show_results

October 10, 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”) conv3_2 = new_conv_layer(conv3_1, [3, 3, 16, 2*10], “conv3_2”) gap = tf.reduce_mean(conv3_2, [1,2]) ccn = tf.reshape(gap,[-1,10,nb_CCN]) ccn = tf.reduce_max(ccn, 2)

1.2 train params

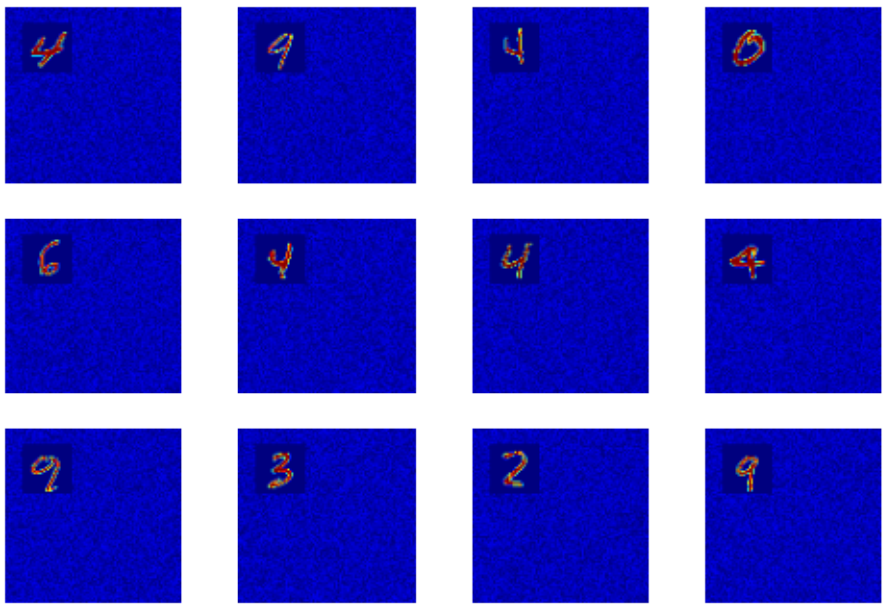
L2 on weights (5e-5)

```
In [8]: 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 [9]: """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 [10]: """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)
```

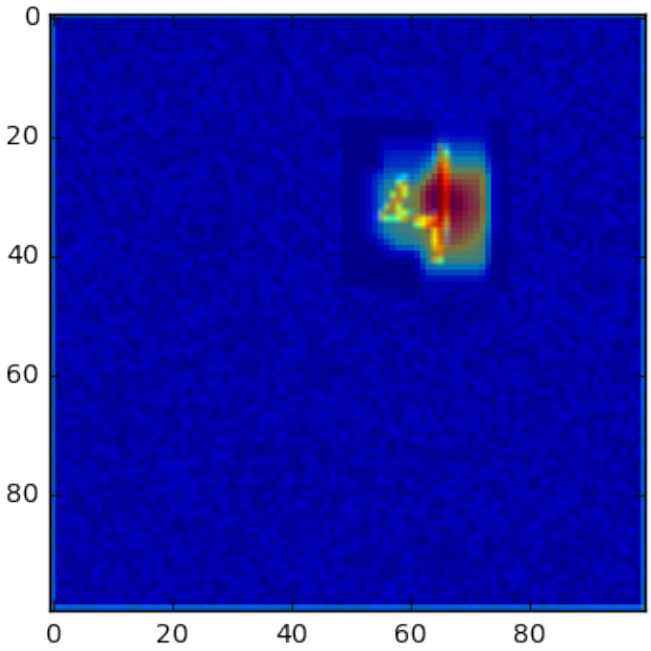
```
***** EPOCH 0 ******
1-Loss on testset is 2.736695
1-Accuracy now is 10.19
2-Loss on testset is 2.737214
2-Accuracy now is 10.20
lr now is 0.00475
max acc so far : 10.19

***** EPOCH 1 ******
1-Loss on testset is 0.864312
1-Accuracy now is 70.09
2-Loss on testset is 0.815974
2-Accuracy now is 71.05
```

```
lr now is 0.00451
max acc so far : 70.09

***** EPOCH 2 ******
1-Loss on testset is 0.218123
1-Accuracy now is 93.48
2-Loss on testset is 0.210013
2-Accuracy now is 93.77
lr now is 0.00429
max acc so far : 93.48

***** EPOCH 3 ******
1-Loss on testset is 0.315590
1-Accuracy now is 90.11
2-Loss on testset is 0.283632
2-Accuracy now is 91.43
lr now is 0.00407
max acc so far : 93.48

***** EPOCH 4 ******
1-Loss on testset is 0.171458
1-Accuracy now is 94.89
2-Loss on testset is 0.158008
2-Accuracy now is 95.31
lr now is 0.00387
max acc so far : 94.89

***** EPOCH 5 ******
1-Loss on testset is 0.108049
1-Accuracy now is 96.99
2-Loss on testset is 0.099668
2-Accuracy now is 97.19
lr now is 0.00368
max acc so far : 96.99

***** EPOCH 6 ******
1-Loss on testset is 0.173525
1-Accuracy now is 94.72
2-Loss on testset is 0.155894
2-Accuracy now is 95.61
lr now is 0.00349
max acc so far : 96.99

***** EPOCH 7 ******
1-Loss on testset is 0.287814
1-Accuracy now is 91.65
2-Loss on testset is 0.245194
2-Accuracy now is 93.13
```

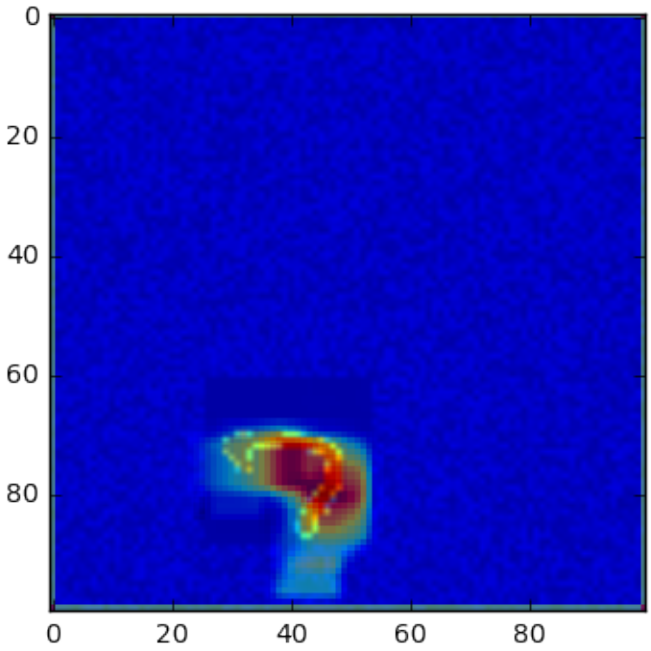
```
lr now is 0.00332
max acc so far : 96.99

***** EPOCH 8 ******
1-Loss on testset is 0.303429
1-Accuracy now is 90.87
2-Loss on testset is 0.078729
2-Accuracy now is 98.17
lr now is 0.00315
max acc so far : 96.99

***** EPOCH 9 ******
1-Loss on testset is 0.094213
1-Accuracy now is 97.44
2-Loss on testset is 0.056484
2-Accuracy now is 98.76
lr now is 0.00299
max acc so far : 97.44
```

```
In [11]: 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])
```

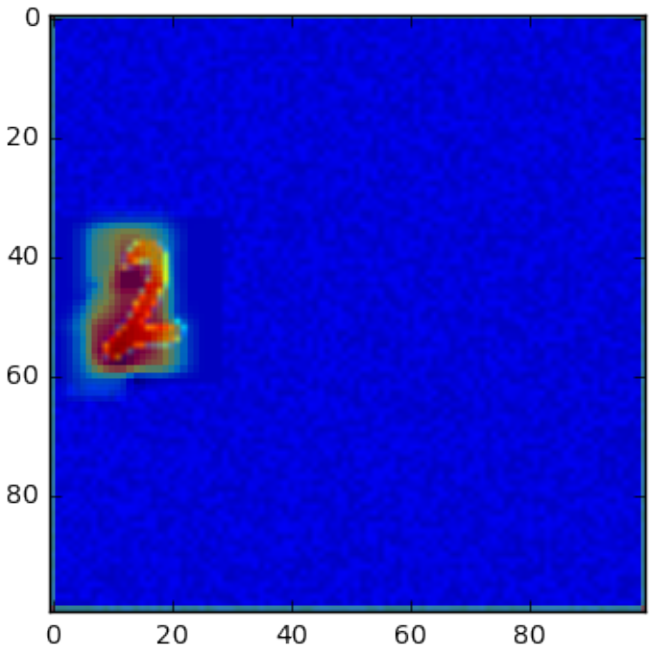
```
--------------------------------------------------------------------------------
prediction is : 4 with 17.907
```



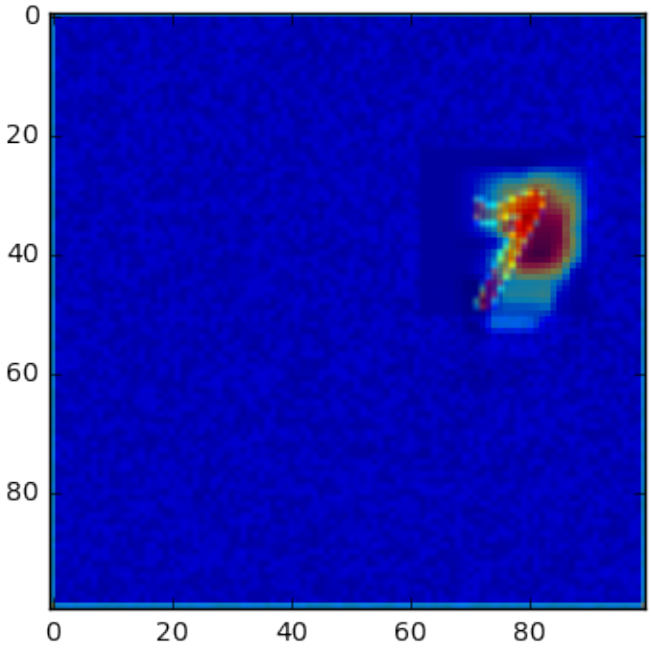
```
prediction is : 7 with 24.534
```



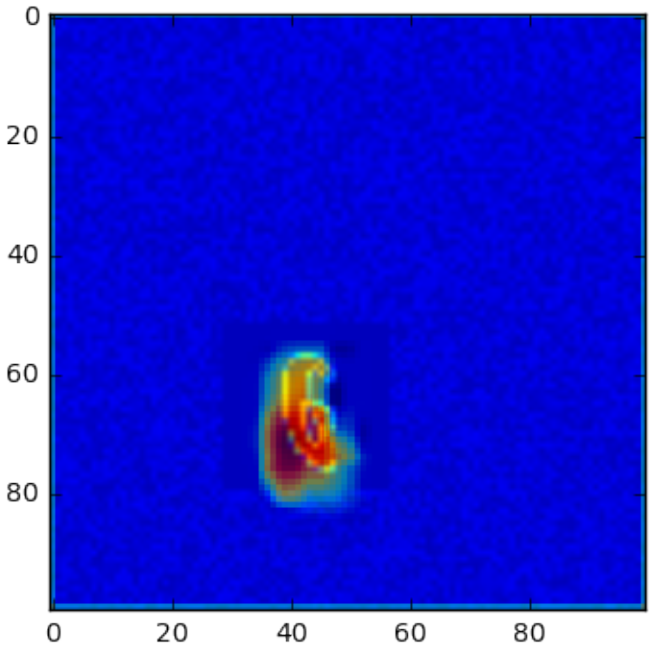
```
--------------------------------------------------------------------------------
prediction is : 2 with 19.189
```



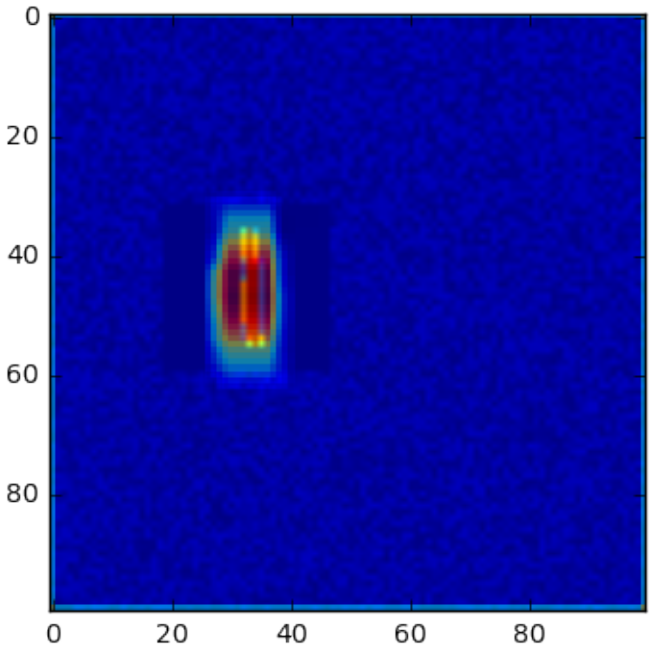

```
--------------------------------------------------------------------------------
prediction is : 7 with 17.225
```



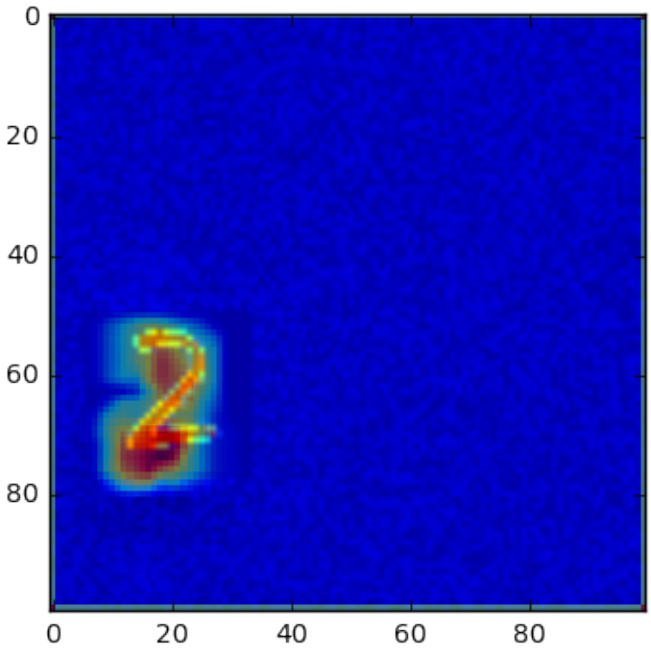
```
--------------------------------------------------------------------------------
prediction is : 6 with 17.594
```



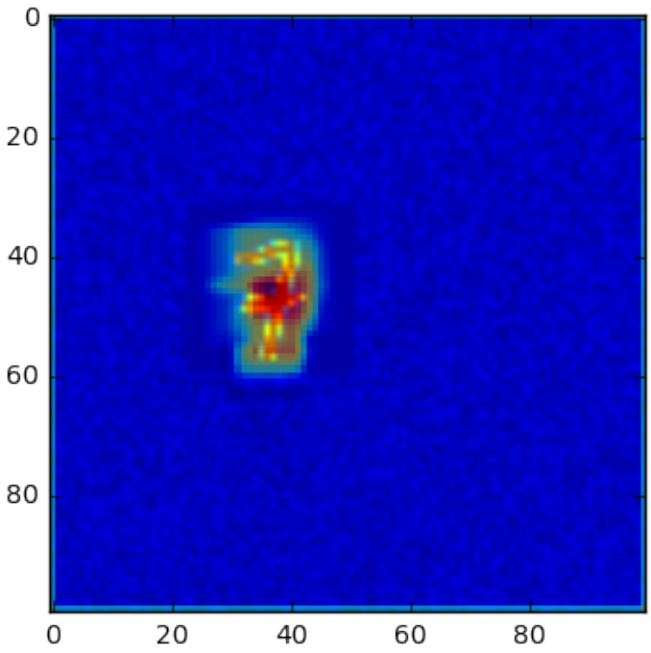

```
--------------------------------------------------------------------------------
prediction is : 1 with 18.406
```



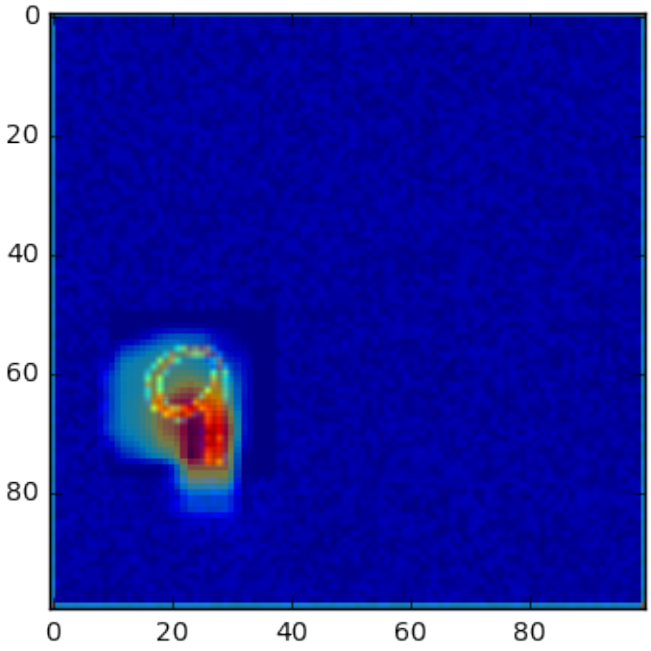
```
prediction is : 2 with 18.862
```



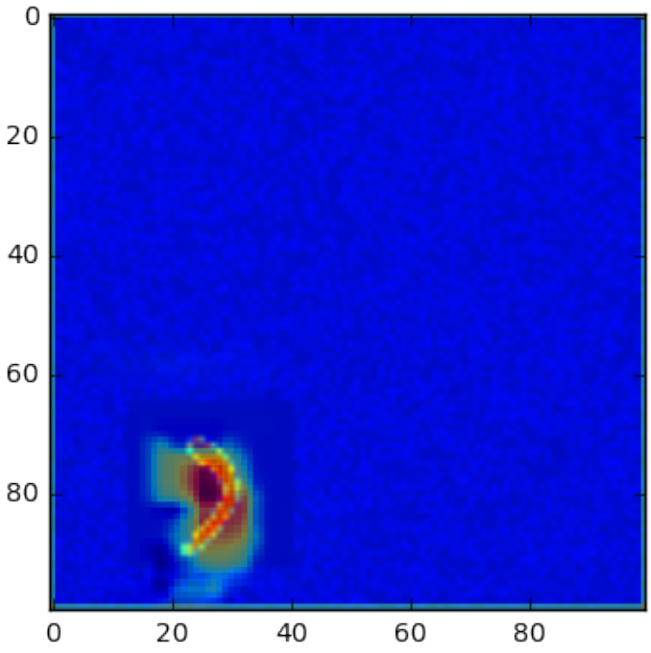
```
--------------------------------------------------------------------------------
prediction is : 7 with 19.930
```



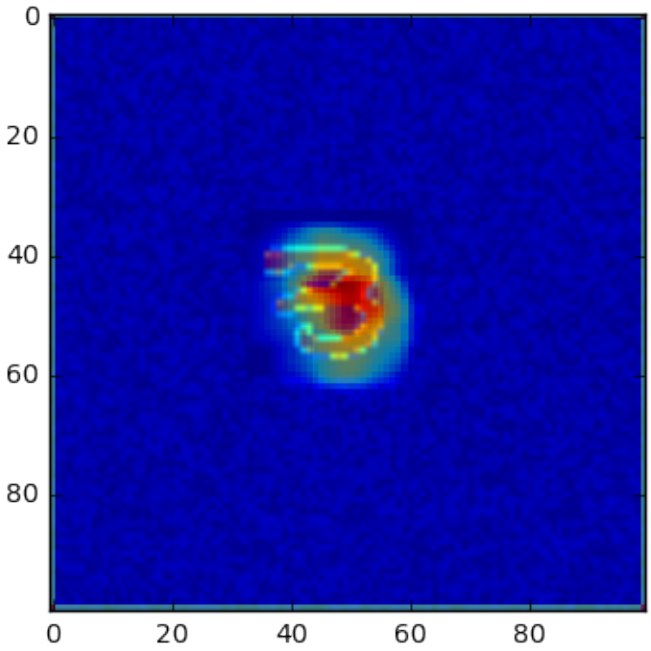

```
--------------------------------------------------------------------------------
prediction is : 9 with 23.781
```



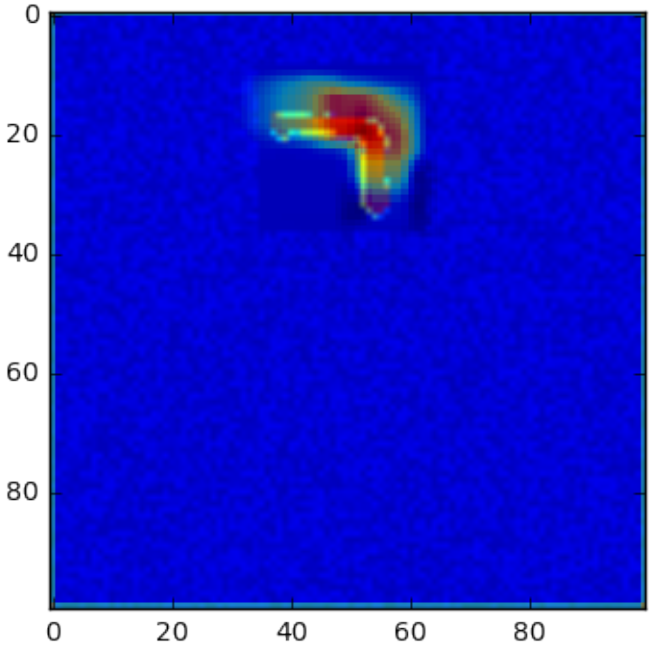
```
--------------------------------------------------------------------------------
prediction is : 7 with 7.969
```



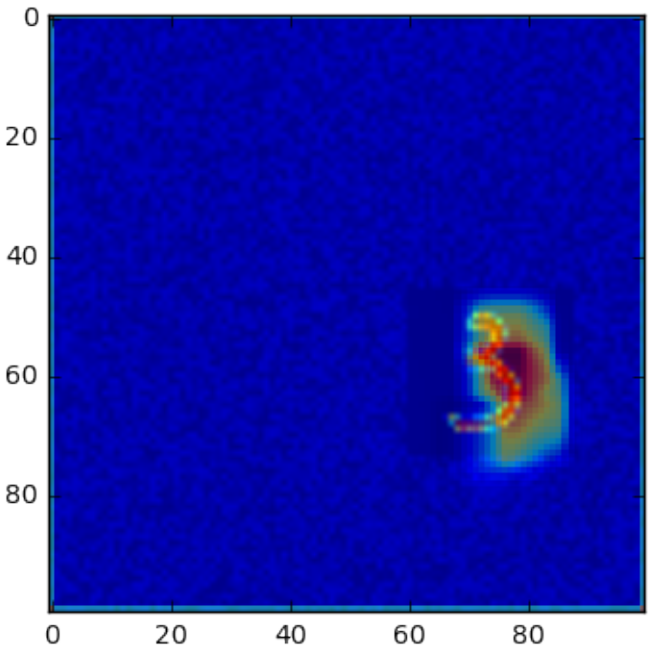

```
--------------------------------------------------------------------------------
prediction is : 3 with 33.462
```



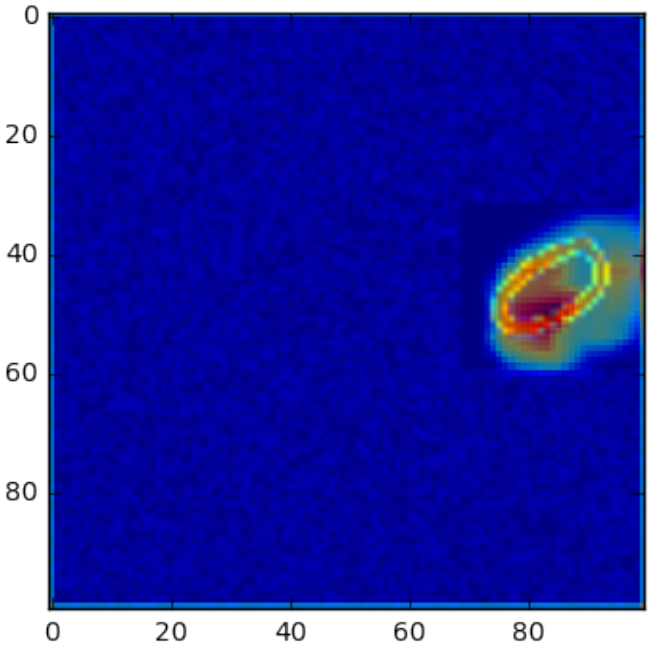
```
prediction is : 7 with 20.122
```



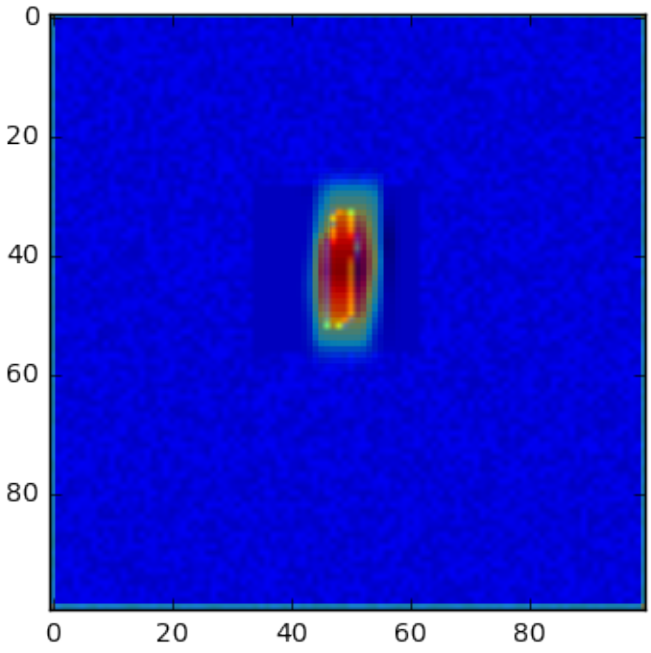
```
--------------------------------------------------------------------------------
prediction is : 3 with 19.034
```



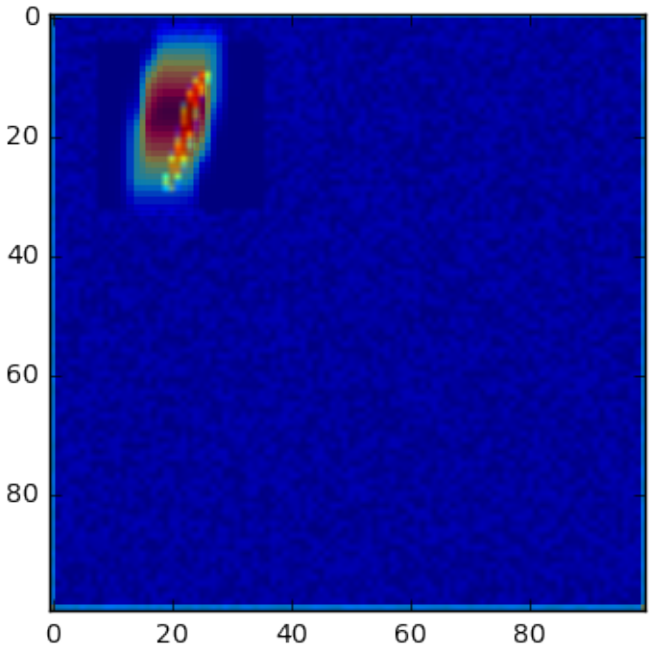

```
--------------------------------------------------------------------------------
prediction is : 0 with 20.104
```



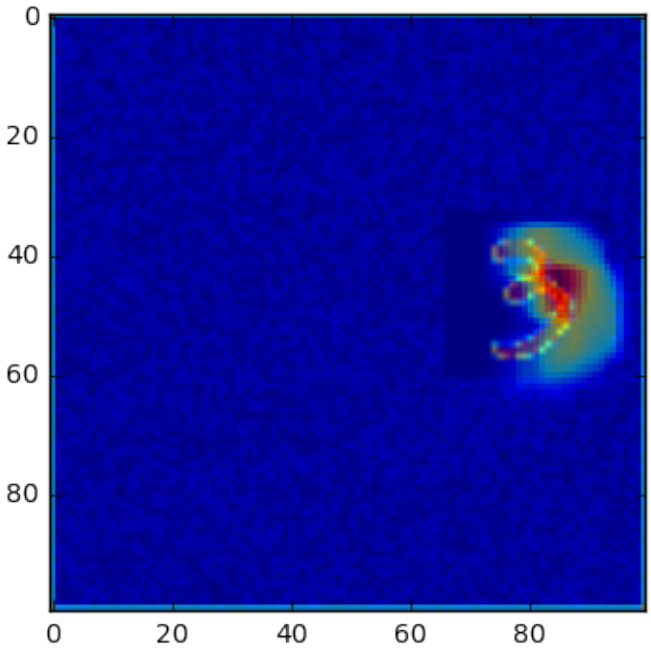
```
--------------------------------------------------------------------------------
prediction is : 1 with 19.097
```



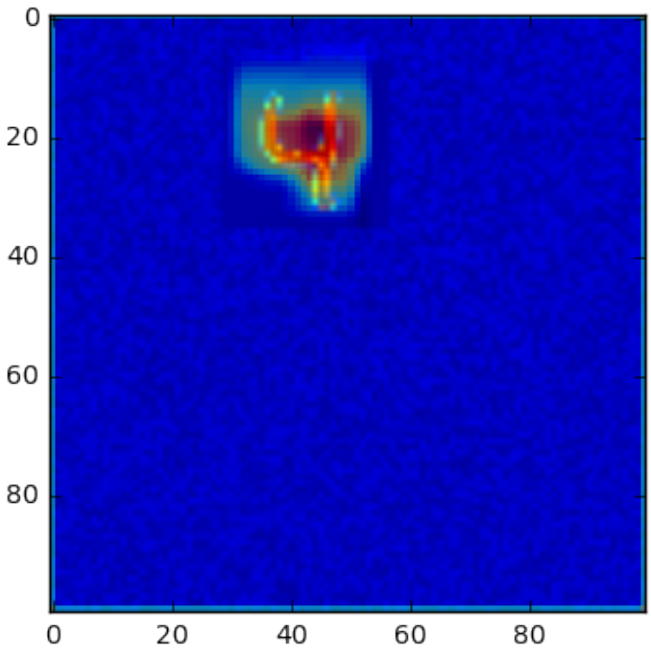

```
--------------------------------------------------------------------------------
prediction is : 1 with 18.645
```



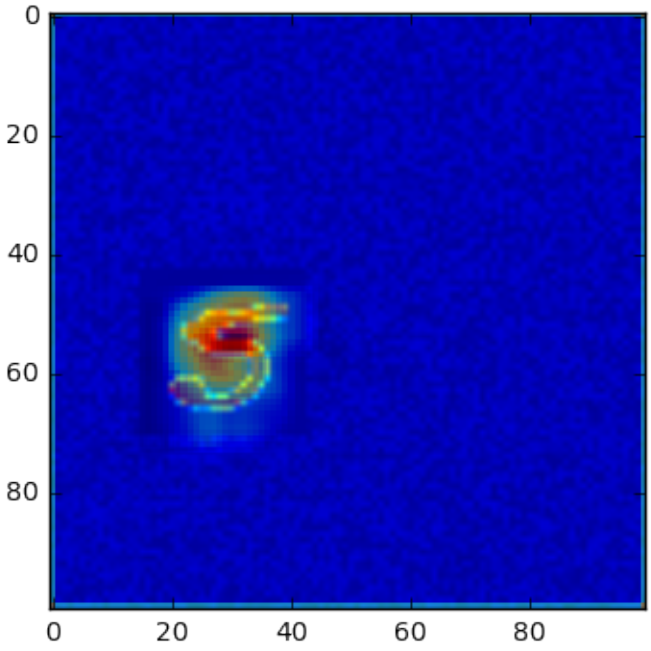
```
prediction is : 3 with 24.403
```



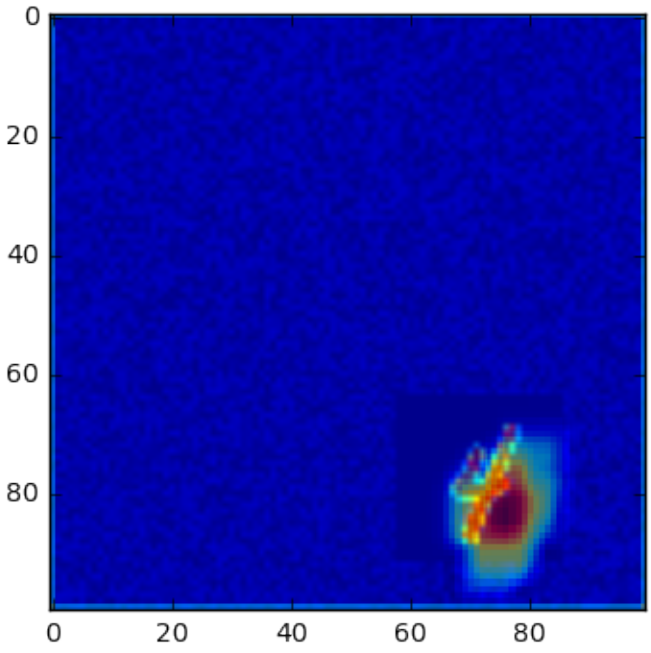
```
--------------------------------------------------------------------------------
prediction is : 4 with 26.639
```



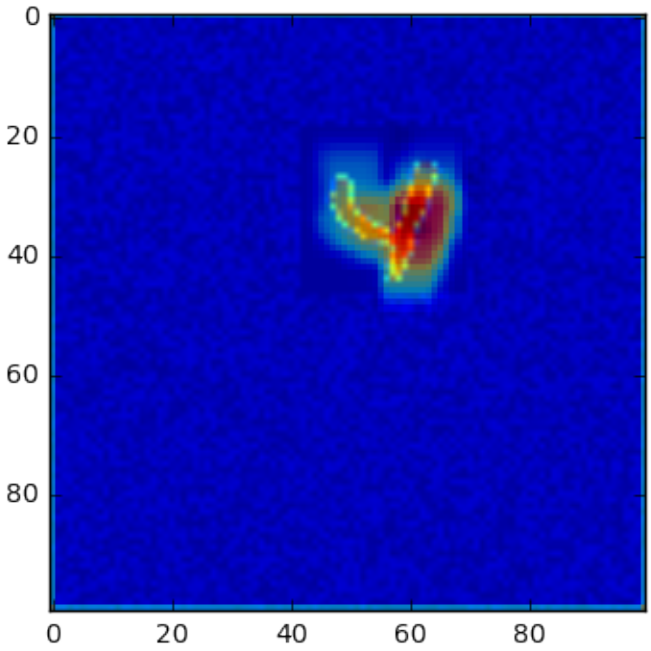

```
--------------------------------------------------------------------------------
prediction is : 5 with 22.644
```



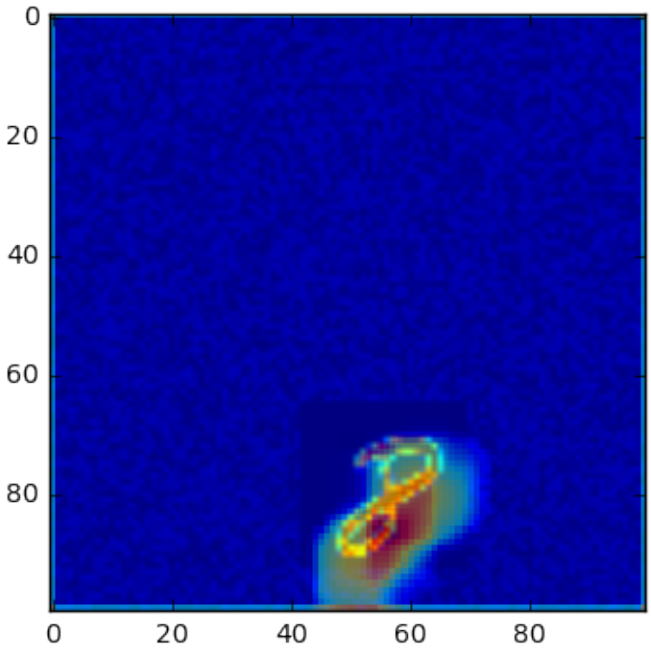
```
--------------------------------------------------------------------------------
prediction is : 4 with 22.264
```



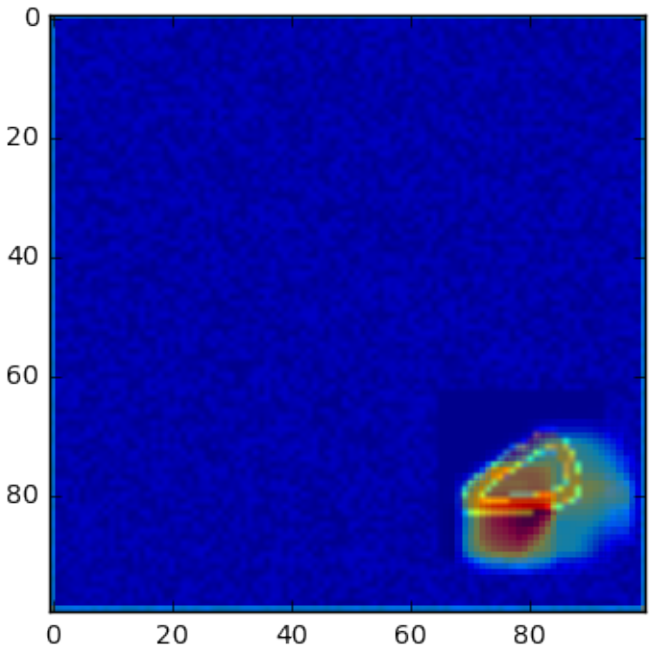

```
--------------------------------------------------------------------------------
prediction is : 4 with 24.623
```



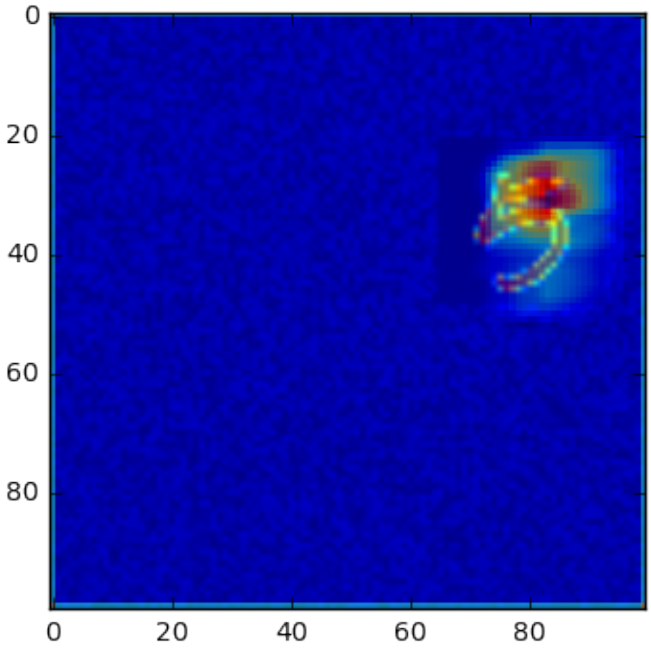
```
prediction is : 8 with 22.785
```



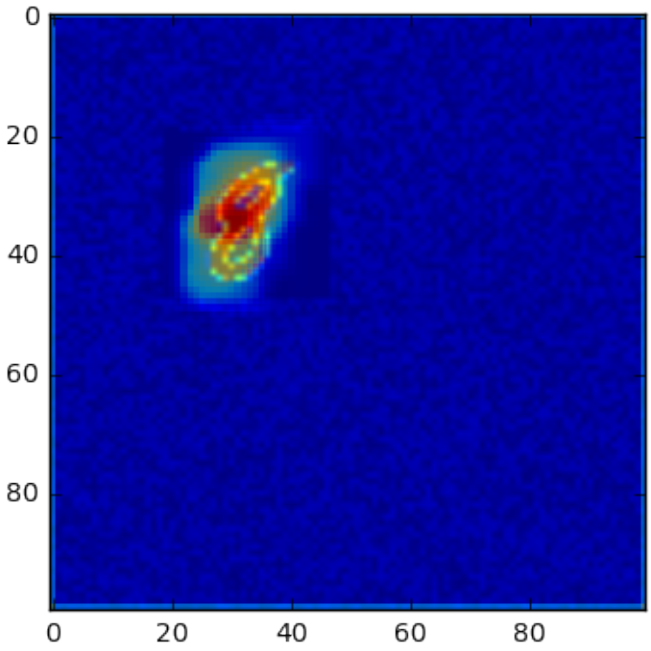
```
--------------------------------------------------------------------------------
prediction is : 0 with 19.203
```




```
--------------------------------------------------------------------------------
prediction is : 5 with 20.794
```



```
--------------------------------------------------------------------------------
prediction is : 8 with 28.389
```




```
In [12]: 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
```

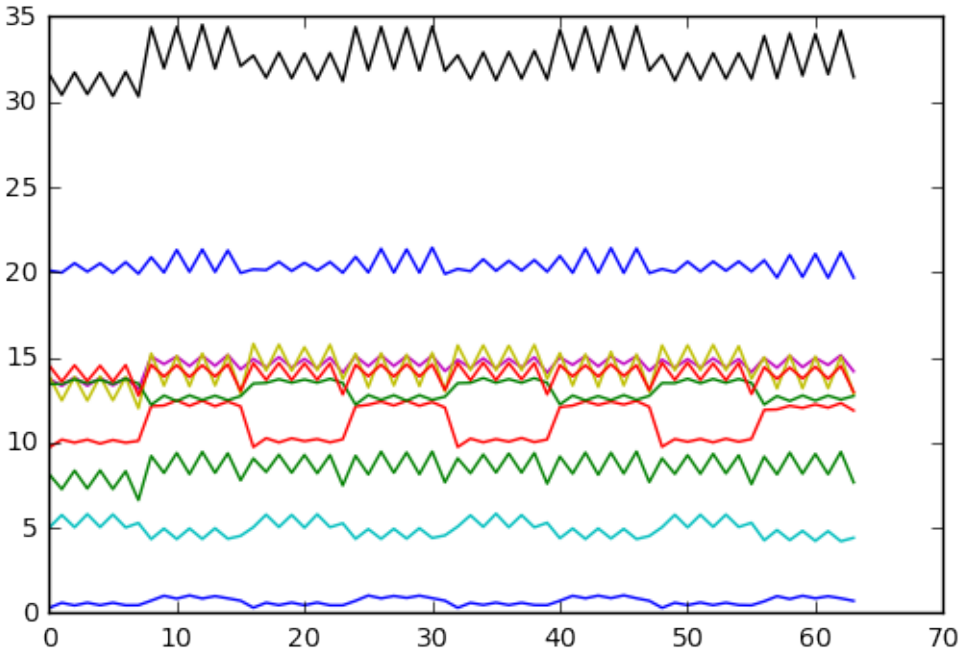
```
    imgs  = imgs.reshape((-1,100,100,1))
    preds = simple_model.sess.run(simple_model.tf_out, feed_dict={simple_m
    pred  = np.argmax(preds[-1])

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

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

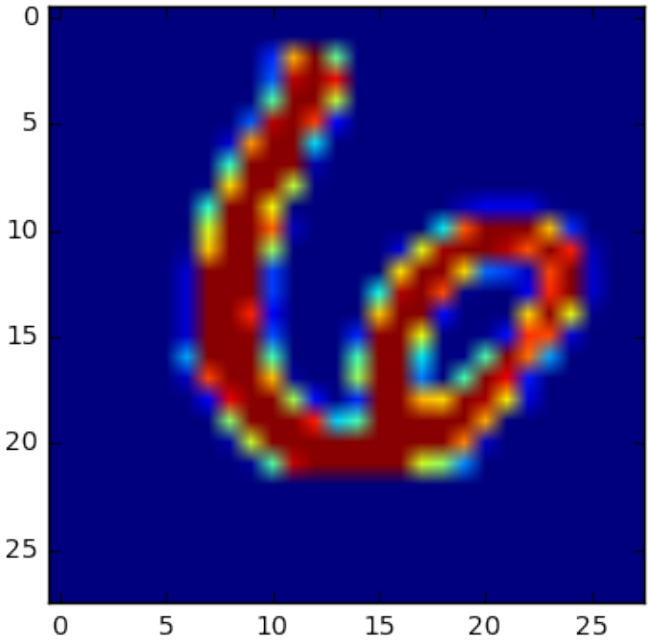
```
0.99921875
```

```
In [13]: 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()
```

```
[6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6]
```

```
100.0
```





```
In [14]:
```

```
Out[14]: <tf.Tensor 'Max:0' shape=(?, 10) dtype=float32>
```