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 [21]:
```
# define loss and optimizer
loss = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=labels
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
update = optimizer.minimize(loss)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

In [22]:
```
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
0.6516666666666671
1
0.8288888888888899
2
0.8172222222222226
3
0.8722222222222235
4
0.8466666666666675
5
0.8777777777777789
6
0.8750000000000012
7
0.8444444444444454
8
0.8777777777777789
9
0.8833333333333339
10
0.9088888888888904
11
0.8883333333333348
12
```

```
0.8927777777777791
13
0.9000000000000011
14
0.896111111111112
15
0.8827777777777791
16
0.8900000000000007
17
0.8794444444444453
18
0.9094444444444455
19
0.9066666666666681
20
0.8961111111111123
21
0.877777777777778
22
0.9061111111111123
23
0.8816666666666682
24
0.9133333333333348
25
0.8716666666666677
26
0.8955555555555568
27
0.875555555555557
28
0.9050000000000016
29
0.9000000000000012
30
0.9277777777777787
31
0.9300000000000009
32
0.9361111111111118
33
0.9261111111111123
34
0.9277777777777791
35
0.9372222222222231
36
0.9305555555555568
37
0.91388888888889
38
0.9133333333333347
39
```

```
0.9111111111111122
40
0.9127777777777792
41
0.9166666666666676
42
0.9244444444444457
43
0.9394444444444452
44
0.9138888888888906
45
0.9222222222222239
46
0.9338888888888908
47
0.9322222222222234
48
0.9200000000000009
49
0.9233333333333342
50
0.9172222222222236
51
0.9177777777777787
52
0.9372222222222231
53
0.9244444444444456
54
0.9144444444444454
55
0.9316666666666678
56
0.9233333333333347
57
0.9283333333333346
58
0.9172222222222235
59
0.9177777777777788
60
0.9166666666666677
61
0.9183333333333346
62
0.9233333333333342
63
0.9188888888888905
64
0.920000000000001
65
0.9350000000000008
```

```
66
0.9433333333333341
67
0.9272222222222234
68
0.9394444444444455
69
0.9372222222222236
70
0.9244444444444461
71
0.9172222222222237
72
0.9488888888888903
73
0.9216666666666679
74
0.9105555555555567
75
0.9072222222222233
76
0.9222222222222236
77
0.9438888888888903
78
0.9327777777777787
79
0.9238888888888899
80
0.9227777777777791
81
0.9183333333333343
82
0.9400000000000014
83
0.917777777777779
84
0.9094444444444455
85
0.9333333333333342
86
0.9183333333333354
87
0.9250000000000012
88
0.9283333333333349
89
0.9300000000000013
90
0.9327777777777789
91
0.9350000000000008
92
```

```
0.9288888888888904
93
0.9333333333333345
94
0.9188888888888903
95
0.9266666666666682
96
0.9316666666666675
97
0.9411111111111125
98
0.9505555555555565
99
0.9566666666666674
```

In [ ]: