In [1]:
```
# Official tutorials: https://www.tensorflow.org/tutorials
# keras totutial for MNIST: https://www.tensorflow.org/tutorials/qu
```

In [2]:
```
import numpy as np
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

```
WARNING:tensorflow:From /opt/anaconda3/lib/python3.7/site-packages
/tensorflow_core/python/compat/v2_compat.py:65: disable_resource_v
ariables (from tensorflow.python.ops.variable_scope) is deprecated
and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

In [3]:
```
# Build an easy calculator
a = tf.placeholder(dtype=tf.float32, shape=[3,3])
b = tf.placeholder(dtype=tf.float32, shape=[3,3])
c = a+b
d = tf.matmul(a, b)
print(a)
print(b)
print(c)
print(d)
```

```
Tensor("Placeholder:0", shape=(3, 3), dtype=float32)
Tensor("Placeholder_1:0", shape=(3, 3), dtype=float32)
Tensor("add:0", shape=(3, 3), dtype=float32)
Tensor("MatMul:0", shape=(3, 3), dtype=float32)
```

In [4]:
```
sess = tf.Session()
a_input = np.array([[1,1,1],[2,2,2],[3,3,3]])
b_input = np.array([[1,2,3],[1,2,3],[1,2,3]])
my_feed_dict = {a: a_input, b: b_input}
res = sess.run([c,d], feed_dict=my_feed_dict)
print(res[0])
print(res[1])
```

```
[[2. 3. 4.]
 [3. 4. 5.]
 [4. 5. 6.]]
[[ 3.  6.  9.]
 [ 6. 12. 18.]
 [ 9. 18. 27.]]
```

In [5]:
```
e = tf.Variable(0.0)
e_add = tf.assign(e, e+1)
```

In [ ]:
```
print(sess.run(e))
```

In [6]:
```
sess.run(tf.global_variables_initializer())
print(sess.run(e))
sess.run(e_add)
print(sess.run(e))
```

```
0.0
1.0
```

In [7]:
```
# build an easy neuron network
# load in the data
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
print(x_train.shape)
print(y_train.shape)
```

```
(60000, 28, 28)
(60000,)
```

In [ ]:
```
import matplotlib.pyplot as plt
plt.imshow(x_train[0], cmap='gray')
```

In [8]:
```
# define structure: 784-->256-->10
input_img = tf.placeholder(dtype=tf.float32, shape=[None, 28*28], n
labels = tf.placeholder(dtype=tf.int32, shape=[None], name='label')
h1 = tf.layers.dense(input_img, units=256, name='h1')
h1 = tf.nn.relu(h1)
h2 = tf.layers.dense(h1, units=10, name='h2')
output = tf.nn.softmax(h2)
print(h1.shape)
print(h2.shape)
print(output.shape)
print(labels.shape)
```

```
WARNING:tensorflow:From <ipython-input-8-509994677059>:4: dense (f
rom tensorflow.python.layers.core) is deprecated and will be remov
ed in a future version.
Instructions for updating:
Use keras.layers.Dense instead.
WARNING:tensorflow:From /opt/anaconda3/lib/python3.7/site-packages
/tensorflow_core/python/layers/core.py:187: Layer.apply (from tens
orflow.python.keras.engine.base_layer) is deprecated and will be r
emoved in a future version.
Instructions for updating:
Please use `layer.__call__` method instead.
(?, 256)
(?, 10)
(?, 10)
(?,)
```

In [9]:
```
# define loss and optimizer
loss = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=labels
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
update = optimizer.minimize(loss)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

In [20]:
```
epoch = 100
epoch_accuracies = []
num_iter = 180
start = 0
for j in range (epoch):
    print (j)
    epoch_accuracy = 0.0
    for it in range (num_iter):
        start += 10
        start %= 60000
        cur_input = np.reshape(x_train[start:(start+10)], (10, 784)
        cur_label = y_train[start:(start+10)]
        my_feed_dict = {input_img: cur_input, labels:cur_label}
        preds,_ = sess.run([output, update], feed_dict=my_feed_dict
        preds_label = np.argmax(preds, axis=1)
        acc_iter = np.sum(1*(preds_label)==(cur_label))/10
        epoch_accuracy += acc_iter
    epoch_accuracy =  epoch_accuracy/180
    epoch_accuracies.append(epoch_accuracy)
    print (epoch_accuracy)
```

```
0
0.7461111111111107
1
0.7594444444444447
2
0.7499999999999993
3
0.7572222222222221
4
0.7383333333333335
5
0.762777777777778
6
0.7555555555555555
7
0.7394444444444441
8
0.7411111111111112
9
0.7522222222222229
10
0.7616666666666672
11
0.7494444444444444
12
```

```
0.7638888888888888
13
0.7483333333333337
14
0.752777777777778
15
0.7488888888888888
16
0.7538888888888892
17
0.7383333333333334
18
0.766111111111111
19
0.7455555555555555
20
0.7433333333333335
21
0.7455555555555552
22
0.805555555555556
23
0.8122222222222225
24
0.8377777777777782
25
0.821111111111112
26
0.840555555555556
27
0.8222222222222227
28
0.8416666666666675
29
0.8755555555555566
30
0.932777777777779
31
0.9388888888888897
32
0.9538888888888899
33
0.9438888888888902
34
0.9477777777777787
35
0.9500000000000006
36
0.9494444444444455
37
0.9372222222222237
38
0.931666666666668
39
```

```
0.9516666666666677
40
0.9361111111111122
41
0.9272222222222228
42
0.9372222222222236
43
0.9505555555555563
44
0.948333333333335
45
0.9405555555555569
46
0.945555555555557
47
0.9505555555555565
48
0.9377777777777789
49
0.9488888888888898
50
0.9366666666666679
51
0.9511111111111121
52
0.9516666666666677
53
0.956111111111112
54
0.9416666666666677
55
0.9483333333333344
56
0.9400000000000012
57
0.9538888888888899
58
0.9411111111111125
59
0.9372222222222231
60
0.9455555555555566
61
0.9461111111111121
62
0.9455555555555561
63
0.9533333333333343
64
0.9611111111111124
65
0.9738888888888896
```

```
66
0.9677777777777786
67
0.9577777777777786
68
0.9666666666666675
69
0.962777777777779
70
0.961111111111112
71
0.9411111111111125
72
0.9677777777777785
73
0.9522222222222234
74
0.9505555555555566
75
0.9577777777777787
76
0.9577777777777787
77
0.9633333333333342
78
0.9566666666666676
79
0.9516666666666677
80
0.962222222222223
81
0.9516666666666677
82
0.9594444444444454
83
0.9577777777777791
84
0.9516666666666679
85
0.9666666666666675
86
0.9677777777777783
87
0.9544444444444453
88
0.9650000000000007
89
0.9594444444444457
90
0.9572222222222231
91
0.9555555555555567
92
```

```
0.9550000000000012
93
0.9538888888888899
94
0.9555555555555562
95
0.9644444444444452
96
0.9588888888888898
97
0.9672222222222231
98
0.9744444444444448
99
0.9783333333333337
```

In [ ]: