Digit classification using Neural Network

Too Hon Lin

October 2020

1

Figure 1 shows the algorithm to train the neural network for digit recognition. The network has 2 layers, where the 1st later has 512 nodes and the 2nd layer has 128 nodes. The test accuracy is 0.9808 and the test loss is 0.0768.

```
Created on Mon Oct 12 14:34:03 2020

@author: honlin
"""
import tensorflow as tf

mnist=tf.keras.datasets.mnist
(x_train,y_train),(x_test,y_test) = mnist.load_data()
x_train,x_test = x_train /255.0, x_test/255.0
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512,activation=tf.nn.relu) , #first layer, 512 nodes and relu activation function
    tf.keras.layers.Dropout(0.2), #randomly drop out 20% of the nodes
    tf.keras.layers.Dense(128,activation=tf.nn.relu), #2nd layer, 128 nodes and relu activation function
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10,activation=tf.nn.softmax)
    ])
model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])

model.fit(x_train,y_train,epochs = 10) #epoch = full cycles
#evaluate the neural network
model.evaluate(x_test,y_test)
```

Figure 1

I tested the trained model with 12 different handwritten digits(refer to Figure 2).

Table 1 shows the prediction by the trained model. The trained model doesn't perform well when the handwriting is slanted/tilted. The trained model has difficulties in distinguishing 2 and 7, and 0, 6 and 8. This is probable due to the inability of the neural network to distinguish the important features that distinguish these numbers. To resolve this issue, I resort to convolutional neural network. Figure 3 shows the algorithm to train a convolutional neural network for digit recognition.

The trained CNN has a test accuracy of 0.9923 and a test loss of 0.02636, which is significantly better than the neural network. Table 2 shows the prediction made by the trained convolutional neural network. From the table, we can see a major improvement on the network. The network can now identify 0, 2 and 8 better. It can also identify the slanted digits better, which is shown from the slanted 2 and 3 examples.

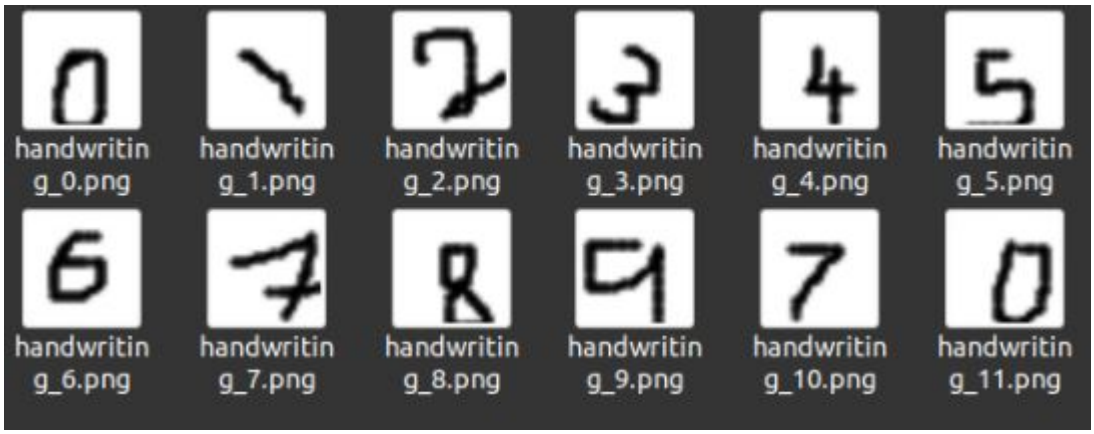

Figure 2: Handwritten test image. handwriting_1, handwriting_2 and handwriting_3 are slanted digits, while handwritten_7 and handwriting_10 are 7 written in different form.

Image's index	Label	Prediction	Probability	2nd Prediction	Probability
0	0	6	0.524149	8	0.3607
1	1	5	0.721855	8	0.231614
2	2	7	0.996348	2	0.00240613
3	3	5	0.594806	3	0.308254
4	4	4	0.796965	6	0.111537
5	5	5	0.99896	8	0.000507094
6	6	4	0.653653	9	0.24941
7	7	7	0.999937	4	0.0000004
8	8	1	0.819028	8	0.173392
9	9	9	0.64866	7	0.349739
10	7	1	0.440437	7	0.371776
11	0	8	0.548215	6	0.267722

Table 1: Prediction made by neural network.

Image's index	Label	Prediction	Probability	2nd Prediction	Probability
0	0	6	0.540577	0	0.447091
1	1	5	0.713493	2	0.241935
2	2	2	0.639921	7	0.360071
3	3	3	0.520768	5	0.448841
4	4	4	0.958269	1	0.0177604
5	5	5	0.998505	9	0.000775782
6	6	9	0.999951	5	0.00004512
7	7	7	0.982772	2	0.0168164
8	8	8	0.75659	2	0.204832
9	9	9	0.998384	7	0.00152264
10	7	2	0.690768	7	0.120752
11	0	0	0.592431	6	0.341117

Table 2: Prediction made by convolutional neural network.

```
#load data
mnist=tf.keras.datasets.mnist
(x_train,y_train),(x_test,y_test) = mnist.load_data()

#save in the right format for CNN input
x_train = x_train.reshape(60000,28,28,1)
x_test = x_test.reshape(10000,28,28,1)

#normalize
x_train = np.array(x_train,dtype = np.float64)
x_test = np.array(x_test,dtype= np.float64)
x_train /= 255
x_test /=255

#build cnn
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation=tf.nn.relu, input_shape=(28, 28, 1)), #image size
    tf.keras.layers.Conv2D(64, (3, 3), activation=tf.nn.relu), #2nd layer
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)) ,
    tf.keras.layers.Dropout(0.25),##end of convolutional layer

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128,activation = tf.nn.relu),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(10,activation = tf.nn.softmax)
    ])

#compile
model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])

batch_size = 128
epochs = 10

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=2,
          validation_data=(x_test, y_test))

score = model.evaluate(x_test, y_test, verbose=0)
```

Figure 3