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”)
conv3_2 = self.new_conv_layer(conv3_1, [3, 3, 16, self.n_labels], “conv3_2”)
gap = tf.reduce_mean(conv3_2, [1,2])

1.2 train params

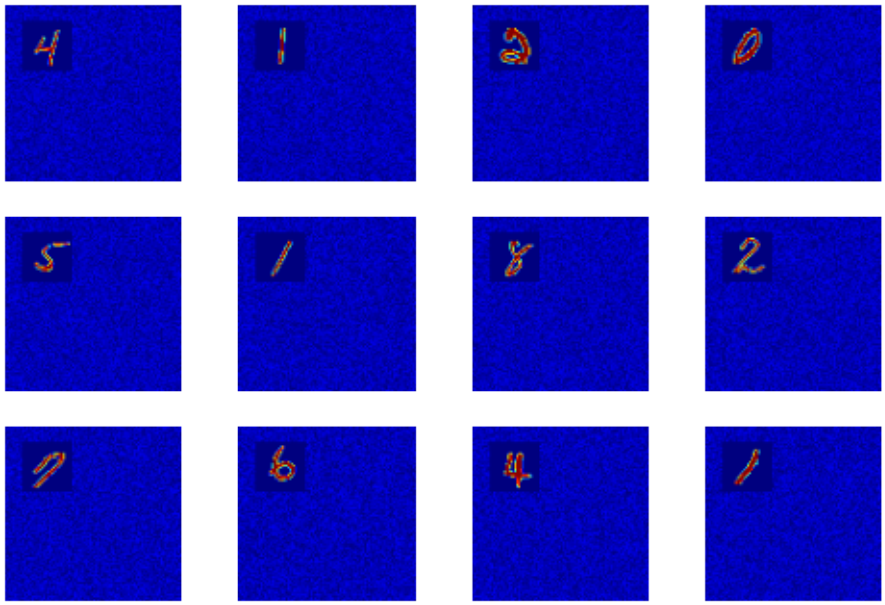
L2 on weights (5e-5)

```
In [35]: 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 [36]: """Plot training samples"""
         batch = utils.get_batch('train', im_size=back_size, noise=noise, crop_pos=
```

```
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 [37]:
```
"""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)
```

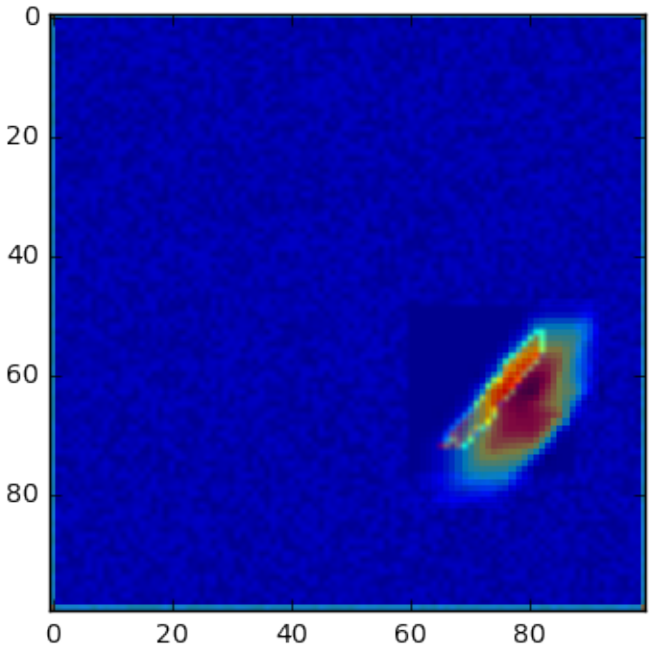
```
***** EPOCH 0 ******
1-Loss on testset is 2.113894
1-Accuracy now is 17.56
2-Loss on testset is 2.105299
2-Accuracy now is 17.85
lr now is 0.00475
max acc so far : 17.56

***** EPOCH 1 ******
1-Loss on testset is 0.956879
1-Accuracy now is 68.82
```

```
2-Loss on testset is 0.928202
2-Accuracy now is 70.21
lr now is 0.00451
max acc so far : 68.82

***** EPOCH 2 ******
1-Loss on testset is 0.229401
1-Accuracy now is 93.46
2-Loss on testset is 0.213393
2-Accuracy now is 93.83
lr now is 0.00429
max acc so far : 93.46

***** EPOCH 3 ******
1-Loss on testset is 0.200623
1-Accuracy now is 94.11
2-Loss on testset is 0.180762
2-Accuracy now is 94.79
lr now is 0.00407
max acc so far : 94.11

***** EPOCH 4 ******
1-Loss on testset is 0.281878
1-Accuracy now is 91.45
2-Loss on testset is 0.240784
2-Accuracy now is 92.59
lr now is 0.00387
max acc so far : 94.11

***** EPOCH 5 ******
1-Loss on testset is 0.135351
1-Accuracy now is 96.14
2-Loss on testset is 0.120120
2-Accuracy now is 96.73
lr now is 0.00368
max acc so far : 96.14

***** EPOCH 6 ******
1-Loss on testset is 0.185089
1-Accuracy now is 94.45
2-Loss on testset is 0.177714
2-Accuracy now is 94.61
lr now is 0.00349
max acc so far : 96.14

***** EPOCH 7 ******
1-Loss on testset is 0.100241
1-Accuracy now is 97.30
```

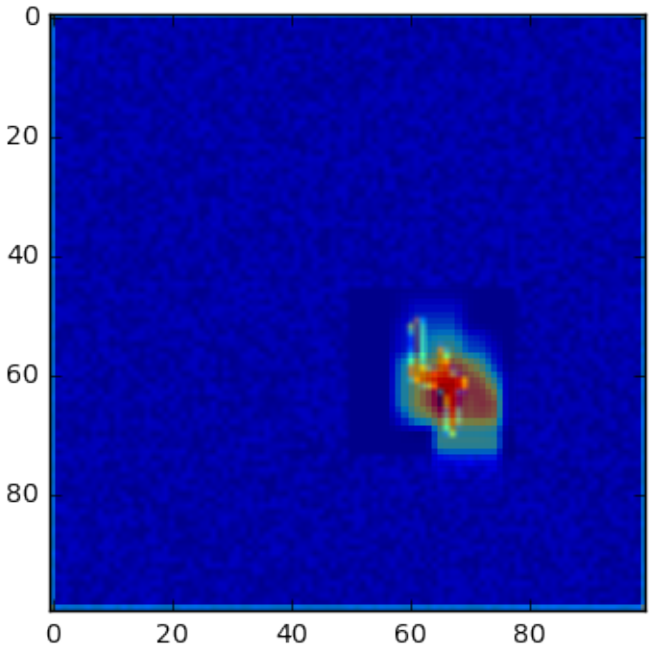
```
2-Loss on testset is 0.090424
2-Accuracy now is 97.70
lr now is 0.00332
max acc so far : 97.3

***** EPOCH 8 ******
1-Loss on testset is 0.149847
1-Accuracy now is 96.03
2-Loss on testset is 0.138017
2-Accuracy now is 96.61
lr now is 0.00315
max acc so far : 97.3

***** EPOCH 9 ******
1-Loss on testset is 0.102845
1-Accuracy now is 97.24
2-Loss on testset is 0.090297
2-Accuracy now is 97.81
lr now is 0.00299
max acc so far : 97.3
```

```
In [38]: 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])
```

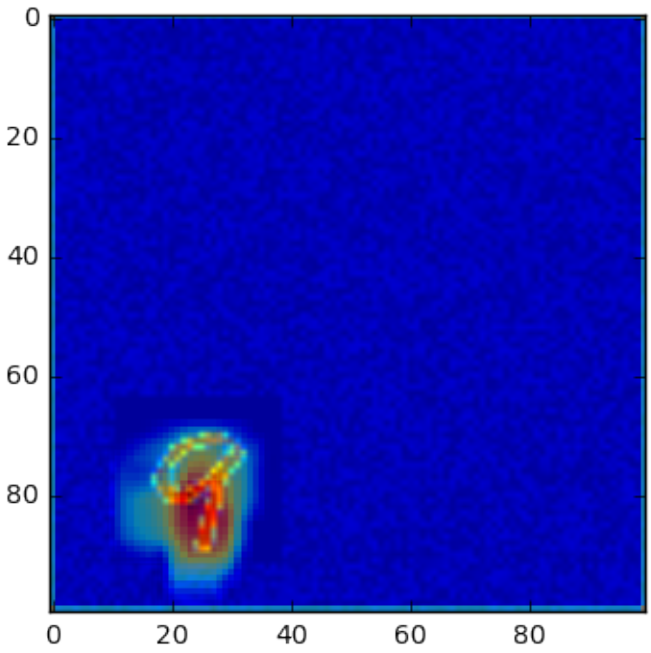
```
--------------------------------------------------------------------------------
prediction is : 1 with 19.373
```



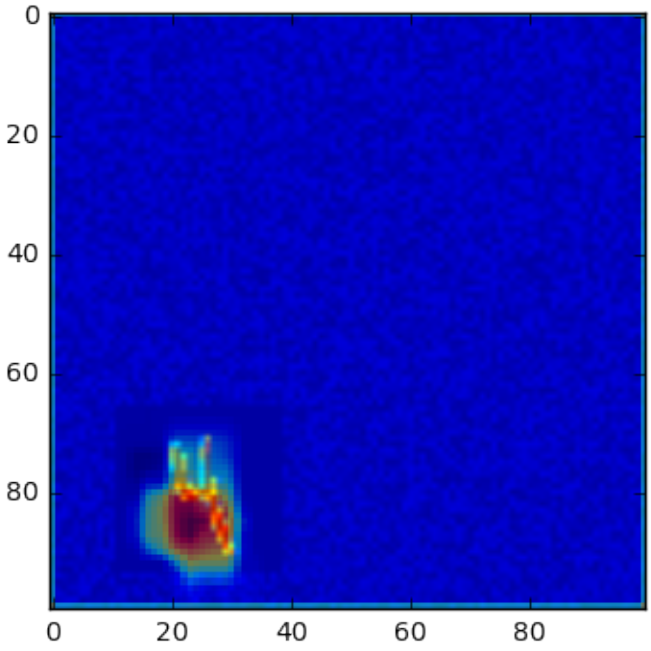
```
prediction is : 4 with 15.434
```



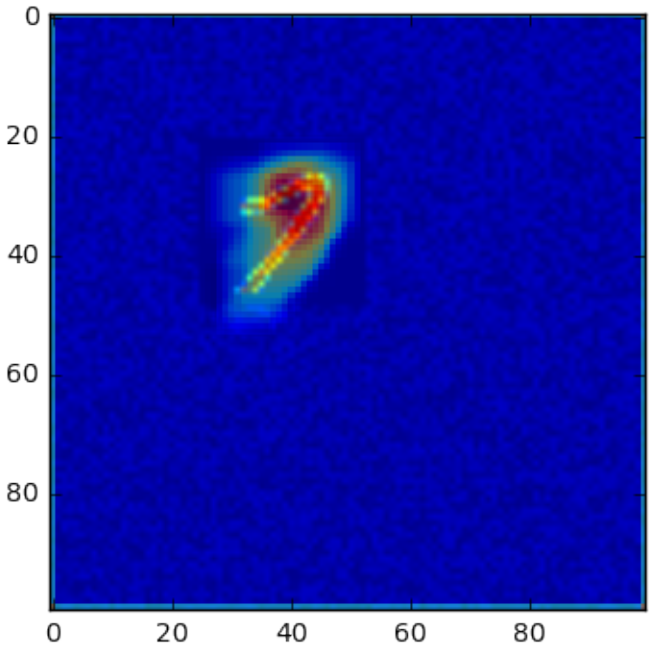
```
--------------------------------------------------------------------------------
prediction is : 9 with 23.156
```



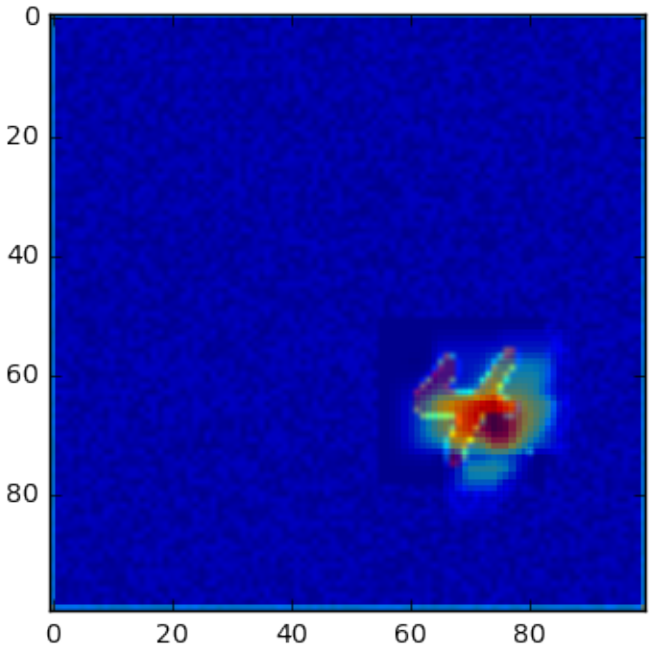

```
--------------------------------------------------------------------------------
prediction is : 4 with 12.029
```



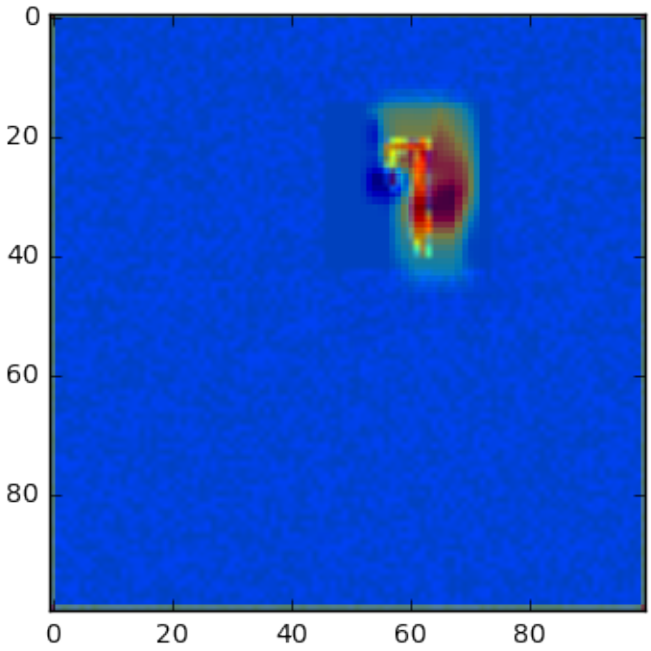
```
--------------------------------------------------------------------------------
prediction is : 7 with 21.820
```



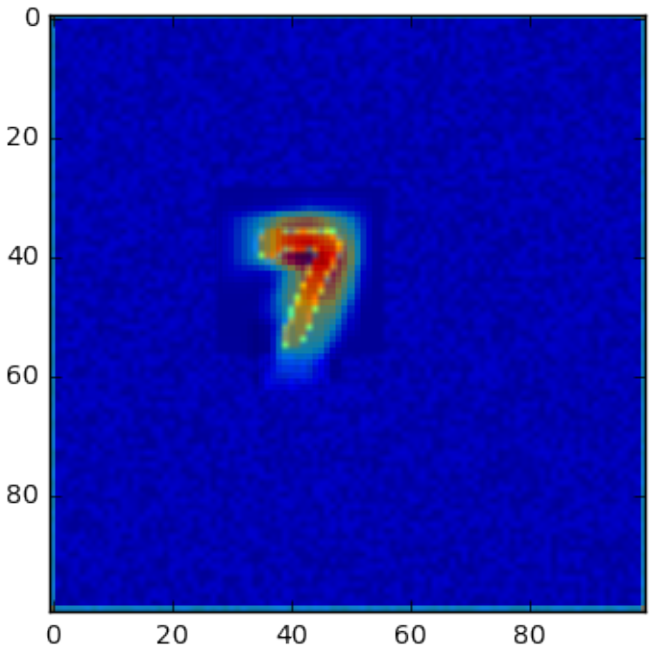

```
--------------------------------------------------------------------------------
prediction is : 4 with 32.359
```



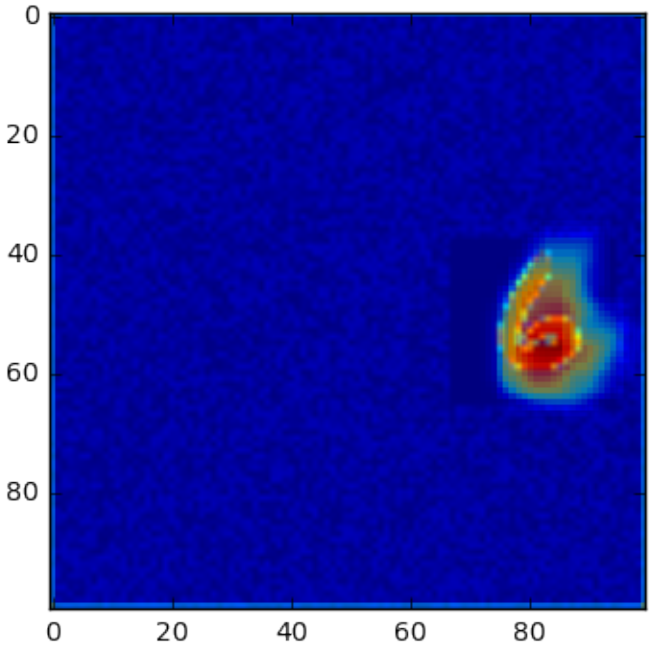
```
prediction is : 7 with 12.587
```



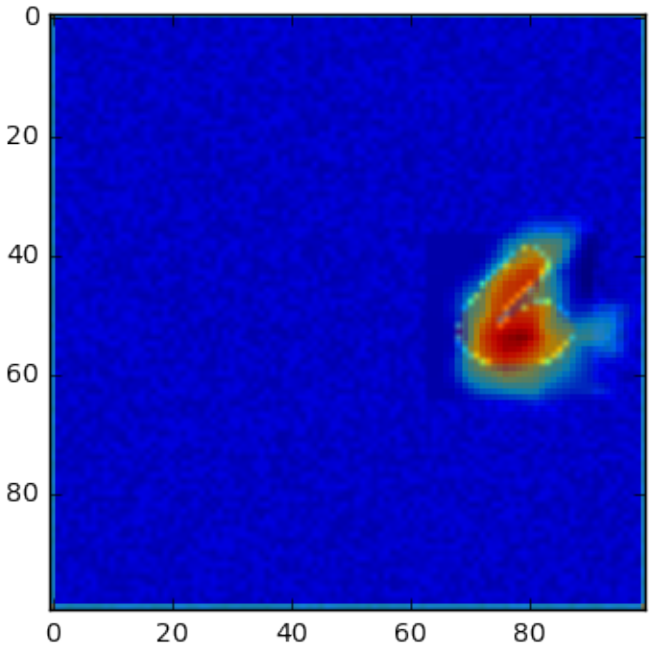
```
--------------------------------------------------------------------------------
prediction is : 7 with 27.253
```



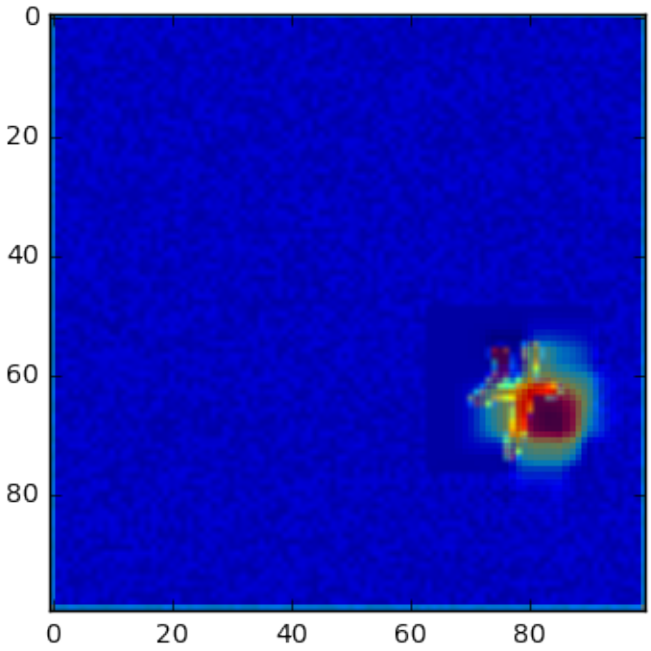

```
--------------------------------------------------------------------------------
prediction is : 6 with 21.509
```



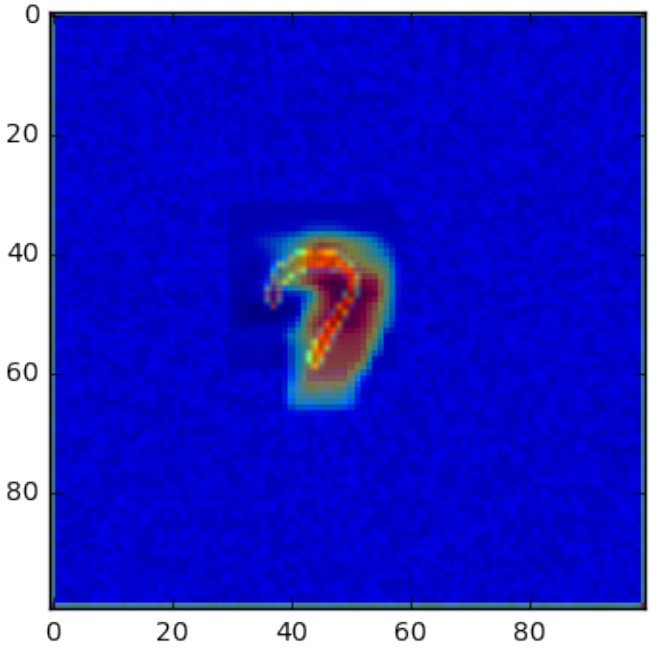
```
--------------------------------------------------------------------------------
prediction is : 6 with 16.680
```



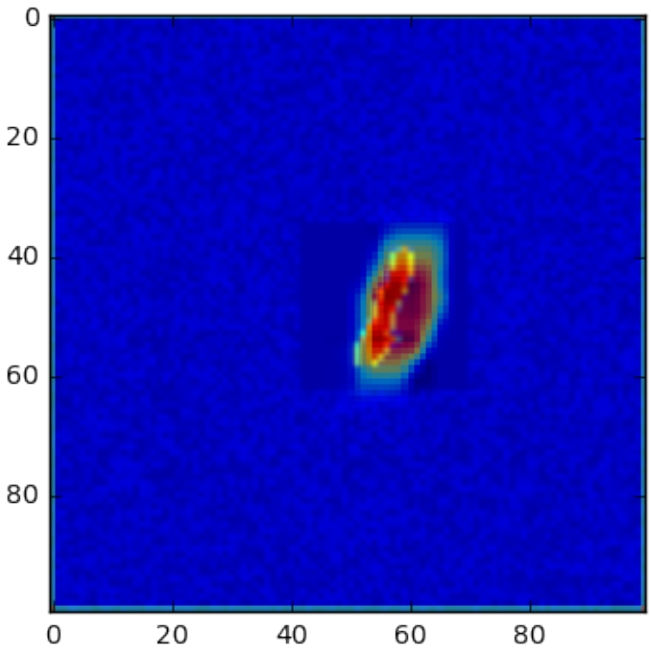

```
--------------------------------------------------------------------------------
prediction is : 4 with 24.070
```



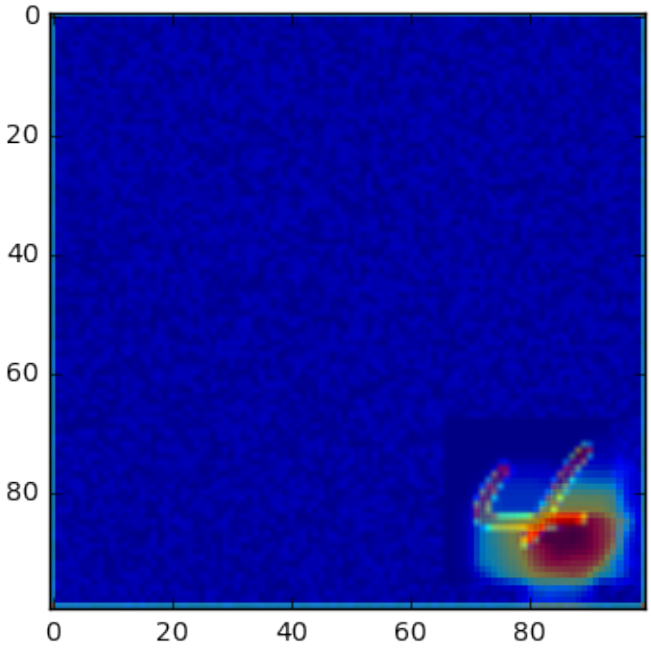
```
prediction is : 7 with 20.280
```



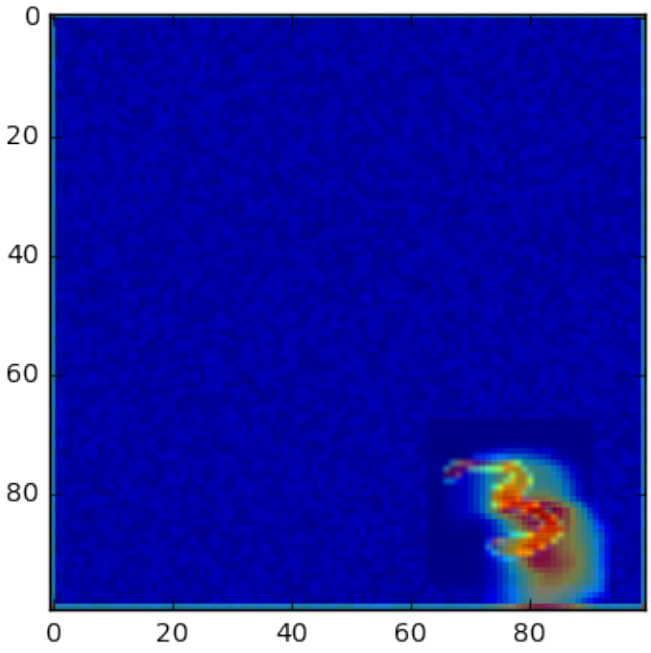
```
--------------------------------------------------------------------------------
prediction is : 1 with 19.088
```



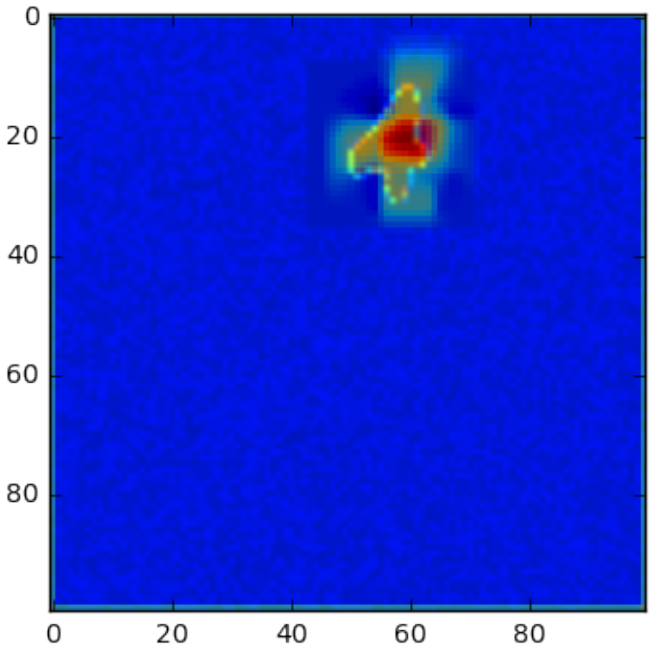

```
--------------------------------------------------------------------------------
prediction is : 4 with 31.507
```



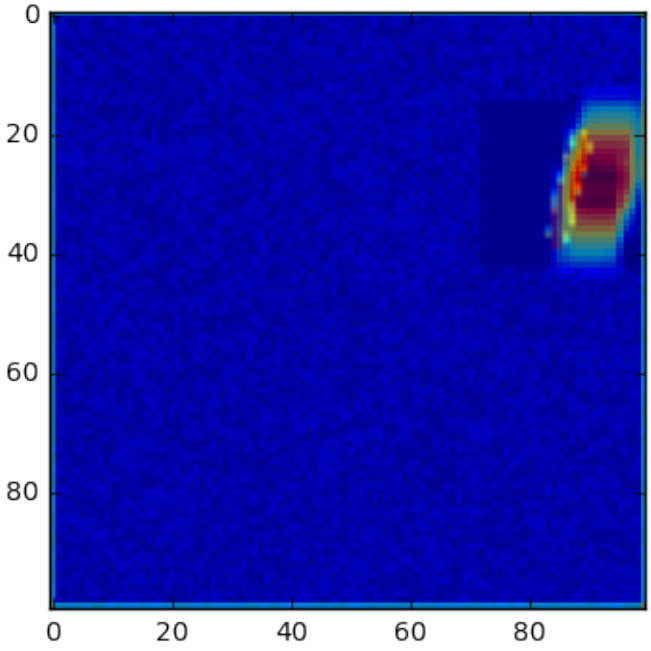
```
--------------------------------------------------------------------------------
prediction is : 3 with 22.426
```



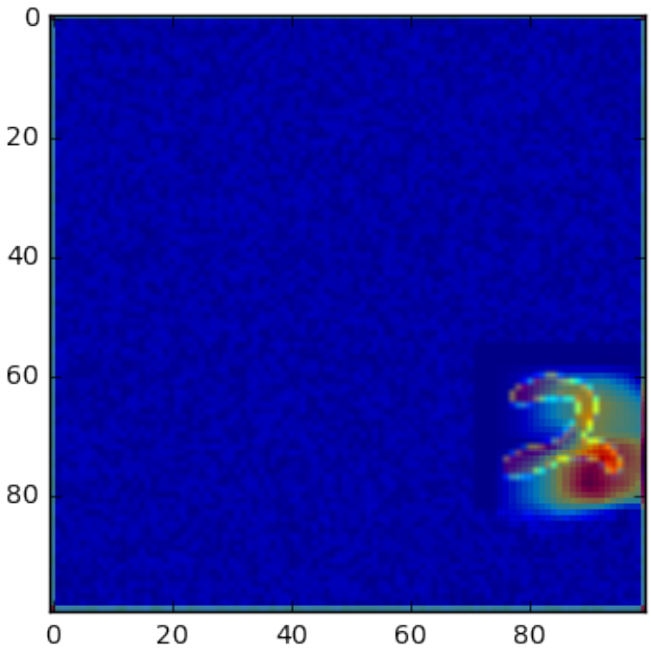

```
--------------------------------------------------------------------------------
prediction is : 4 with 19.360
```



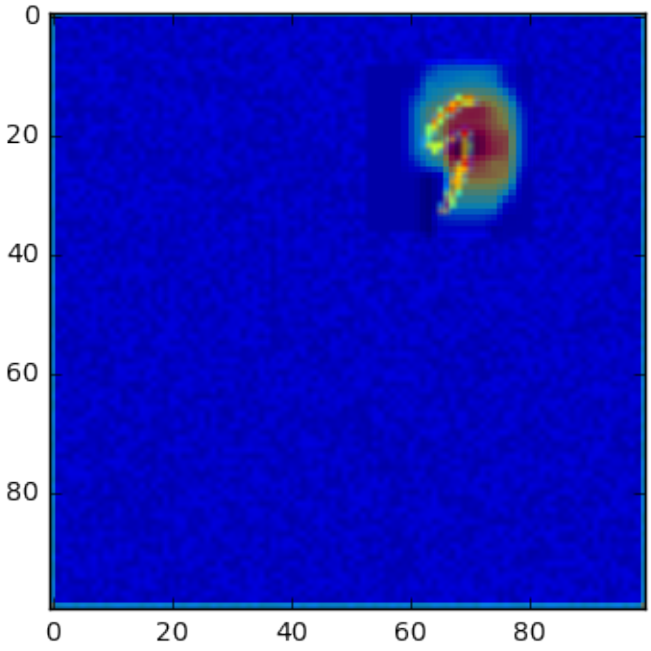
```
prediction is : 1 with 20.271
```



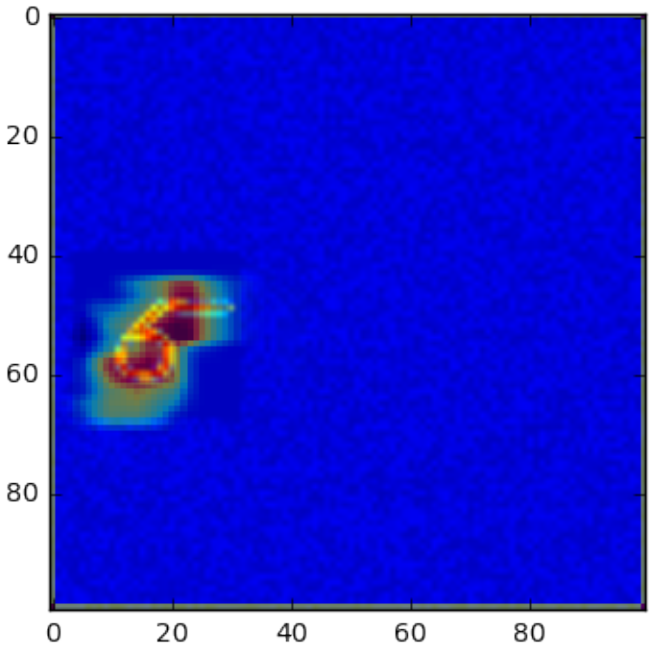
```
--------------------------------------------------------------------------------
prediction is : 2 with 24.522
```



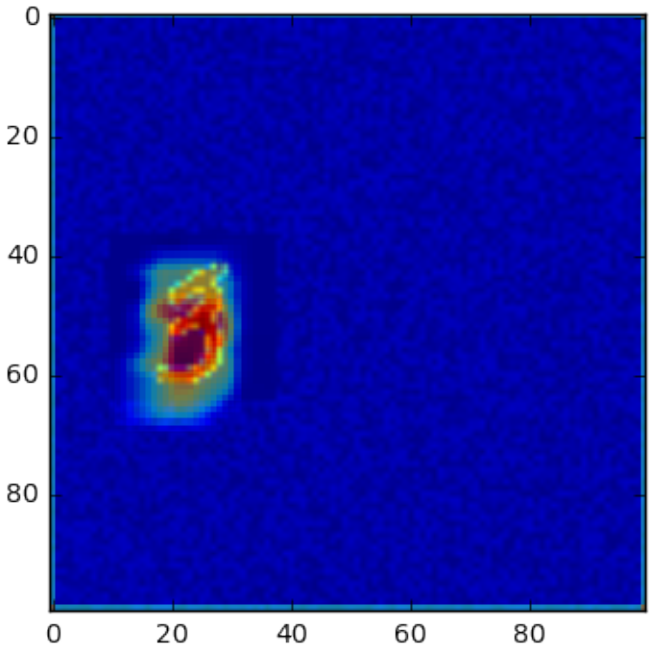

```
--------------------------------------------------------------------------------
prediction is : 9 with 8.612
```



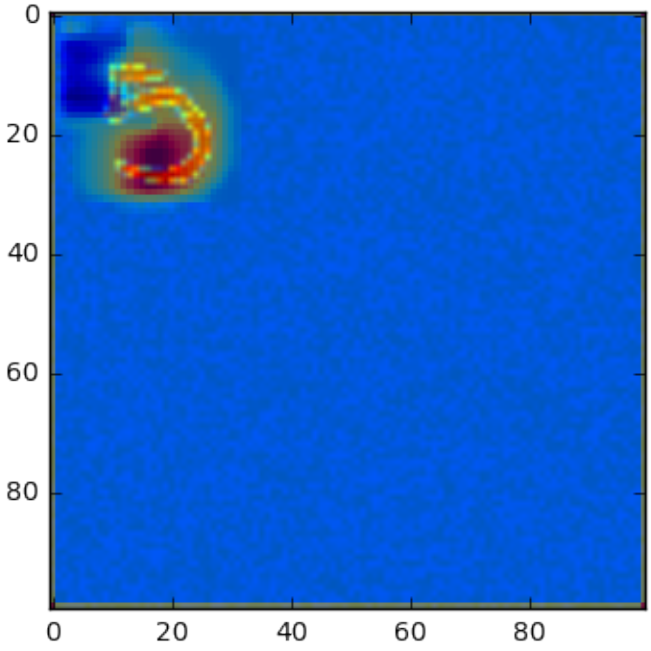
```
--------------------------------------------------------------------------------
prediction is : 5 with 15.216
```



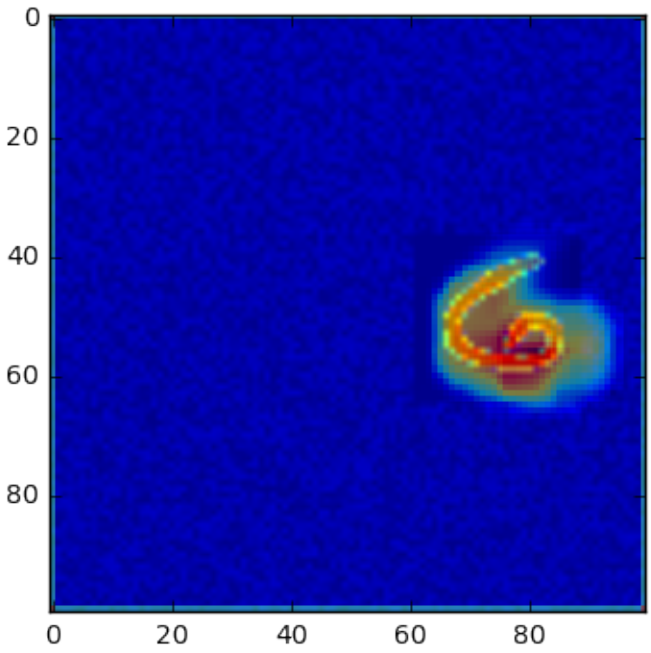

```
--------------------------------------------------------------------------------
prediction is : 3 with 26.528
```



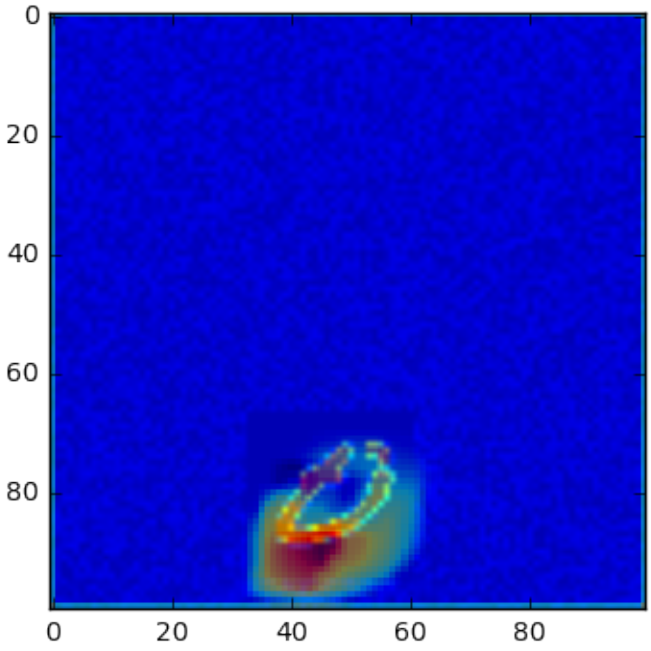
```
prediction is : 3 with 16.921
```



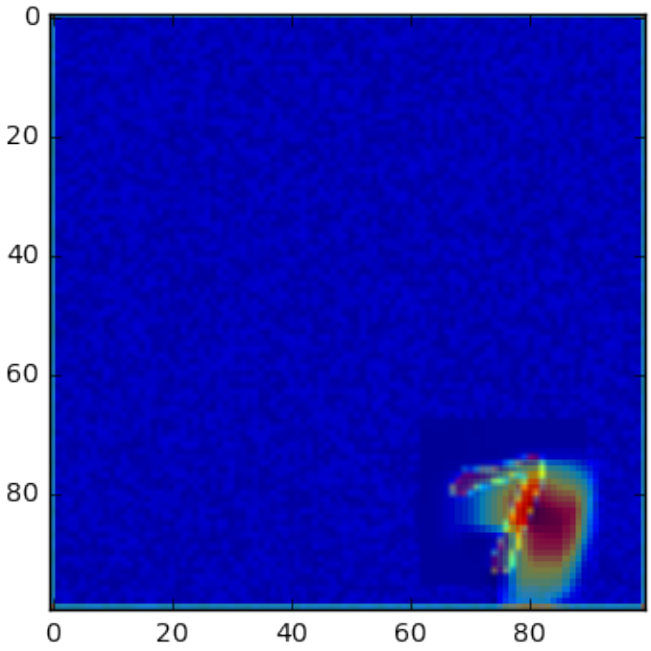
```
--------------------------------------------------------------------------------
prediction is : 6 with 19.785
```




```
--------------------------------------------------------------------------------
prediction is : 0 with 15.461
```



```
--------------------------------------------------------------------------------
prediction is : 7 with 26.654
```




```
In [39]: 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)
```

```
0.98265625
```

```
In [40]: 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()
```

```
[8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8]
```

```
100.0
```