show_results

October 7, 2016

1 Load model

1.1 Model

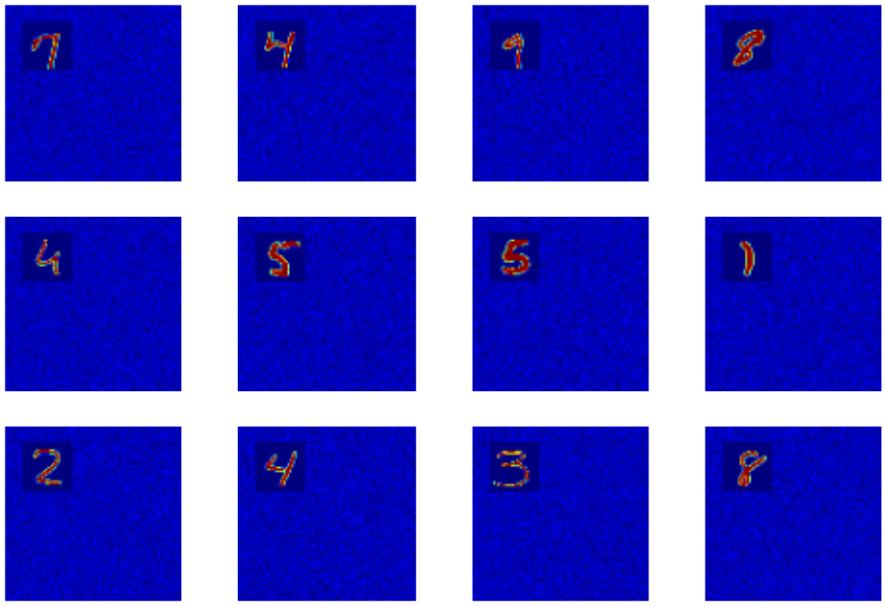
conv1_1 = new_conv_layer(image, [3, 3, 1 , 16], “conv1_1”)
conv1_2 = new_conv_layer(conv1_1, [3, 3, 16, 16], “conv1_2”)
pool1 = tf.nn.max_pool(conv1_2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding=‘SAME’, name=‘pool1’)
conv2_1 = new_conv_layer(pool1, [3, 3, 16, 16], “conv2_1”)
conv2_2 = new_conv_layer(conv2_1, [3, 3, 16, 16], “conv2_2”)
pool2 = tf.nn.max_pool(conv2_2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding=‘SAME’, name=‘pool2’)
conv3_1 = new_conv_layer(pool2, [3, 3, 16, 16], “conv3_1”)
fc1 = self.new_fc_layer(conv3_1, 252516, 512, ‘fc1’)
fc2 = self.new_fc_layer(fc1 , 512 , 10 , ‘fc2’)

1.2 train params

L2 on weights (5e-5)

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import simple_model
        from simple_model import training_generator
        import utils


        simple_model=reload(simple_model)
        lr         = .005
        back_size = 100
        noise      = .1
        crop_pos   = (10,10)

Exception AssertionError: AssertionError("Nesting violated for default stack of <ty


In [2]: """Plot training samples"""
        batch = utils.get_batch('train', im_size=back_size, noise=noise, crop_pos=c
```

```
fig, axs = plt.subplots(3,4)
for ax,img in zip([b for a in axs for b in a],batch[0]):
    ax.imshow(img.reshape(back_size,back_size), vmin=0, vmax=1)
    ax.set_axis_off()
plt.show()
```



```
In [3]: """Perform training"""
        gen = training_generator(lr=lr, back_size=back_size, noise=noise, crop_pos=
        for _ in range(10):
            _,accs = gen.next()
            print "max acc so far : "+str(max(accs)*100)
```

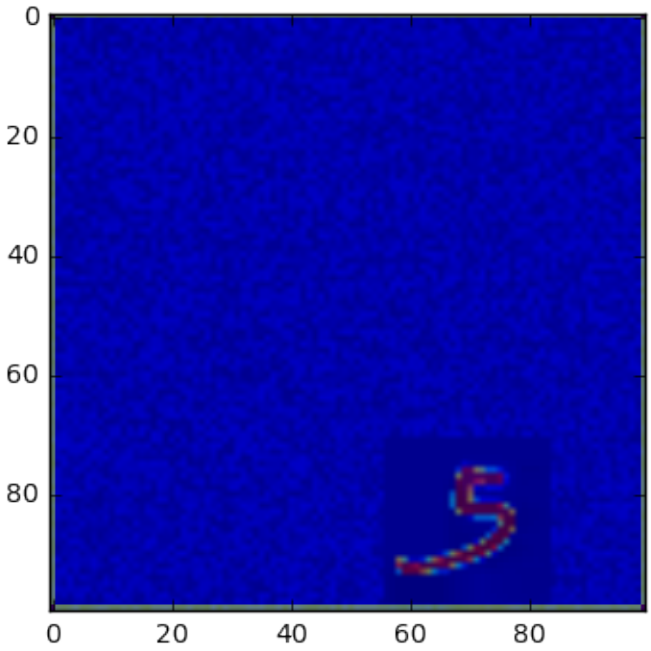
```
***** EPOCH 0 ******
1-Loss on testset is 2.504283
1-Accuracy now is 14.65
2-Loss on testset is 0.429126
2-Accuracy now is 86.83
lr now is 0.00475
max acc so far : 14.65

***** EPOCH 1 ******
1-Loss on testset is 2.924178
1-Accuracy now is 11.60
```

```
2-Loss on testset is 0.154854
2-Accuracy now is 95.45
lr now is 0.00451
max acc so far : 14.65

***** EPOCH 2 ******
1-Loss on testset is 3.231258
1-Accuracy now is 13.40
2-Loss on testset is 0.081570
2-Accuracy now is 98.18
lr now is 0.00429
max acc so far : 14.65

***** EPOCH 3 ******
1-Loss on testset is 3.474708
1-Accuracy now is 11.30
2-Loss on testset is 0.075181
2-Accuracy now is 98.45
lr now is 0.00407
max acc so far : 14.65

***** EPOCH 4 ******
1-Loss on testset is 2.863537
1-Accuracy now is 13.53
2-Loss on testset is 0.066274
2-Accuracy now is 98.50
lr now is 0.00387
max acc so far : 14.65

***** EPOCH 5 ******
1-Loss on testset is 2.609213
1-Accuracy now is 12.55
2-Loss on testset is 0.060298
2-Accuracy now is 98.69
lr now is 0.00368
max acc so far : 14.65

***** EPOCH 6 ******
1-Loss on testset is 3.190901
1-Accuracy now is 12.67
2-Loss on testset is 0.053704
2-Accuracy now is 98.77
lr now is 0.00349
max acc so far : 14.65

***** EPOCH 7 ******
1-Loss on testset is 3.709180
1-Accuracy now is 10.50
```

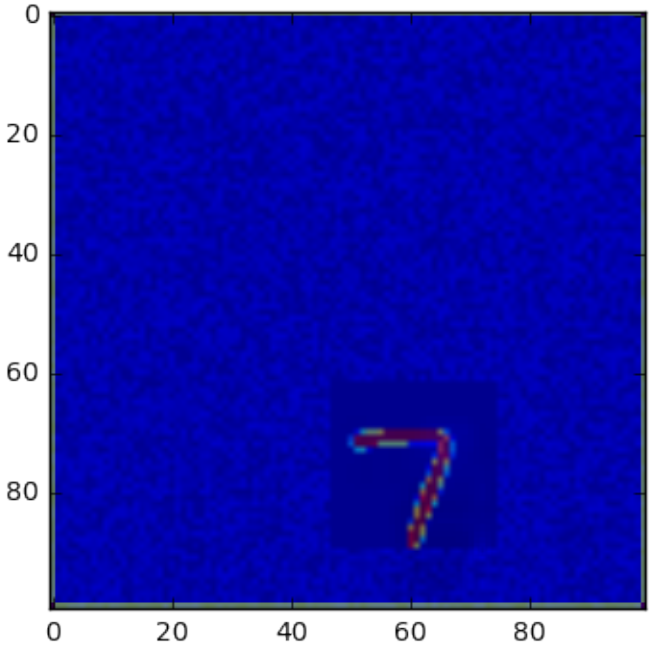
```
2-Loss on testset is 0.058447
2-Accuracy now is 98.79
lr now is 0.00332
max acc so far : 14.65

***** EPOCH 8 ******
1-Loss on testset is 3.059532
1-Accuracy now is 16.03
2-Loss on testset is 0.051990
2-Accuracy now is 99.07
lr now is 0.00315
max acc so far : 16.03

***** EPOCH 9 ******
1-Loss on testset is 2.826725
1-Accuracy now is 11.23
2-Loss on testset is 0.052506
2-Accuracy now is 98.88
lr now is 0.00299
max acc so far : 16.03
```

```
In [7]: back_size  = 100

        for _ in range(25):
            print '----------------'*5
            imgs, lbls = utils.get_batch('test', 1, back_size, .1).next()

            simple_model.show_activation(imgs[0])
```

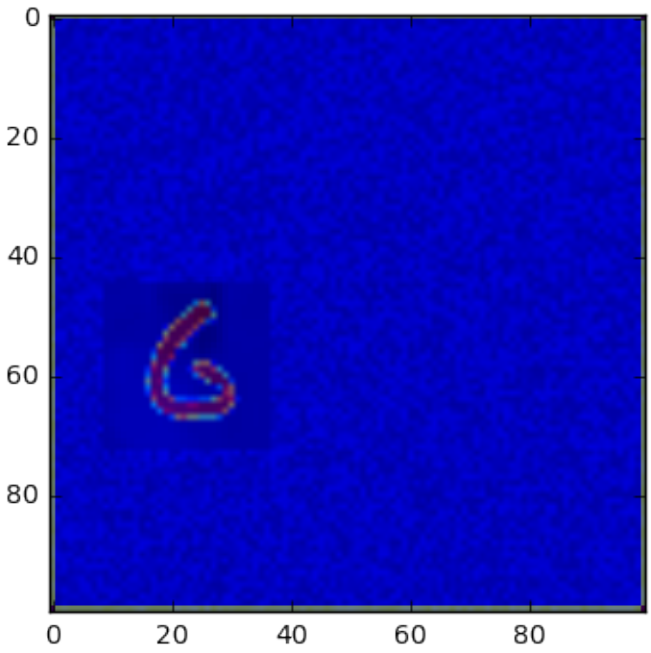
```
--------------------------------------------------------------------------------
prediction is : 1 with 1.220
```



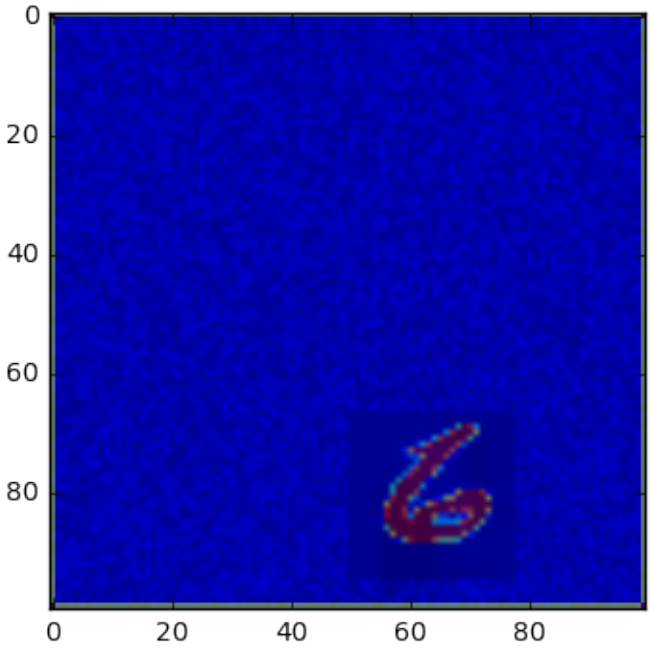
```
prediction is : 1 with 1.242
```



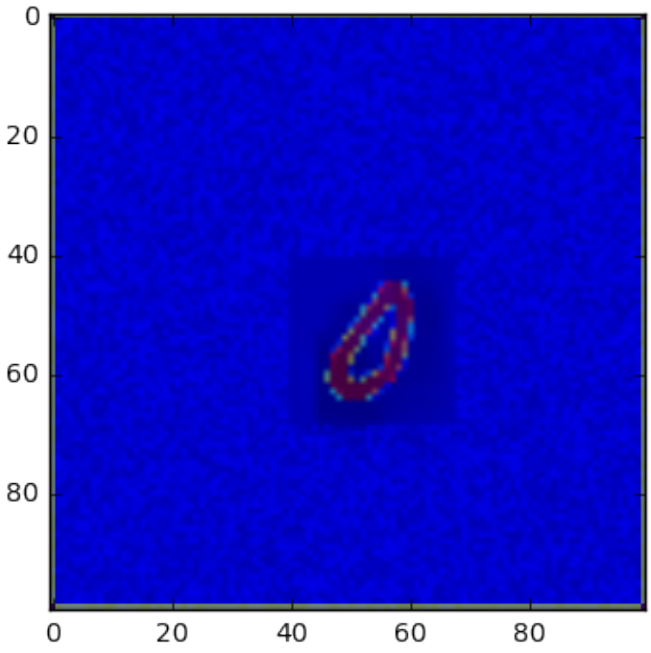
```
--------------------------------------------------------------------------------
prediction is : 1 with 1.271
```



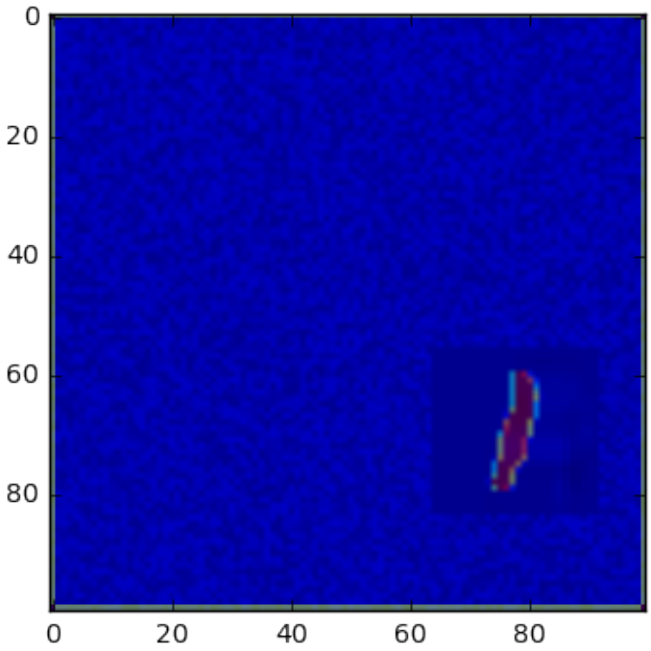

```
--------------------------------------------------------------------------------
prediction is : 1 with 1.227
```



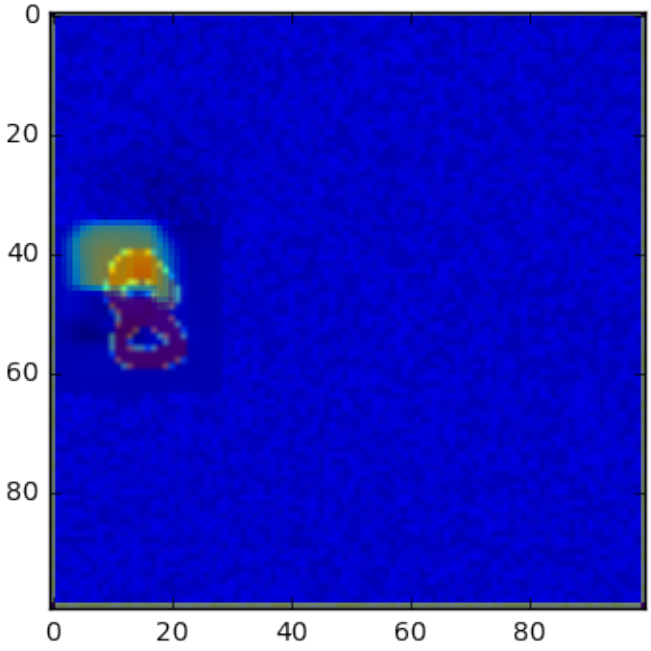
```
--------------------------------------------------------------------------------
prediction is : 1 with 1.119
```



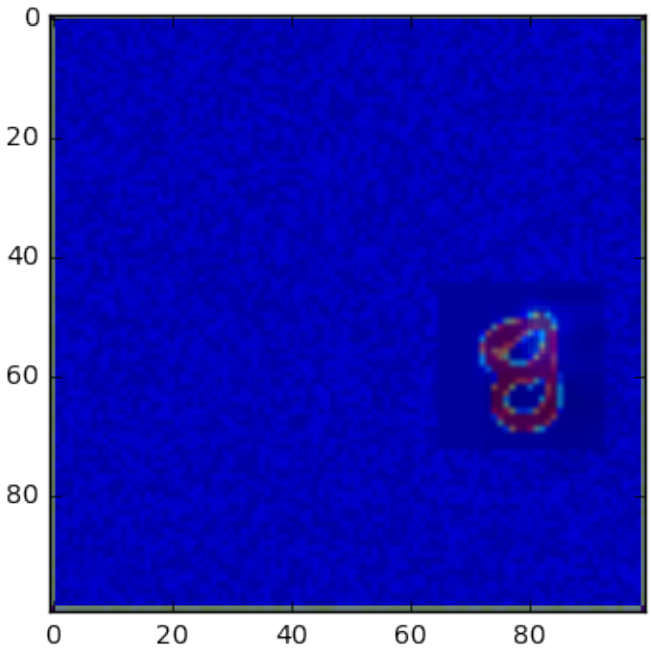

```
--------------------------------------------------------------------------------
prediction is : 1 with 1.297
```



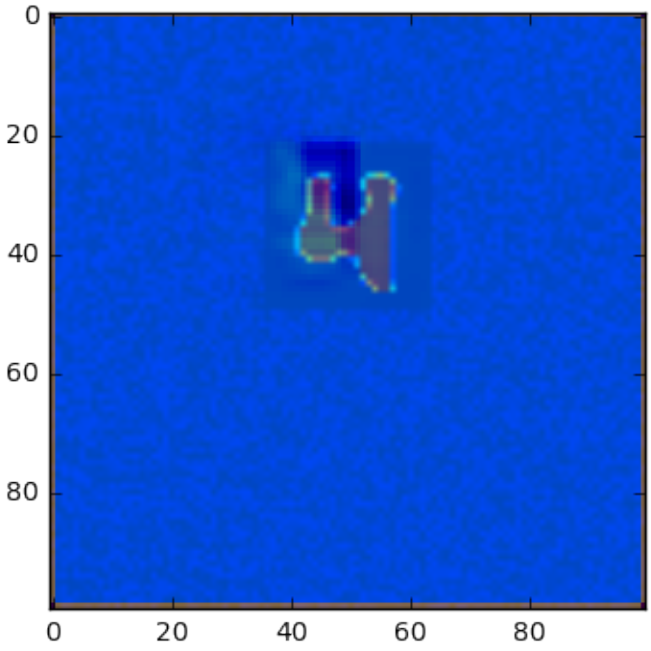
```
prediction is : 1 with 3.012
```



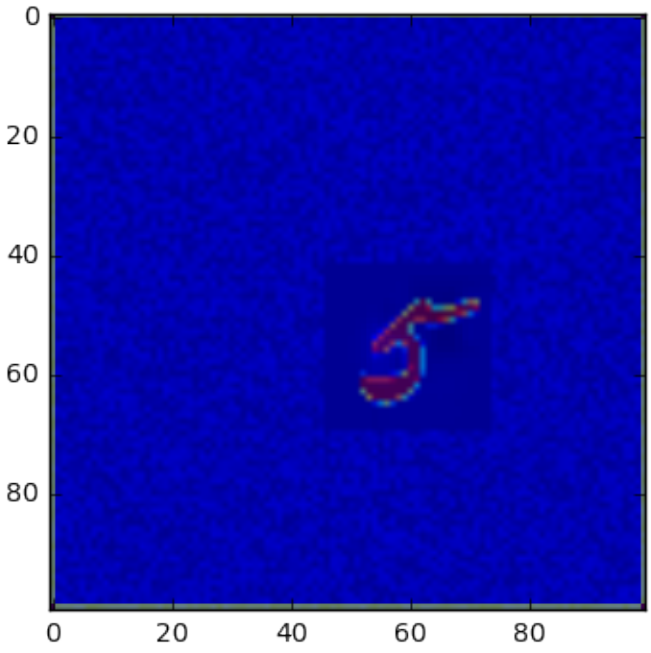
```
--------------------------------------------------------------------------------
prediction is : 1 with 1.207
```



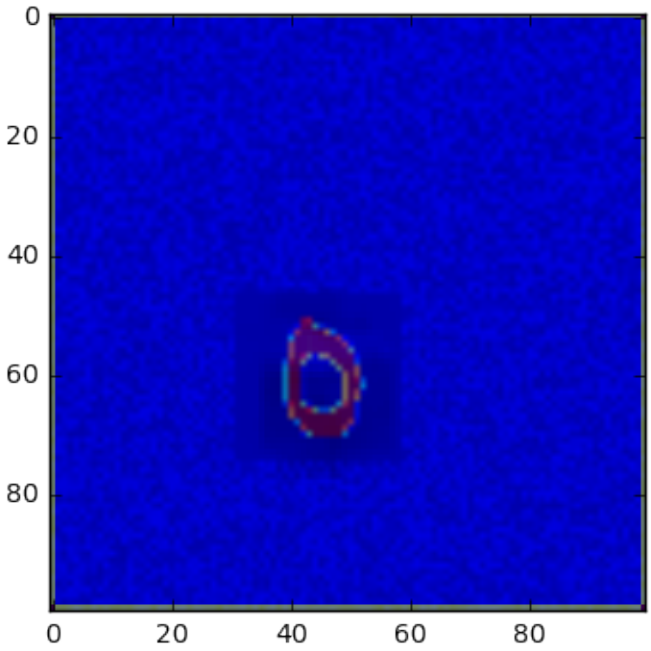

```
--------------------------------------------------------------------------------
prediction is : 3 with 0.920
```



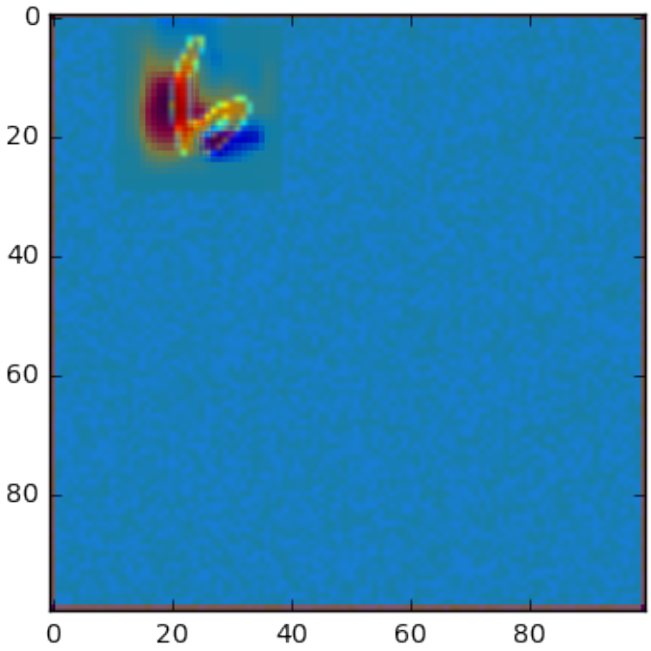
```
--------------------------------------------------------------------------------
prediction is : 1 with 1.254
```



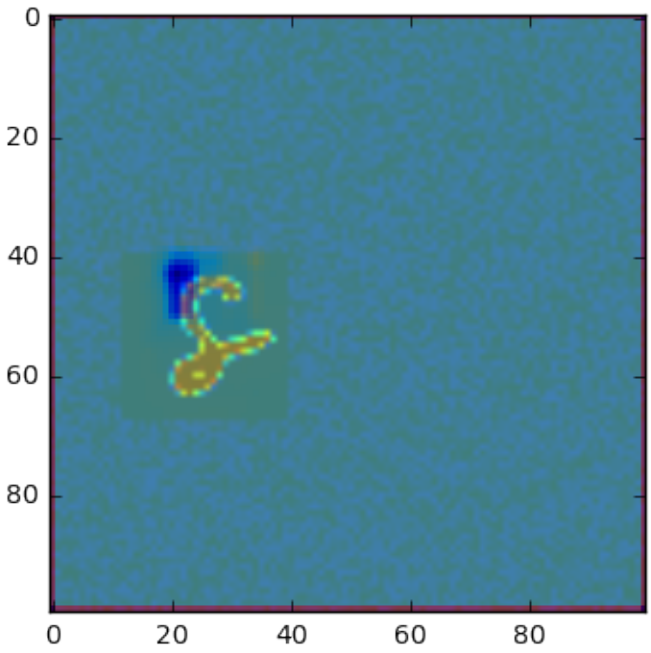

```
--------------------------------------------------------------------------------
prediction is : 1 with 1.102
```



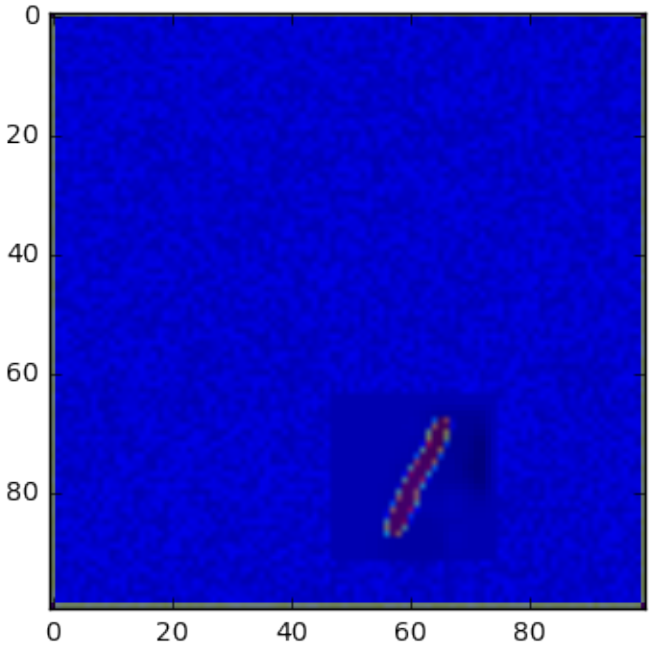
```
prediction is : 5 with 11.696
```



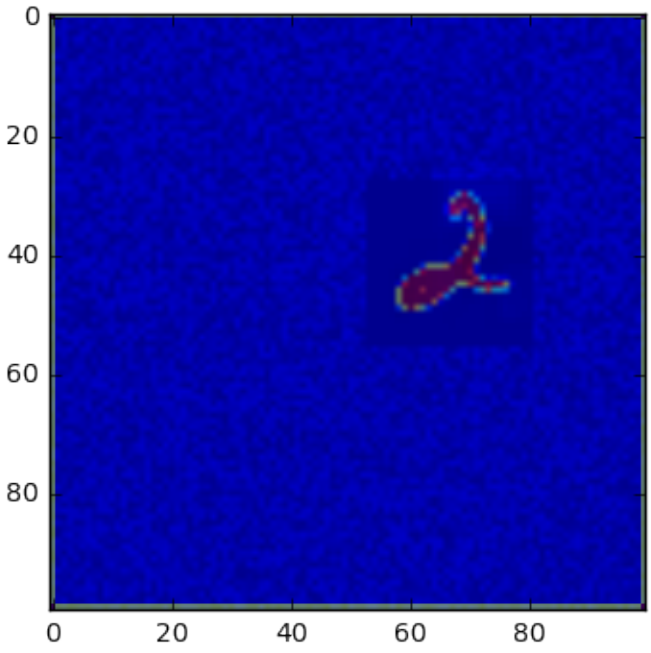
```
--------------------------------------------------------------------------------
prediction is : 1 with 1.174
```



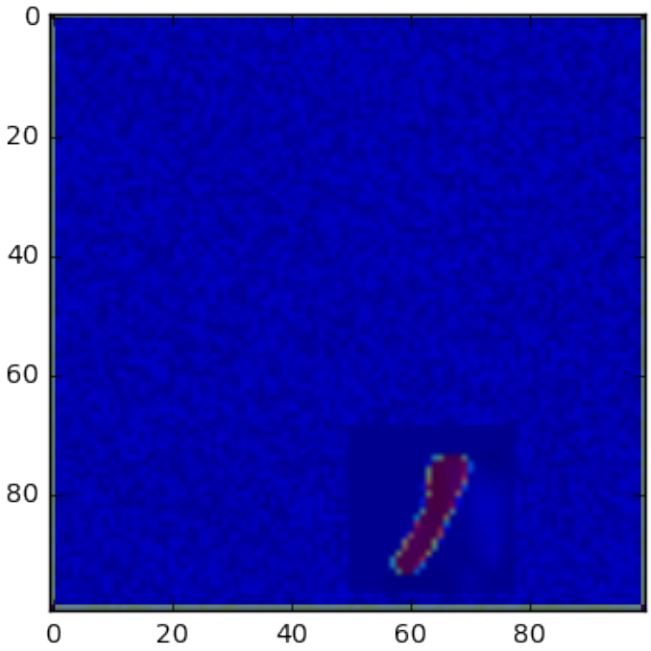

```
--------------------------------------------------------------------------------
prediction is : 1 with 1.210
```



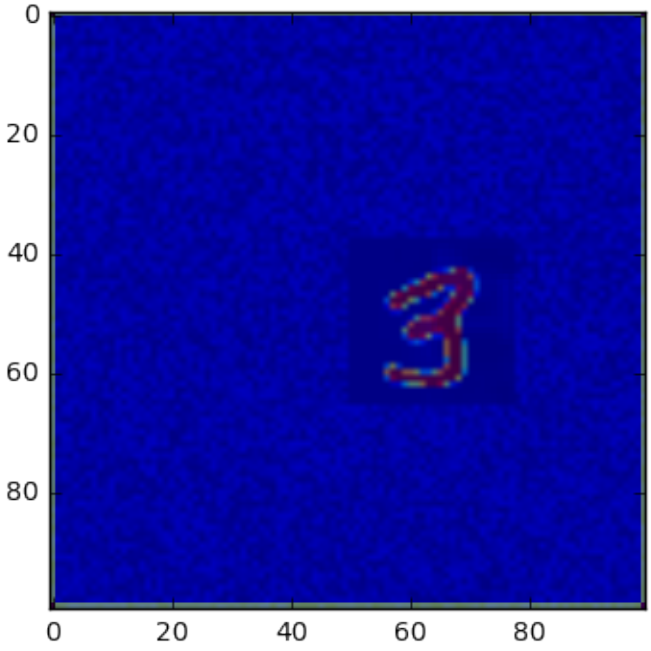
```
--------------------------------------------------------------------------------
prediction is : 1 with 1.249
```



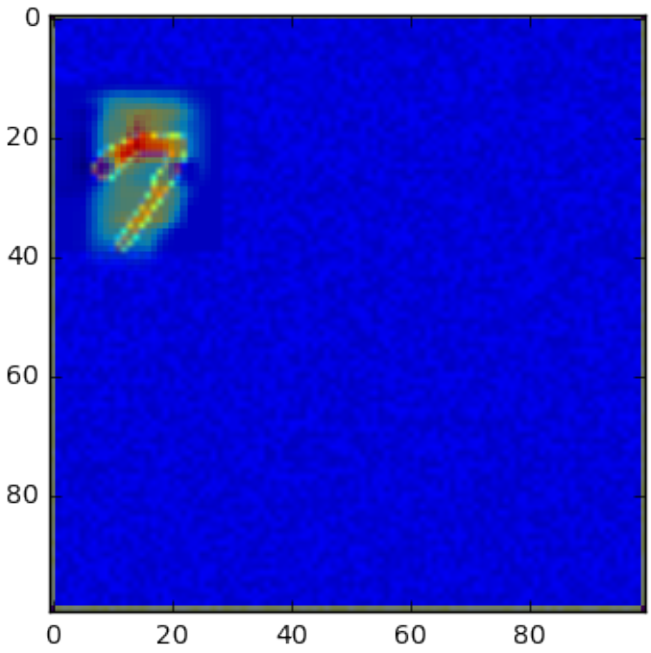

```
--------------------------------------------------------------------------------
prediction is : 1 with 1.250
```



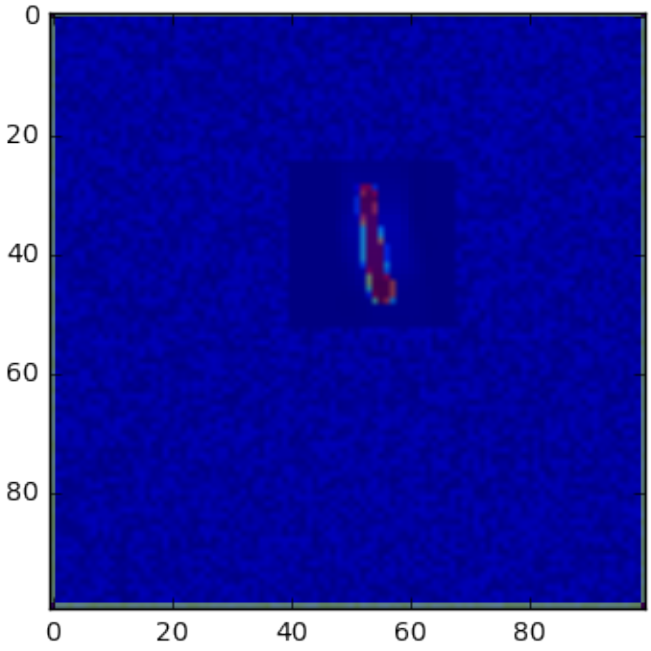
```
prediction is : 1 with 1.269
```



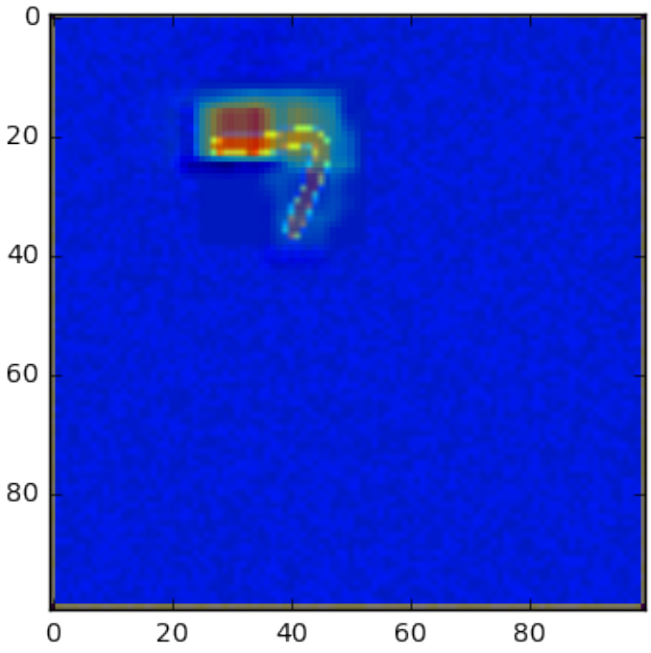
```
--------------------------------------------------------------------------------
prediction is : 7 with 9.108
```



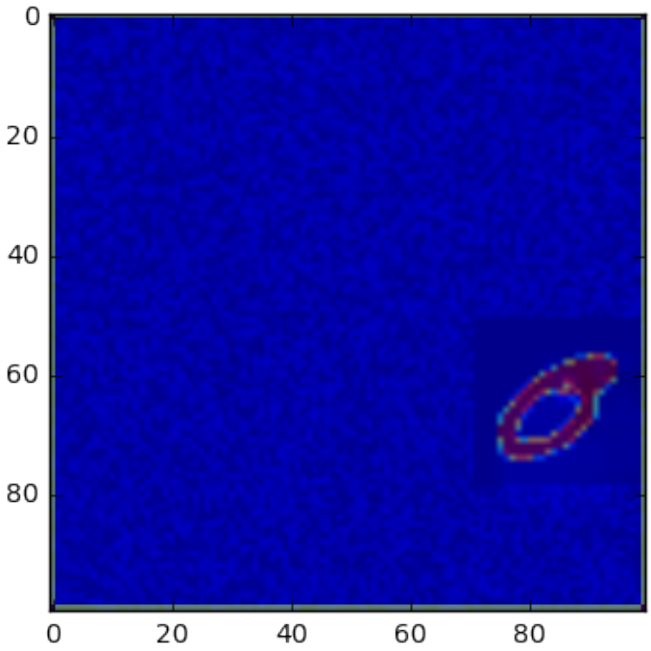

```
--------------------------------------------------------------------------------
prediction is : 3 with 1.036
```



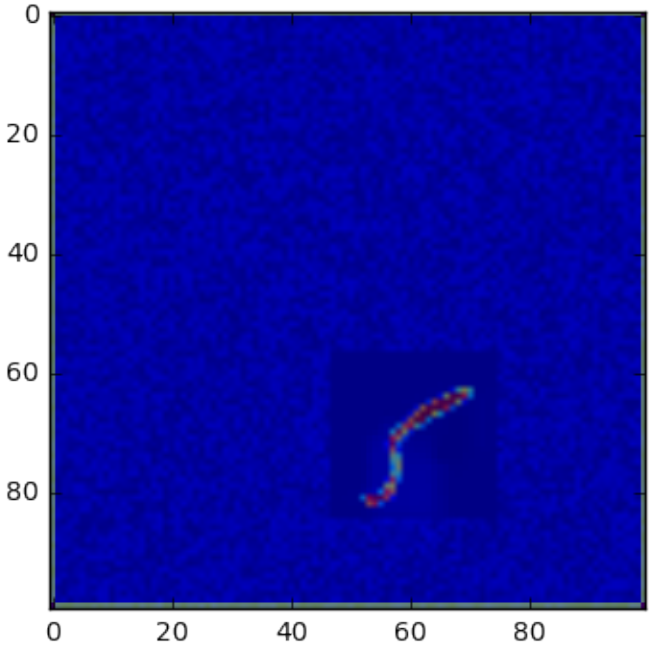
```
--------------------------------------------------------------------------------
prediction is : 5 with 9.448
```



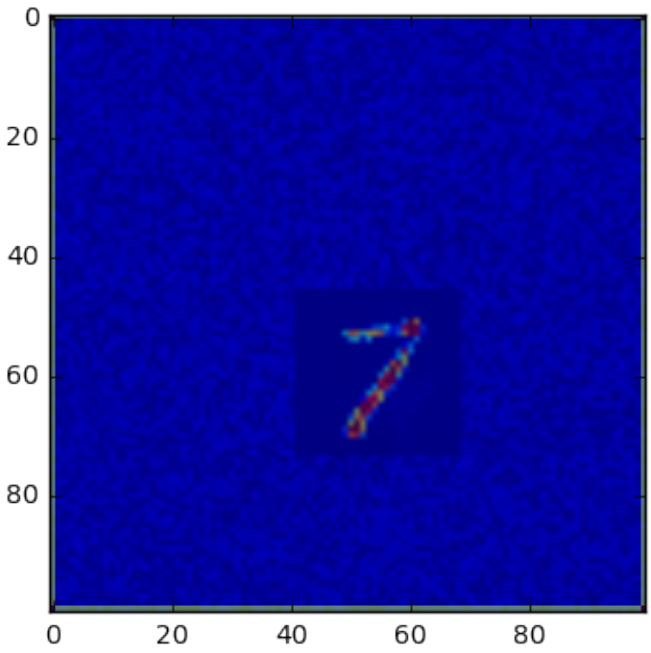

```
--------------------------------------------------------------------------------
prediction is : 1 with 1.334
```



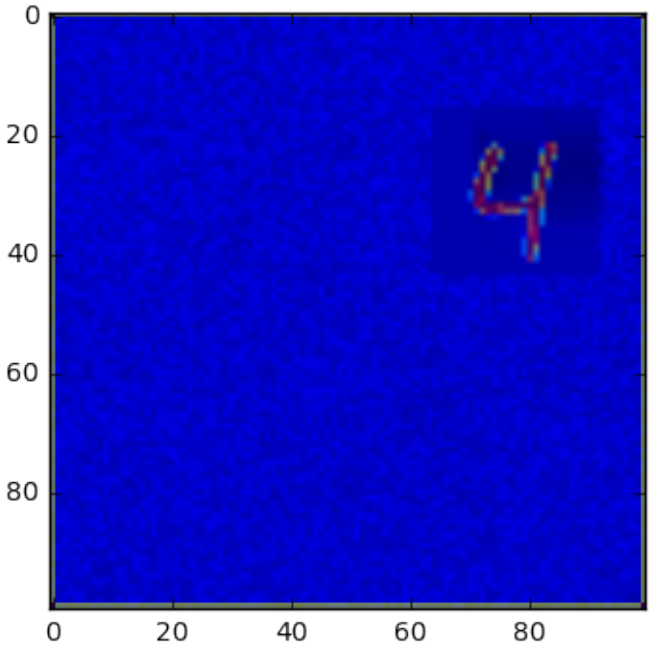
```
prediction is : 1 with 1.306
```



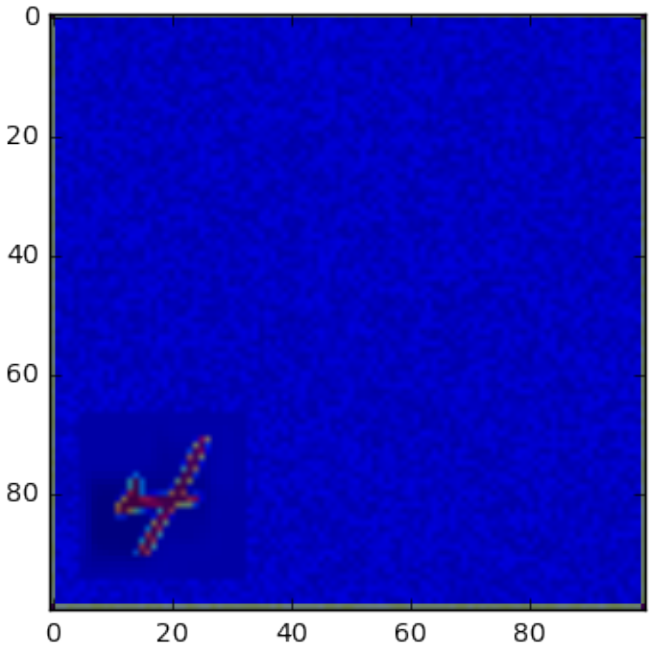
```
--------------------------------------------------------------------------------
prediction is : 1 with 1.268
```




```
--------------------------------------------------------------------------------
prediction is : 1 with 1.101
```



```
--------------------------------------------------------------------------------
prediction is : 1 with 1.138
```




```
In [5]: accuracies = []
        for _ in range(100):
            digit,lbl  = utils.get_batch('test', 1, .1).next()
            digit      = digit[0].reshape(28,28)
            img        = np.random.random((100,100))*.1
            height     = img.shape[0]
            width      = img.shape[1]
            box_size   = 28
            step_size  = 10
            n_x_boxes  = (width -box_size)/step_size +1
            n_y_boxes  = (height-box_size)/step_size +1
            imgs       = np.tile(img, (n_x_boxes*n_y_boxes,1,1))

            for xx in range(0, n_x_boxes ):
                for yy in range(0, n_y_boxes ):
                    idx = xx*n_x_boxes+yy
                    x   = xx*step_size
                    y   = yy*step_size
                    imgs[idx, x:x+box_size, y:y+box_size ] = digit
```

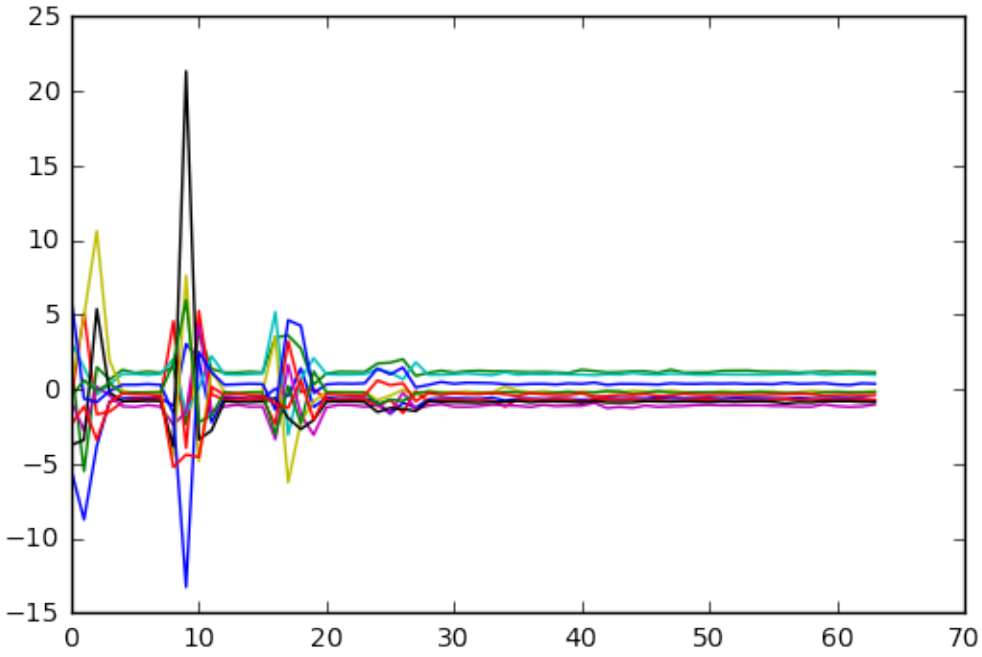
```
            imgs  = imgs.reshape((-1,100,100,1))
            preds = simple_model.sess.run(simple_model.tf_out, feed_dict={simple_mo
            pred  = np.argmax(preds[-1])

            accuracy = sum(preds.argmax(axis=1) == lbl) / float(len(preds))
            accuracies.append(accuracy)

        print sum(accuracies)/len(accuracies)
```

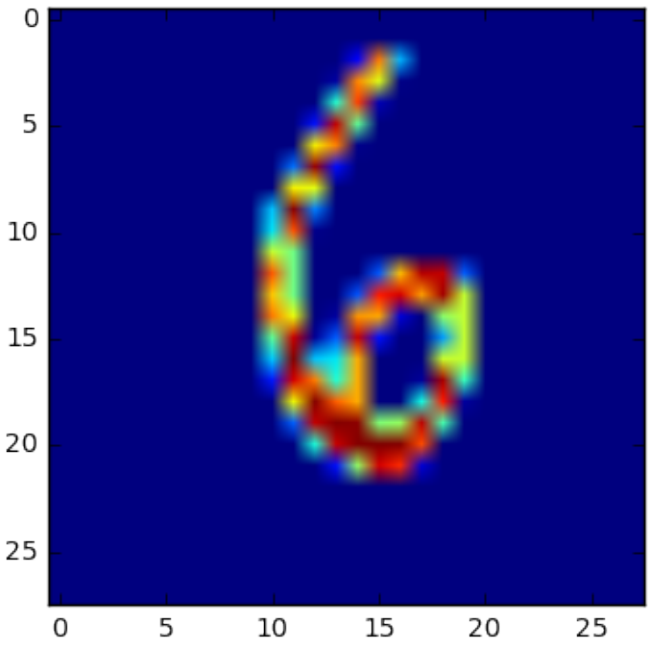
```
0.13453125
```

```
In [6]: digit,lbl  = utils.get_batch('test', 1, .1).next()
        digit      = digit[0].reshape(28,28)
        img        = np.random.random((100,100))*.1
        height     = img.shape[0]
        width      = img.shape[1]
        box_size   = 28
        step_size  = 10
        n_x_boxes  = (width -box_size)/step_size +1
        n_y_boxes  = (height-box_size)/step_size +1
        imgs       = np.tile(img, (n_x_boxes*n_y_boxes,1,1))

        for xx in range(0, n_x_boxes ):
            for yy in range(0, n_y_boxes ):
                idx = xx*n_x_boxes+yy
                x   = xx*step_size
                y   = yy*step_size
                imgs[idx, x:x+box_size, y:y+box_size ] = digit


        imgs  = imgs.reshape((-1,100,100,1))
        preds = simple_model.sess.run(simple_model.tf_out, feed_dict={simple_model.
        pred  = np.argmax(preds[-1])

        accuracy = sum(preds.argmax(axis=1) == lbl) / float(len(preds))

        print preds.argmax(axis=1)
        print accuracy*100
        plt.plot(preds)
        plt.show()

        plt.imshow(digit)
        plt.show()
```

```
[7 2 5 5 1 1 1 1 2 6 2 3 1 1 1 1 3 7 7 3 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
1.5625
```


25
20
15
10
5
0
−5
−10
−15
0
10
20
30
40
50
60
70


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