Machine Learning Model

February 17, 2020

[6]:
```
import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train,y_train), (x_test,y_test) = mnist.load_data()

x_train = tf.keras.utils.normalize(x_train, axis = 1)
x_test = tf.keras.utils.normalize(x_test, axis = 1)

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))
model.add(tf.keras.layers.Dense(128,activation = tf.nn.relu))
model.add(tf.keras.layers.Dense(128,activation = tf.nn.relu))
model.add(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 = 3)
```

```
Epoch 1/3
60000/60000 [==============================] - 14s 227us/sample - loss: 0.2662 -
acc: 0.9222 - loss: 0.2736 - acc
Epoch 2/3
60000/60000 [==============================] - 13s 225us/sample - loss: 0.1068 -
acc: 0.9670
Epoch 3/3
60000/60000 [==============================] - 14s 226us/sample - loss: 0.0731 -
acc: 0.9768
```

[6]:
```
<tensorflow.python.keras.callbacks.History at 0x7fe6781012d0>
```

[7]:
```
val_loss, val_acc = model.evaluate(x_test,y_test)
print(val_loss,val_acc)
```

```
10000/10000 [==============================] - 1s 70us/sample - loss: 0.0919 -
acc: 0.9712
0.09194680654853582 0.9712
```

[15]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[9]:
```
model.save('Classifier.model')
```

[10]:
```
new_model = tf.keras.models.load_model('Classifier.model')
```

[11]:
```
predictions = new_model.predict([x_test])
```

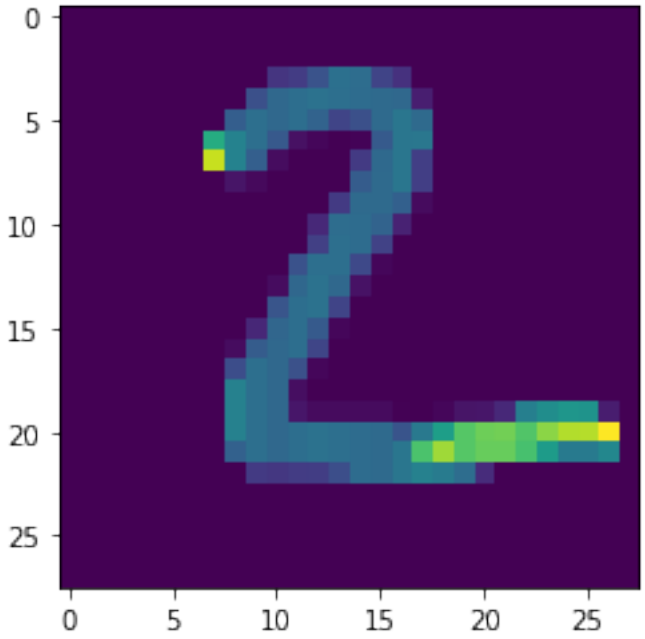
[12]:
```

```

```
[[1.0076888e-09 6.8337727e-08 4.6345976e-06 … 9.9996781e-01
  4.4685420e-09 3.9225264e-07]
 [7.5624040e-09 8.5645197e-06 9.9999082e-01 … 3.0634194e-08
  4.7639322e-08 2.1509040e-12]
 [9.5370092e-07 9.9908614e-01 2.0158912e-04 … 2.1512306e-04
  4.2552181e-04 3.8558403e-07]
 …
 [3.9838202e-08 9.5543328e-07 2.0274273e-07 … 7.9759433e-05
  2.8028571e-05 7.7852244e-05]
 [4.9068086e-07 1.8736263e-06 4.1946528e-08 … 2.0806735e-07
  1.2828168e-04 7.7092260e-10]
 [7.0595001e-07 2.3227467e-09 1.9458500e-07 … 1.5603674e-09
  7.1361512e-07 1.1532122e-09]]
```

[17]:
```
print (np.argmax(predictions[1]))
```

```
2
```

[18]:
```
plt.imshow(x_test[1])
plt.show()
```


0
5
10
15
20
25
0
5
10
15
20
25

[ ]: