 - accuracy: 0.4216 - val_loss: 2.
8132 - val_accuracy: 0.3236
Epoch 40/200
80/80 [==============================] - 0s 1ms/step - loss: 2.4204 - accuracy: 0.4357 - val_loss: 2.
8085 - val_accuracy: 0.3294
Epoch 41/200
80/80 [==============================] - 0s 1ms/step - loss: 2.3908 - accuracy: 0.4341 - val_loss: 2.
8282 - val_accuracy: 0.3212
Epoch 42/200
80/80 [==============================] - 0s 1ms/step - loss: 2.3757 - accuracy: 0.4361 - val_loss: 2.
7758 - val_accuracy: 0.3283
Epoch 43/200
80/80 [==============================] - 0s 1ms/step - loss: 2.3523 - accuracy: 0.4501 - val_loss: 2.
7997 - val_accuracy: 0.3212
Epoch 44/200
80/80 [==============================] - 0s 1ms/step - loss: 2.3358 - accuracy: 0.4505 - val_loss: 2.
7751 - val_accuracy: 0.3306
Epoch 45/200
80/80 [==============================] - 0s 1ms/step - loss: 2.3184 - accuracy: 0.4580 - val_loss: 2.
7678 - val_accuracy: 0.3236
Epoch 46/200
80/80 [==============================] - 0s 1ms/step - loss: 2.2922 - accuracy: 0.4564 - val_loss: 2.
7532 - val_accuracy: 0.3411
Epoch 47/200
80/80 [==============================] - 0s 1ms/step - loss: 2.2790 - accuracy: 0.4540 - val_loss: 2.
7757 - val_accuracy: 0.3294
Epoch 48/200
80/80 [==============================] - 0s 1ms/step - loss: 2.2617 - accuracy: 0.4627 - val_loss: 2.
7255 - val_accuracy: 0.3306
Epoch 49/200
80/80 [==============================] - 0s 1ms/step - loss: 2.2471 - accuracy: 0.4724 - val_loss: 2.
7612 - val_accuracy: 0.3306
Epoch 50/200
80/80 [==============================] - 0s 1ms/step - loss: 2.2327 - accuracy: 0.4685 - val_loss: 2.
7286 - val_accuracy: 0.3341
Epoch 51/200
80/80 [==============================] - 0s 1ms/step - loss: 2.2081 - accuracy: 0.4732 - val_loss: 2.
7543 - val_accuracy: 0.3318
Epoch 52/200
80/80 [==============================] - 0s 1ms/step - loss: 2.1967 - accuracy: 0.4767 - val_loss: 2.
6864 - val_accuracy: 0.3494
Epoch 53/200
80/80 [==============================] - 0s 1ms/step - loss: 2.1778 - accuracy: 0.4814 - val_loss: 2.
6680 - val_accuracy: 0.3646
Epoch 54/200
80/80 [==============================] - 0s 1ms/step - loss: 2.1605 - accuracy: 0.4842 - val_loss: 2.
6784 - val_accuracy: 0.3458
Epoch 55/200
80/80 [==============================] - 0s 1ms/step - loss: 2.1482 - accuracy: 0.4869 - val_loss: 2.
6778 - val_accuracy: 0.3458
Epoch 56/200
80/80 [==============================] - 0s 1ms/step - loss: 2.1289 - accuracy: 0.4842 - val_loss: 2.
6863 - val_accuracy: 0.3505
Epoch 57/200
80/80 [==============================] - 0s 1ms/step - loss: 2.1158 - accuracy: 0.4955 - val_loss: 2.
7195 - val_accuracy: 0.3376
Epoch 58/200
80/80 [==============================] - 0s 1ms/step - loss: 2.0992 - accuracy: 0.5025 - val_loss: 2.
6999 - val_accuracy: 0.3505
Epoch 59/200
80/80 [==============================] - 0s 1ms/step - loss: 2.0808 - accuracy: 0.4963 - val_loss: 2.
6564 - val_accuracy: 0.3669
Epoch 60/200
80/80 [==============================] - 0s 1ms/step - loss: 2.0723 - accuracy: 0.5057 - val_loss: 2.
6330 - val_accuracy: 0.3646
Epoch 61/200
80/80 [==============================] - 0s 1ms/step - loss: 2.0624 - accuracy: 0.5053 - val_loss: 2.
6518 - val_accuracy: 0.3623
Epoch 62/200
80/80 [==============================] - 0s 1ms/step - loss: 2.0498 - accuracy: 0.5158 - val_loss: 2.
6741 - val_accuracy: 0.3540
Epoch 63/200
80/80 [==============================] - 0s 1ms/step - loss: 2.0339 - accuracy: 0.5029 - val_loss: 2.
6312 - val_accuracy: 0.3681
Epoch 64/200
80/80 [==============================] - 0s 1ms/step - loss: 2.0186 - accuracy: 0.5123 - val_loss: 2.
6320 - val_accuracy: 0.3587
Epoch 65/200
80/80 [==============================] - 0s 1ms/step - loss: 2.0082 - accuracy: 0.5151 - val_loss: 2.
6527 - val_accuracy: 0.3599
Epoch 66/200
80/80 [==============================] - 0s 1ms/step - loss: 2.0041 - accuracy: 0.5139 - val_loss: 2.
6225 - val_accuracy: 0.3517
Epoch 67/200
80/80 [==============================] - 0s 2ms/step - loss: 1.9797 - accuracy: 0.5209 - val_loss: 2.
6221 - val_accuracy: 0.3740
Epoch 68/200
80/80 [==============================] - 0s 1ms/step - loss: 1.9757 - accuracy: 0.5162 - val_loss: 2.
6333 - val_accuracy: 0.3447
Epoch 69/200
80/80 [==============================] - 0s 1ms/step - loss: 1.9567 - accuracy: 0.5276 - val_loss: 2.
6780 - val_accuracy: 0.3552
Epoch 70/200
80/80 [==============================] - 0s 1ms/step - loss: 1.9378 - accuracy: 0.5420 - val_loss: 2.
6214 - val_accuracy: 0.3634
Epoch 71/200
80/80 [==============================] - 0s 1ms/step - loss: 1.9349 - accuracy: 0.5264 - val_loss: 2.
5941 - val_accuracy: 0.3693
Epoch 72/200
80/80 [==============================] - 0s 1ms/step - loss: 1.9199 - accuracy: 0.5330 - val_loss: 2.
6288 - val_accuracy: 0.3634
Epoch 73/200
80/80 [==============================] - 0s 1ms/step - loss: 1.9074 - accuracy: 0.5385 - val_loss: 2.
6420 - val_accuracy: 0.3564
Epoch 74/200
80/80 [==============================] - 0s 1ms/step - loss: 1.8956 - accuracy: 0.5440 - val_loss: 2.
5801 - val_accuracy: 0.3787
Epoch 75/200
80/80 [==============================] - 0s 1ms/step - loss: 1.8769 - accuracy: 0.5460 - val_loss: 2.
6222 - val_accuracy: 0.3693
Epoch 76/200
80/80 [==============================] - 0s 1ms/step - loss: 1.8714 - accuracy: 0.5397 - val_loss: 2.
6009 - val_accuracy: 0.3623
Epoch 77/200
80/80 [==============================] - 0s 1ms/step - loss: 1.8620 - accuracy: 0.5460 - val_loss: 2.
6238 - val_accuracy: 0.3599
Epoch 78/200
80/80 [==============================] - 0s 2ms/step - loss: 1.8504 - accuracy: 0.5460 - val_loss: 2.
5777 - val_accuracy: 0.3740
Epoch 79/200
80/80 [==============================] - 0s 1ms/step - loss: 1.8367 - accuracy: 0.5573 - val_loss: 2.
5973 - val_accuracy: 0.3693
Epoch 80/200
80/80 [==============================] - 0s 1ms/step - loss: 1.8248 - accuracy: 0.5581 - val_loss: 2.
5970 - val_accuracy: 0.3540
Epoch 81/200
80/80 [==============================] - 0s 1ms/step - loss: 1.8114 - accuracy: 0.5522 - val_loss: 2.
5441 - val_accuracy: 0.3857
Epoch 82/200
80/80 [==============================] - 0s 1ms/step - loss: 1.8101 - accuracy: 0.5632 - val_loss: 2.
5790 - val_accuracy: 0.3822
Epoch 83/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7915 - accuracy: 0.5620 - val_loss: 2.
5928 - val_accuracy: 0.3646
Epoch 84/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7851 - accuracy: 0.5702 - val_loss: 2.
5718 - val_accuracy: 0.3646
Epoch 85/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7765 - accuracy: 0.5667 - val_loss: 2.
5987 - val_accuracy: 0.3658
Epoch 86/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7607 - accuracy: 0.5643 - val_loss: 2.
5907 - val_accuracy: 0.3611
Epoch 87/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7473 - accuracy: 0.5710 - val_loss: 2.
6001 - val_accuracy: 0.3669
Epoch 88/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7334 - accuracy: 0.5765 - val_loss: 2.
5699 - val_accuracy: 0.3669
Epoch 89/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7341 - accuracy: 0.5772 - val_loss: 2.
5240 - val_accuracy: 0.3810
Epoch 90/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7235 - accuracy: 0.5788 - val_loss: 2.
5802 - val_accuracy: 0.3705
Epoch 91/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7164 - accuracy: 0.5827 - val_loss: 2.
5749 - val_accuracy: 0.3599
Epoch 92/200
80/80 [==============================] - 0s 1ms/step - loss: 1.7023 - accuracy: 0.5804 - val_loss: 2.
5802 - val_accuracy: 0.3763
Epoch 93/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6890 - accuracy: 0.5886 - val_loss: 2.
5524 - val_accuracy: 0.3669
Epoch 94/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6822 - accuracy: 0.5925 - val_loss: 2.
5769 - val_accuracy: 0.3834
Epoch 95/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6709 - accuracy: 0.5866 - val_loss: 2.
5459 - val_accuracy: 0.3751
Epoch 96/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6635 - accuracy: 0.5964 - val_loss: 2.
5385 - val_accuracy: 0.3728
Epoch 97/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6551 - accuracy: 0.5925 - val_loss: 2.
5906 - val_accuracy: 0.3705
Epoch 98/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6472 - accuracy: 0.6023 - val_loss: 2.
5407 - val_accuracy: 0.3763
Epoch 99/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6306 - accuracy: 0.6105 - val_loss: 2.
5412 - val_accuracy: 0.3716
Epoch 100/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6195 - accuracy: 0.6050 - val_loss: 2.
5366 - val_accuracy: 0.4021
Epoch 101/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6143 - accuracy: 0.6089 - val_loss: 2.
5119 - val_accuracy: 0.3880
Epoch 102/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6029 - accuracy: 0.6120 - val_loss: 2.
6120 - val_accuracy: 0.3576
Epoch 103/200
80/80 [==============================] - 0s 1ms/step - loss: 1.6026 - accuracy: 0.6124 - val_loss: 2.
5839 - val_accuracy: 0.3623
Epoch 104/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5811 - accuracy: 0.6179 - val_loss: 2.
5457 - val_accuracy: 0.3705
Epoch 105/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5757 - accuracy: 0.6089 - val_loss: 2.
5536 - val_accuracy: 0.3658
Epoch 106/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5715 - accuracy: 0.6289 - val_loss: 2.
5222 - val_accuracy: 0.3892
Epoch 107/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5649 - accuracy: 0.6253 - val_loss: 2.
5622 - val_accuracy: 0.3669
Epoch 108/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5454 - accuracy: 0.6203 - val_loss: 2.
5417 - val_accuracy: 0.3751
Epoch 109/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5352 - accuracy: 0.6300 - val_loss: 2.
5308 - val_accuracy: 0.3857
Epoch 110/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5322 - accuracy: 0.6226 - val_loss: 2.
5482 - val_accuracy: 0.3822
Epoch 111/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5233 - accuracy: 0.6312 - val_loss: 2.
5157 - val_accuracy: 0.3904
Epoch 112/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5130 - accuracy: 0.6328 - val_loss: 2.
5743 - val_accuracy: 0.3646
Epoch 113/200
80/80 [==============================] - 0s 1ms/step - loss: 1.5093 - accuracy: 0.6379 - val_loss: 2.
5326 - val_accuracy: 0.3880
Epoch 114/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4987 - accuracy: 0.6339 - val_loss: 2.
5051 - val_accuracy: 0.3810
Epoch 115/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4848 - accuracy: 0.6394 - val_loss: 2.
5515 - val_accuracy: 0.3763
Epoch 116/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4801 - accuracy: 0.6390 - val_loss: 2.
5525 - val_accuracy: 0.3705
Epoch 117/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4696 - accuracy: 0.6414 - val_loss: 2.
5132 - val_accuracy: 0.3892
Epoch 118/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4585 - accuracy: 0.6437 - val_loss: 2.
5079 - val_accuracy: 0.3740
Epoch 119/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4584 - accuracy: 0.6476 - val_loss: 2.
5949 - val_accuracy: 0.3693
Epoch 120/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4416 - accuracy: 0.6469 - val_loss: 2.
5203 - val_accuracy: 0.3927
Epoch 121/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4348 - accuracy: 0.6461 - val_loss: 2.
5352 - val_accuracy: 0.3834
Epoch 122/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4263 - accuracy: 0.6562 - val_loss: 2.
5212 - val_accuracy: 0.3857
Epoch 123/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4208 - accuracy: 0.6547 - val_loss: 2.
5371 - val_accuracy: 0.3751
Epoch 124/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4085 - accuracy: 0.6598 - val_loss: 2.
5558 - val_accuracy: 0.3763
Epoch 125/200
80/80 [==============================] - 0s 1ms/step - loss: 1.4007 - accuracy: 0.6551 - val_loss: 2.
5634 - val_accuracy: 0.3693
Epoch 126/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3936 - accuracy: 0.6539 - val_loss: 2.
5465 - val_accuracy: 0.3834
Epoch 127/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3893 - accuracy: 0.6637 - val_loss: 2.
5137 - val_accuracy: 0.3986
Epoch 128/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3758 - accuracy: 0.6578 - val_loss: 2.
5291 - val_accuracy: 0.3857
Epoch 129/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3705 - accuracy: 0.6727 - val_loss: 2.
5856 - val_accuracy: 0.3693
Epoch 130/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3553 - accuracy: 0.6734 - val_loss: 2.
5105 - val_accuracy: 0.3962
Epoch 131/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3559 - accuracy: 0.6754 - val_loss: 2.
5299 - val_accuracy: 0.3939
Epoch 132/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3464 - accuracy: 0.6762 - val_loss: 2.
5134 - val_accuracy: 0.3892
Epoch 133/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3378 - accuracy: 0.6750 - val_loss: 2.
5776 - val_accuracy: 0.3646
Epoch 134/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3289 - accuracy: 0.6864 - val_loss: 2.
5083 - val_accuracy: 0.3857
Epoch 135/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3194 - accuracy: 0.6699 - val_loss: 2.
5229 - val_accuracy: 0.3904
Epoch 136/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3125 - accuracy: 0.6727 - val_loss: 2.
5014 - val_accuracy: 0.3974
Epoch 137/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3076 - accuracy: 0.6871 - val_loss: 2.
5368 - val_accuracy: 0.3880
Epoch 138/200
80/80 [==============================] - 0s 1ms/step - loss: 1.3011 - accuracy: 0.6934 - val_loss: 2.
4971 - val_accuracy: 0.3962
Epoch 139/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2844 - accuracy: 0.6852 - val_loss: 2.
5610 - val_accuracy: 0.3892
Epoch 140/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2846 - accuracy: 0.6887 - val_loss: 2.
5355 - val_accuracy: 0.3880
Epoch 141/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2700 - accuracy: 0.6993 - val_loss: 2.
5672 - val_accuracy: 0.3787
Epoch 142/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2615 - accuracy: 0.6973 - val_loss: 2.
5386 - val_accuracy: 0.3728
Epoch 143/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2654 - accuracy: 0.6981 - val_loss: 2.
5273 - val_accuracy: 0.3880
Epoch 144/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2464 - accuracy: 0.7008 - val_loss: 2.
5610 - val_accuracy: 0.3834
Epoch 145/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2480 - accuracy: 0.7114 - val_loss: 2.
5013 - val_accuracy: 0.3857
Epoch 146/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2270 - accuracy: 0.7141 - val_loss: 2.
5328 - val_accuracy: 0.3810
Epoch 147/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2249 - accuracy: 0.7063 - val_loss: 2.
5879 - val_accuracy: 0.3728
Epoch 148/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2207 - accuracy: 0.7008 - val_loss: 2.
5851 - val_accuracy: 0.3763
Epoch 149/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2160 - accuracy: 0.7114 - val_loss: 2.
5429 - val_accuracy: 0.3892
Epoch 150/200
80/80 [==============================] - 0s 1ms/step - loss: 1.2077 - accuracy: 0.7165 - val_loss: 2.
5850 - val_accuracy: 0.3857
Epoch 151/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1970 - accuracy: 0.7188 - val_loss: 2.
5580 - val_accuracy: 0.3927
Epoch 152/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1888 - accuracy: 0.7176 - val_loss: 2.
5282 - val_accuracy: 0.3810
Epoch 153/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1853 - accuracy: 0.7180 - val_loss: 2.
5706 - val_accuracy: 0.3869
Epoch 154/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1767 - accuracy: 0.7290 - val_loss: 2.
5489 - val_accuracy: 0.3892
Epoch 155/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1679 - accuracy: 0.7270 - val_loss: 2.
5322 - val_accuracy: 0.3892
Epoch 156/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1627 - accuracy: 0.7313 - val_loss: 2.
5590 - val_accuracy: 0.3916
Epoch 157/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1512 - accuracy: 0.7259 - val_loss: 2.
5295 - val_accuracy: 0.3892
Epoch 158/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1476 - accuracy: 0.7309 - val_loss: 2.
5387 - val_accuracy: 0.3892
Epoch 159/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1422 - accuracy: 0.7345 - val_loss: 2.
5820 - val_accuracy: 0.3763
Epoch 160/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1343 - accuracy: 0.7341 - val_loss: 2.
5123 - val_accuracy: 0.4045
Epoch 161/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1263 - accuracy: 0.7333 - val_loss: 2.
5070 - val_accuracy: 0.4033
Epoch 162/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1129 - accuracy: 0.7415 - val_loss: 2.
5600 - val_accuracy: 0.3880
Epoch 163/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1134 - accuracy: 0.7384 - val_loss: 2.
5449 - val_accuracy: 0.3904
Epoch 164/200
80/80 [==============================] - 0s 1ms/step - loss: 1.1056 - accuracy: 0.7391 - val_loss: 2.
5383 - val_accuracy: 0.3857
Epoch 165/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0944 - accuracy: 0.7481 - val_loss: 2.
5108 - val_accuracy: 0.3916
Epoch 166/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0922 - accuracy: 0.7442 - val_loss: 2.
5450 - val_accuracy: 0.3751
Epoch 167/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0899 - accuracy: 0.7419 - val_loss: 2.
5546 - val_accuracy: 0.3869
Epoch 168/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0757 - accuracy: 0.7497 - val_loss: 2.
5651 - val_accuracy: 0.3927
Epoch 169/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0661 - accuracy: 0.7560 - val_loss: 2.
5570 - val_accuracy: 0.3869
Epoch 170/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0634 - accuracy: 0.7446 - val_loss: 2.
5231 - val_accuracy: 0.3775
Epoch 171/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0578 - accuracy: 0.7595 - val_loss: 2.
5231 - val_accuracy: 0.3916
Epoch 172/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0505 - accuracy: 0.7548 - val_loss: 2.
5450 - val_accuracy: 0.3810
Epoch 173/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0443 - accuracy: 0.7575 - val_loss: 2.
5473 - val_accuracy: 0.3810
Epoch 174/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0376 - accuracy: 0.7689 - val_loss: 2.
5653 - val_accuracy: 0.3845
Epoch 175/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0343 - accuracy: 0.7654 - val_loss: 2.
5836 - val_accuracy: 0.3939
Epoch 176/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0191 - accuracy: 0.7673 - val_loss: 2.
5516 - val_accuracy: 0.3857
Epoch 177/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0131 - accuracy: 0.7669 - val_loss: 2.
5816 - val_accuracy: 0.3927
Epoch 178/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0061 - accuracy: 0.7689 - val_loss: 2.
5957 - val_accuracy: 0.3810
Epoch 179/200
80/80 [==============================] - 0s 1ms/step - loss: 1.0087 - accuracy: 0.7646 - val_loss: 2.
6407 - val_accuracy: 0.3916
Epoch 180/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9983 - accuracy: 0.7740 - val_loss: 2.
5784 - val_accuracy: 0.3798
Epoch 181/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9932 - accuracy: 0.7646 - val_loss: 2.
5488 - val_accuracy: 0.3951
Epoch 182/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9794 - accuracy: 0.7818 - val_loss: 2.
6057 - val_accuracy: 0.3939
Epoch 183/200
80/80 [==============================] - 0s 2ms/step - loss: 0.9812 - accuracy: 0.7724 - val_loss: 2.
6079 - val_accuracy: 0.3728
Epoch 184/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9730 - accuracy: 0.7775 - val_loss: 2.
5540 - val_accuracy: 0.3705
Epoch 185/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9612 - accuracy: 0.7794 - val_loss: 2.
5599 - val_accuracy: 0.3716
Epoch 186/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9566 - accuracy: 0.7826 - val_loss: 2.
5842 - val_accuracy: 0.3845
Epoch 187/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9557 - accuracy: 0.7826 - val_loss: 2.
5804 - val_accuracy: 0.3810
Epoch 188/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9509 - accuracy: 0.7849 - val_loss: 2.
5698 - val_accuracy: 0.3845
Epoch 189/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9424 - accuracy: 0.7763 - val_loss: 2.
5667 - val_accuracy: 0.3822
Epoch 190/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9376 - accuracy: 0.7970 - val_loss: 2.
5728 - val_accuracy: 0.3869
Epoch 191/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9295 - accuracy: 0.7951 - val_loss: 2.
5632 - val_accuracy: 0.3822
Epoch 192/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9199 - accuracy: 0.7916 - val_loss: 2.
6095 - val_accuracy: 0.3951
Epoch 193/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9162 - accuracy: 0.7861 - val_loss: 2.
5975 - val_accuracy: 0.3728
Epoch 194/200
80/80 [==============================] - 0s 1ms/step - loss: 0.9039 - accuracy: 0.7974 - val_loss: 2.
6058 - val_accuracy: 0.3751
Epoch 195/200
80/80 [==============================] - 0s 1ms/step - loss: 0.8992 - accuracy: 0.8029 - val_loss: 2.
6099 - val_accuracy: 0.3939
Epoch 196/200
80/80 [==============================] - 0s 1ms/step - loss: 0.8963 - accuracy: 0.7986 - val_loss: 2.
5966 - val_accuracy: 0.3834
Epoch 197/200
80/80 [==============================] - 0s 1ms/step - loss: 0.8909 - accuracy: 0.8025 - val_loss: 2.
6315 - val_accuracy: 0.3763
Epoch 198/200
80/80 [==============================] - 0s 1ms/step - loss: 0.8753 - accuracy: 0.8099 - val_loss: 2.
6533 - val_accuracy: 0.3775
Epoch 199/200
80/80 [==============================] - 0s 1ms/step - loss: 0.8857 - accuracy: 0.8041 - val_loss: 2.
5433 - val_accuracy: 0.3939
Epoch 200/200
80/80 [==============================] - 0s 1ms/step - loss: 0.8709 - accuracy: 0.8103 - val_loss: 2.
6049 - val_accuracy: 0.3822
Training time: 0:00:23.531346
```

```
In [170]: score = model_1.evaluate(x_test_flat, y_test, verbose=0)
          print('Test score:', score[0])
          print('Test accuracy:', score[1])
```

```
Test score: 2.60488224029541
Test accuracy: 0.3821805417537689
```

The full connection neural network perform no better than random forest. The accuray is similar to random forest. We can increase the hidden layers and number of epochs, however the time will increase too.

3.3 Convolutional Neural Network

```
In [171]: # set some parameters
          img_rows, img_cols=x_train.shape[1:]
          input_shape = (img_rows, img_cols, 1)

          filters = 32
          pool_size = 2
          kernel_size = 3
```

```
In [172]: input_shape
```

```
Out[172]: (30, 40, 1)
```

```
In [173]: model_2 = Sequential()

          model_2.add(Conv2D(32, (3, 4), padding='same',
                          input_shape=input_shape))
          model_2.add(Activation('relu'))
          model_2.add(Conv2D(32, (3, 4), padding='same',
                          input_shape=input_shape))
          model_2.add(Activation('relu'))
          model_2.add(MaxPooling2D(pool_size=(3, 4)))
          model_2.add(Dropout(0.25))

          model_2.add(Conv2D(64, (3, 3), padding='same'))
          model_2.add(Activation('relu'))
          model_2.add(Conv2D(64, (3, 3)))
          model_2.add(Activation('relu'))
          model_2.add(MaxPooling2D(pool_size=(2, 2)))
          model_2.add(Dropout(0.25))

          model_2.add(Flatten())
          model_2.add(Dense(512))
          model_2.add(Activation('relu'))
          model_2.add(Dropout(0.5))
          model_2.add(Dense(num_classes))
          model_2.add(Activation('softmax'))
```

```
In [174]: model_2.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 30, 40, 32)        416
_________________________________________________________________
activation_17 (Activation)   (None, 30, 40, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 30, 40, 32)        12320
_________________________________________________________________
activation_18 (Activation)   (None, 30, 40, 32)        0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 10, 10, 32)        0
_________________________________________________________________
dropout_8 (Dropout)          (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 10, 10, 64)        18496
_________________________________________________________________
activation_19 (Activation)   (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 8, 8, 64)          36928
_________________________________________________________________
activation_20 (Activation)   (None, 8, 8, 64)          0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_9 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_12 (Dense)             (None, 512)               524800
_________________________________________________________________
activation_21 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_10 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_13 (Dense)             (None, 62)                31806
_________________________________________________________________
activation_22 (Activation)   (None, 62)                0
=================================================================
Total params: 624,766
Trainable params: 624,766
Non-trainable params: 0
_________________________________________________________________
```

```
In [175]: batch_size = 32

          # initiate RMSprop optimizer
          opt_2 = keras.optimizers.RMSprop(lr=0.0005)

          # Let's train the model using RMSprop
          model_2.compile(loss='categorical_crossentropy',
                        optimizer=opt_2,
                        metrics=['accuracy'])
```

```
In [176]: x_train=x_train.reshape((x_train.shape[0],)+input_shape)
          x_test=x_test.reshape((x_test.shape[0],)+input_shape)
```

```
In [179]: t = now()
          model_2.fit(x_train, y_train,
                        batch_size=batch_size,
                        epochs=30,
                        validation_data=(x_test, y_test),
                        shuffle=True)
          print('Training time: %s' % (now() - t))
```

```
Epoch 1/30
80/80 [==============================] - 6s 70ms/step - loss: 1.2305 - accuracy: 0.6277 - val_loss:
1.1017 - val_accuracy: 0.6917
Epoch 2/30
80/80 [==============================] - 5s 67ms/step - loss: 1.1552 - accuracy: 0.6461 - val_loss:
1.0788 - val_accuracy: 0.6917
Epoch 3/30
80/80 [==============================] - 5s 67ms/step - loss: 1.0053 - accuracy: 0.6899 - val_loss:
1.0149 - val_accuracy: 0.7069
Epoch 4/30
80/80 [==============================] - 5s 68ms/step - loss: 0.9632 - accuracy: 0.7004 - val_loss:
0.9579 - val_accuracy: 0.7128
Epoch 5/30
80/80 [==============================] - 6s 72ms/step - loss: 0.8756 - accuracy: 0.7223 - val_loss:
0.9342 - val_accuracy: 0.7280
Epoch 6/30
80/80 [==============================] - 5s 69ms/step - loss: 0.8299 - accuracy: 0.7368 - val_loss:
0.9239 - val_accuracy: 0.7245
Epoch 7/30
80/80 [==============================] - 5s 68ms/step - loss: 0.7681 - accuracy: 0.7450 - val_loss:
0.8713 - val_accuracy: 0.7351
Epoch 8/30
80/80 [==============================] - 5s 69ms/step - loss: 0.7284 - accuracy: 0.7673 - val_loss:
0.8935 - val_accuracy: 0.7444
Epoch 9/30
80/80 [==============================] - 5s 68ms/step - loss: 0.6816 - accuracy: 0.7888 - val_loss:
0.8722 - val_accuracy: 0.7515
Epoch 10/30
80/80 [==============================] - 6s 70ms/step - loss: 0.6348 - accuracy: 0.8045 - val_loss:
0.8240 - val_accuracy: 0.7597
Epoch 11/30
80/80 [==============================] - 6s 71ms/step - loss: 0.5816 - accuracy: 0.8080 - val_loss:
0.9016 - val_accuracy: 0.7339
Epoch 12/30
80/80 [==============================] - 6s 71ms/step - loss: 0.5470 - accuracy: 0.8162 - val_loss:
0.8006 - val_accuracy: 0.7655
Epoch 13/30
80/80 [==============================] - 5s 67ms/step - loss: 0.5308 - accuracy: 0.8314 - val_loss:
0.8131 - val_accuracy: 0.7620
Epoch 14/30
80/80 [==============================] - 5s 68ms/step - loss: 0.5298 - accuracy: 0.8197 - val_loss:
0.7980 - val_accuracy: 0.7585
Epoch 15/30
80/80 [==============================] - 5s 68ms/step - loss: 0.4866 - accuracy: 0.8346 - val_loss:
0.7957 - val_accuracy: 0.7655
Epoch 16/30
80/80 [==============================] - 5s 68ms/step - loss: 0.4581 - accuracy: 0.8479 - val_loss:
0.8082 - val_accuracy: 0.7796
Epoch 17/30
80/80 [==============================] - 5s 68ms/step - loss: 0.4099 - accuracy: 0.8612 - val_loss:
0.8022 - val_accuracy: 0.7644
Epoch 18/30
80/80 [==============================] - 5s 68ms/step - loss: 0.4255 - accuracy: 0.8557 - val_loss:
0.8055 - val_accuracy: 0.7691
Epoch 19/30
80/80 [==============================] - 5s 69ms/step - loss: 0.3849 - accuracy: 0.8686 - val_loss:
0.7738 - val_accuracy: 0.7855
Epoch 20/30
80/80 [==============================] - 5s 68ms/step - loss: 0.3844 - accuracy: 0.8580 - val_loss:
0.7779 - val_accuracy: 0.7737
Epoch 21/30
80/80 [==============================] - 5s 68ms/step - loss: 0.3500 - accuracy: 0.8827 - val_loss:
0.7604 - val_accuracy: 0.7831
Epoch 22/30
80/80 [==============================] - 6s 71ms/step - loss: 0.3509 - accuracy: 0.8819 - val_loss:
0.8023 - val_accuracy: 0.7761
Epoch 23/30
80/80 [==============================] - 6s 70ms/step - loss: 0.3437 - accuracy: 0.8827 - val_loss:
0.7478 - val_accuracy: 0.7878
Epoch 24/30
80/80 [==============================] - 6s 70ms/step - loss: 0.3078 - accuracy: 0.8882 - val_loss:
0.7946 - val_accuracy: 0.7866
Epoch 25/30
80/80 [==============================] - 5s 68ms/step - loss: 0.3084 - accuracy: 0.8901 - val_loss:
0.8030 - val_accuracy: 0.7843
Epoch 26/30
80/80 [==============================] - 6s 69ms/step - loss: 0.2950 - accuracy: 0.8921 - val_loss:
0.8147 - val_accuracy: 0.7691
Epoch 27/30
80/80 [==============================] - 6s 70ms/step - loss: 0.2802 - accuracy: 0.9042 - val_loss:
0.7880 - val_accuracy: 0.7749
Epoch 28/30
80/80 [==============================] - 5s 68ms/step - loss: 0.2687 - accuracy: 0.9034 - val_loss:
0.8032 - val_accuracy: 0.7866
Epoch 29/30
80/80 [==============================] - 5s 69ms/step - loss: 0.2654 - accuracy: 0.9147 - val_loss:
0.8331 - val_accuracy: 0.7866
Epoch 30/30
80/80 [==============================] - 5s 68ms/step - loss: 0.2674 - accuracy: 0.9081 - val_loss:
0.8297 - val_accuracy: 0.7937
Training time: 0:02:47.210564
```

```
In [180]: score = model_2.evaluate(x_test, y_test, verbose=0)
          print('Test score:', score[0])
          print('Test accuracy:', score[1])
```

```
Test score: 0.8297411799430847
Test accuracy: 0.7936694025993347
```

4. Conclusion

This task involves in recognizing hand wirtten english characters. There are more than 60 categories. We know that it is more difficult compared to MNIST digit data classification. For example, z may look like 2 or B may look like 8 and so on. From the analysis above, we found that convolution neural network can achieve very high accuracy for image classification. Each pixel can be treated as one feature. The relative location and value of each pixel is very important for image. The final accuracy in test set is close to 80% while the training accuracy is 90%. It indicates there may be some overfitting. This model is still very simple, using deep neural network to build more layers and using dropout regularization may help to achieve higher accuracy.

In this course, we learn different deep learning architecture. They are suitable for different task.