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 [25]:
```
# define loss and optimizer
loss = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=labels
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.05)
update = optimizer.minimize(loss)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

In [26]:
```
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
0.6516666666666666
1
0.8538888888888901
2
0.8722222222222235
3
0.891666666666668
4
0.8705555555555566
5
0.9055555555555568
6
0.890000000000001
7
0.8761111111111126
8
0.8933333333333345
9
0.9066666666666681
10
0.9344444444444456
11
0.9166666666666677
12
```

```
0.9227777777777788
13
0.9127777777777786
14
0.9238888888888904
15
0.9238888888888904
16
0.9250000000000013
17
0.907222222222224
18
0.9400000000000011
19
0.9344444444444456
20
0.9322222222222234
21
0.9355555555555567
22
0.9466666666666678
23
0.9283333333333347
24
0.9494444444444458
25
0.9205555555555568
26
0.9294444444444456
27
0.92888888888889
28
0.9461111111111125
29
0.9361111111111122
30
0.9455555555555567
31
0.9522222222222231
32
0.9672222222222231
33
0.9422222222222234
34
0.9588888888888898
35
0.9550000000000012
36
0.9577777777777787
37
0.9511111111111124
38
0.9344444444444459
39
```

```
0.9566666666666677
40
0.9283333333333346
41
0.9527777777777786
42
0.9461111111111122
43
0.9583333333333345
44
0.9533333333333349
45
0.9511111111111126
46
0.9538888888888898
47
0.9522222222222232
48
0.9427777777777788
49
0.9505555555555568
50
0.9472222222222236
51
0.9516666666666677
52
0.9594444444444452
53
0.9561111111111124
54
0.9544444444444454
55
0.9572222222222231
56
0.9500000000000011
57
0.9527777777777788
58
0.950000000000001
59
0.93888888888889
60
0.9533333333333343
61
0.9516666666666675
62
0.9522222222222232
63
0.9544444444444453
64
0.9627777777777788
65
0.9755555555555562
```

```
66
0.9677777777777785
67
0.9566666666666676
68
0.9666666666666673
69
0.9577777777777791
70
0.9616666666666678
71
0.9472222222222236
72
0.9688888888888897
73
0.9611111111111117
74
0.9572222222222232
75
0.9666666666666676
76
0.9616666666666676
77
0.960000000000001
78
0.9605555555555563
79
0.9544444444444449
80
0.9672222222222231
81
0.9527777777777786
82
0.956111111111112
83
0.9644444444444453
84
0.9583333333333341
85
0.9727777777777782
86
0.9611111111111119
87
0.9572222222222234
88
0.9616666666666676
89
0.9644444444444453
90
0.9616666666666676
91
0.9650000000000009
92
```

```
0.9505555555555563
93
0.9627777777777788
94
0.9605555555555564
95
0.9655555555555563
96
0.9700000000000006
97
0.9694444444444452
98
0.9716666666666671
99
0.9794444444444447
```

In [ ]: