show_results

October 11, 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’)

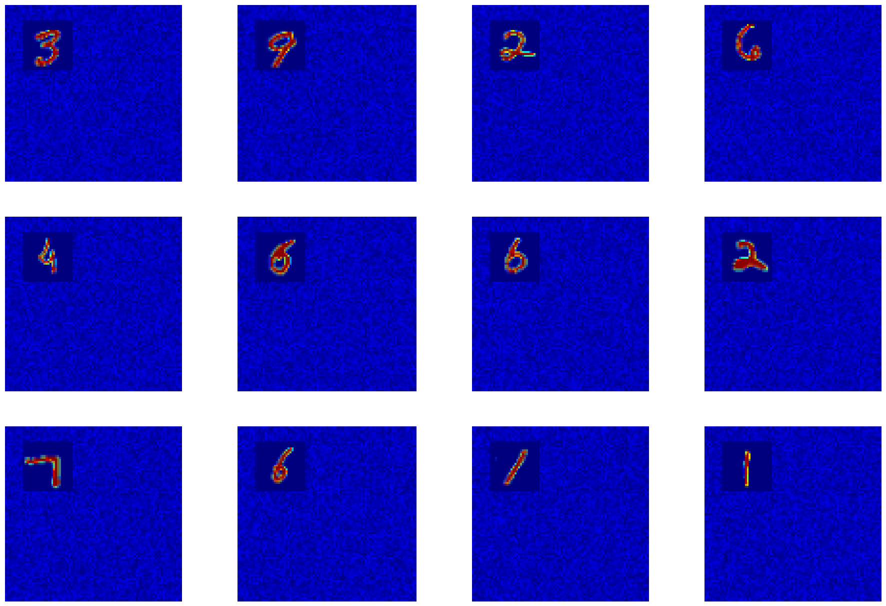
conv3_1 = new_conv_layer(pool2, [3, 3, 16, 16], “conv3_1”) conv3_2 = new_conv_layer(conv3_1, [3, 3, 16, 2*10], “conv3_2”) gap = tf.reduce_mean(conv3_2, [1,2]) ccn = tf.reshape(gap,[-1,10,nb_CCN]) ccn = tf.reduce_max(ccn, 2)

1.2 train params

L2 on weights (5e-5)

```
In [20]: 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)
```

```
In [21]: """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 [27]: """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)
```

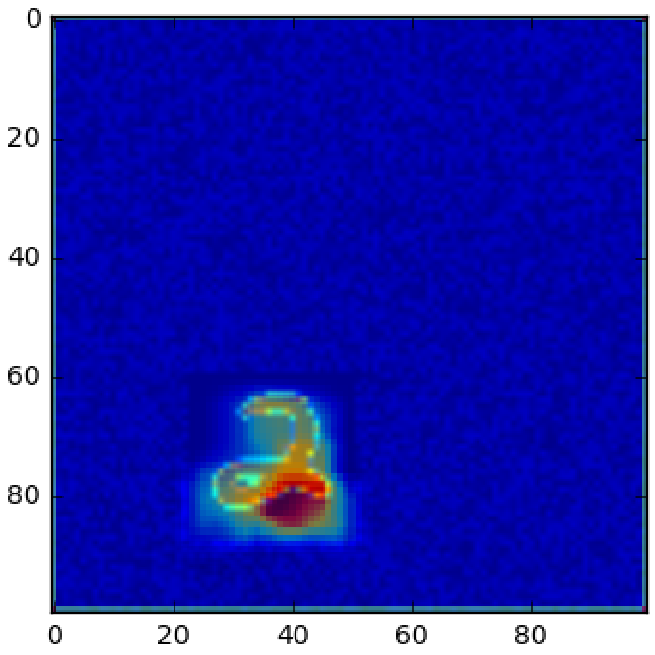
```
***** EPOCH 0 ******
1-Loss on testset is 0.176172
1-Accuracy now is 95.35
2-Loss on testset is 0.167103
2-Accuracy now is 95.68
lr now is 0.00475
max acc so far : 95.35

***** EPOCH 1 ******
1-Loss on testset is 0.072908
1-Accuracy now is 98.28
2-Loss on testset is 0.068332
2-Accuracy now is 98.55
lr now is 0.00451
max acc so far : 98.28

***** EPOCH 2 ******
1-Loss on testset is 0.111745
```

```
1-Accuracy now is 97.14
2-Loss on testset is 0.103483
2-Accuracy now is 97.42
lr now is 0.00429
max acc so far : 98.28

***** EPOCH 3 ******
1-Loss on testset is 0.210473
1-Accuracy now is 94.53
2-Loss on testset is 0.198733
2-Accuracy now is 94.83
lr now is 0.00407
max acc so far : 98.28

***** EPOCH 4 ******
1-Loss on testset is 0.096334
1-Accuracy now is 97.80
2-Loss on testset is 0.093539
2-Accuracy now is 97.90
lr now is 0.00387
max acc so far : 98.28

***** EPOCH 5 ******
1-Loss on testset is 0.118445
1-Accuracy now is 96.95
2-Loss on testset is 0.111644
2-Accuracy now is 96.96
lr now is 0.00368
max acc so far : 98.28

***** EPOCH 6 ******
1-Loss on testset is 0.087897
1-Accuracy now is 97.90
2-Loss on testset is 0.081676
2-Accuracy now is 98.22
lr now is 0.00349
max acc so far : 98.28

***** EPOCH 7 ******
1-Loss on testset is 0.055968
1-Accuracy now is 98.96
2-Loss on testset is 0.052658
2-Accuracy now is 98.97
lr now is 0.00332
max acc so far : 98.96

***** EPOCH 8 ******
1-Loss on testset is 0.060173
```

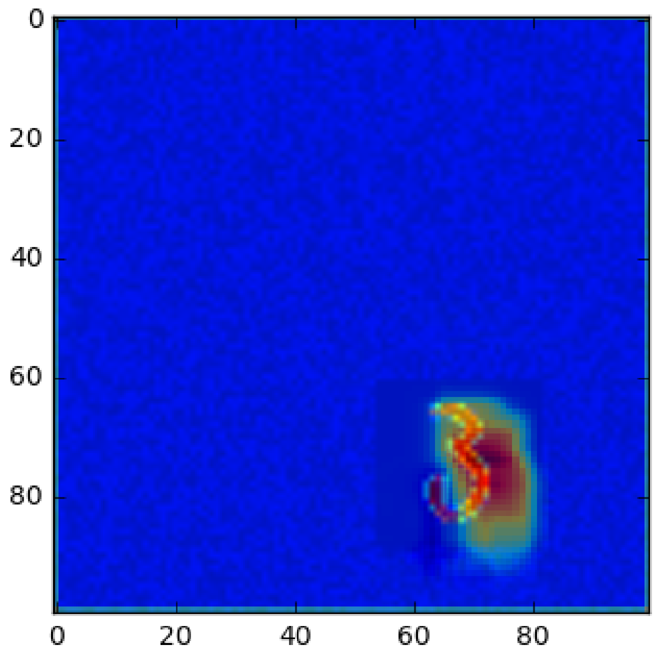
```
1-Accuracy now is 98.72
2-Loss on testset is 0.057280
2-Accuracy now is 98.87
lr now is 0.00315
max acc so far : 98.96

***** EPOCH 9 ******
1-Loss on testset is 0.062256
1-Accuracy now is 98.73
2-Loss on testset is 0.057922
2-Accuracy now is 98.85
lr now is 0.00299
max acc so far : 98.96
```

```
In [23]: back_size  = 100

         for _ in range(10):
             print '---------------'*5
             imgs, lbls = utils.get_batch('test', 1, back_size, .1).next()

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

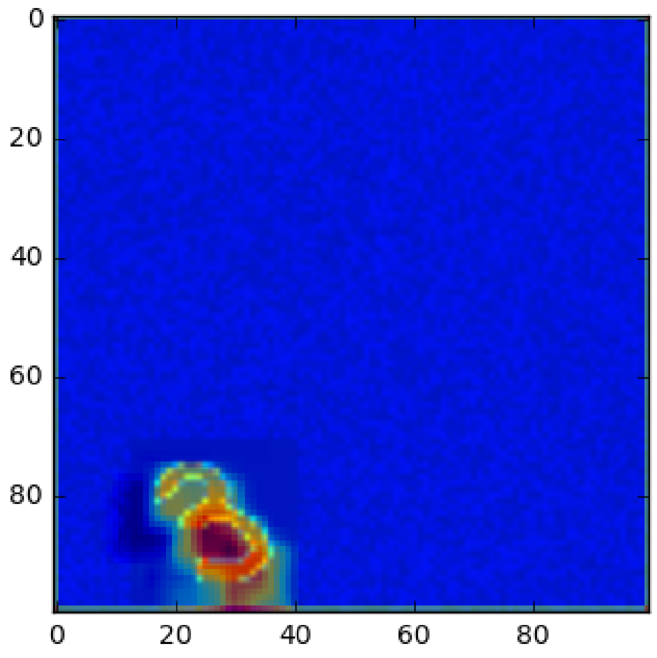
```
---------------------------------------------------------------------------
prediction is : 2 with 31.882
```



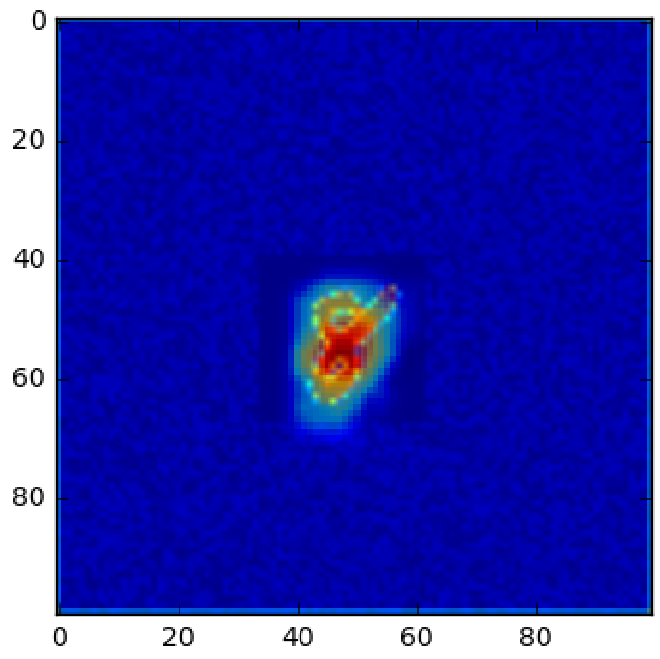
```
--------------------------------------------------------------------------------
prediction is : 3 with 18.150
```



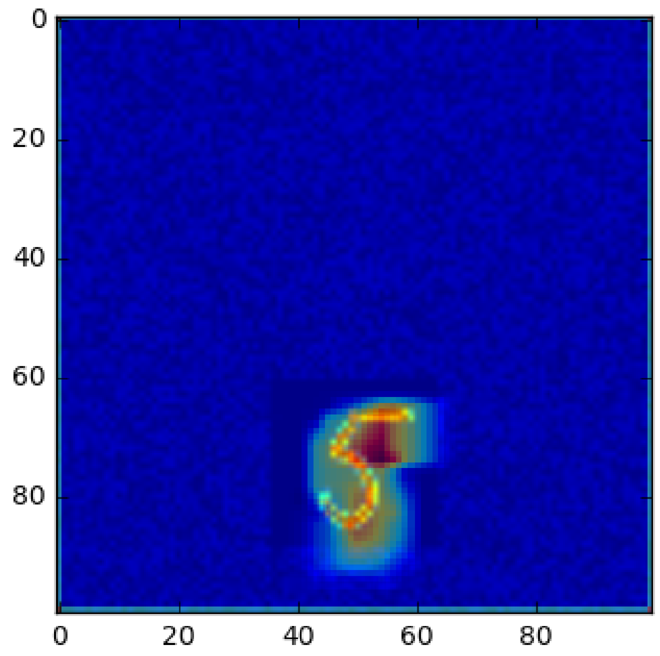
```
--------------------------------------------------------------------------------
prediction is : 3 with 18.778
```



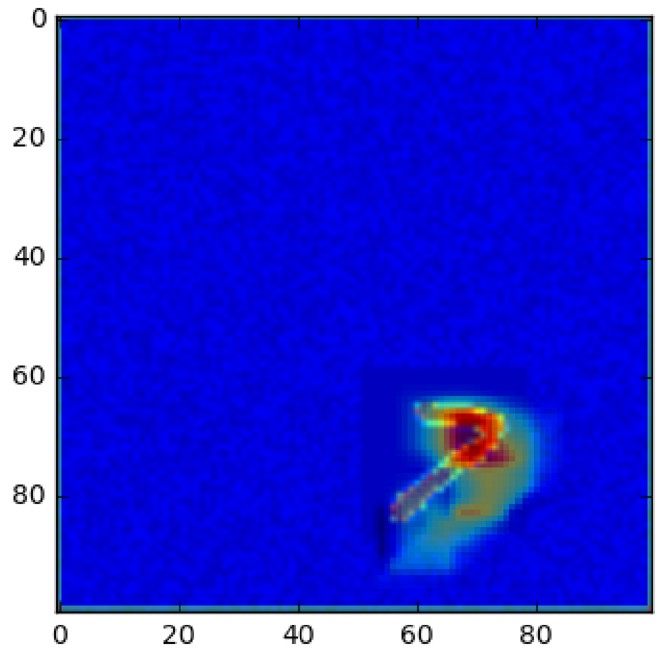
```
--------------------------------------------------------------------------------
prediction is : 8 with 21.397
```



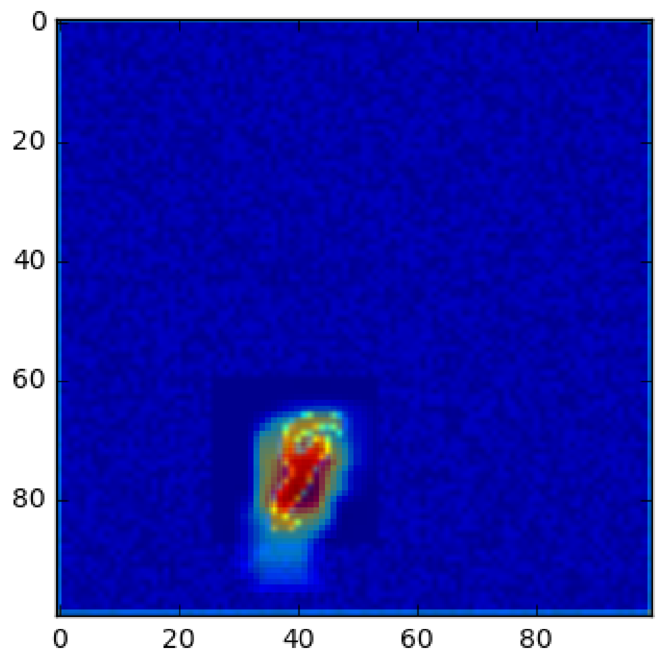
```
--------------------------------------------------------------------------------
prediction is : 5 with 18.432
```



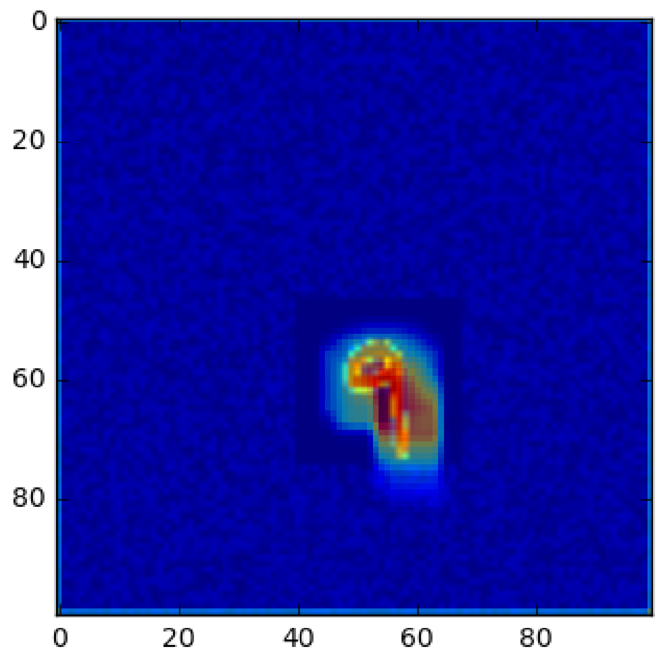
```
--------------------------------------------------------------------------------
prediction is : 7 with 19.625
```



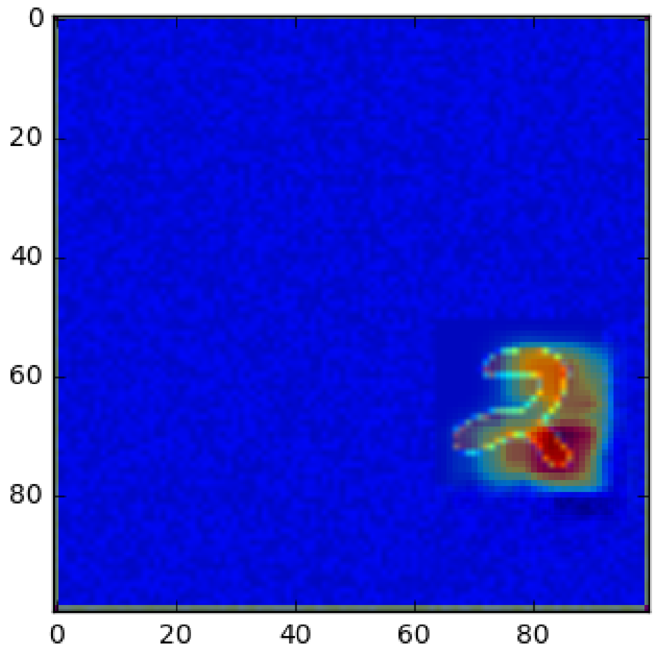
```
--------------------------------------------------------------------------------
prediction is : 8 with 17.069
```



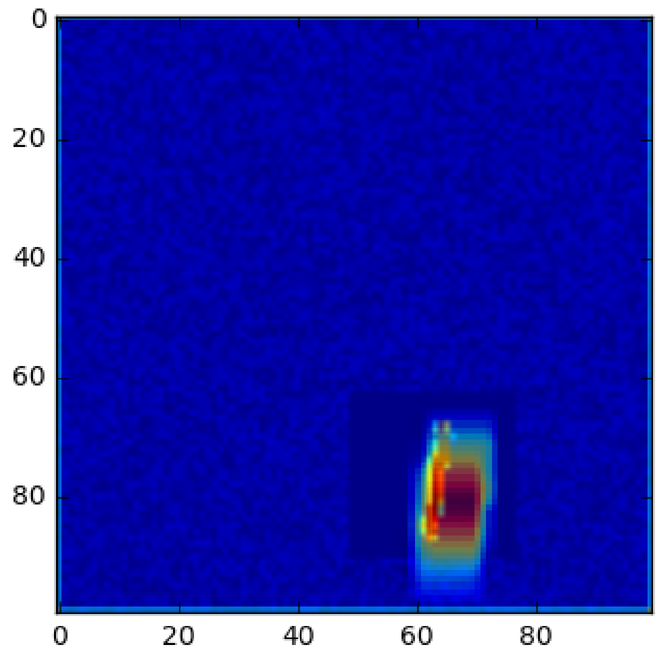
```
--------------------------------------------------------------------------------
prediction is : 9 with 22.256
```



```
--------------------------------------------------------------------------------
prediction is : 2 with 26.576
```



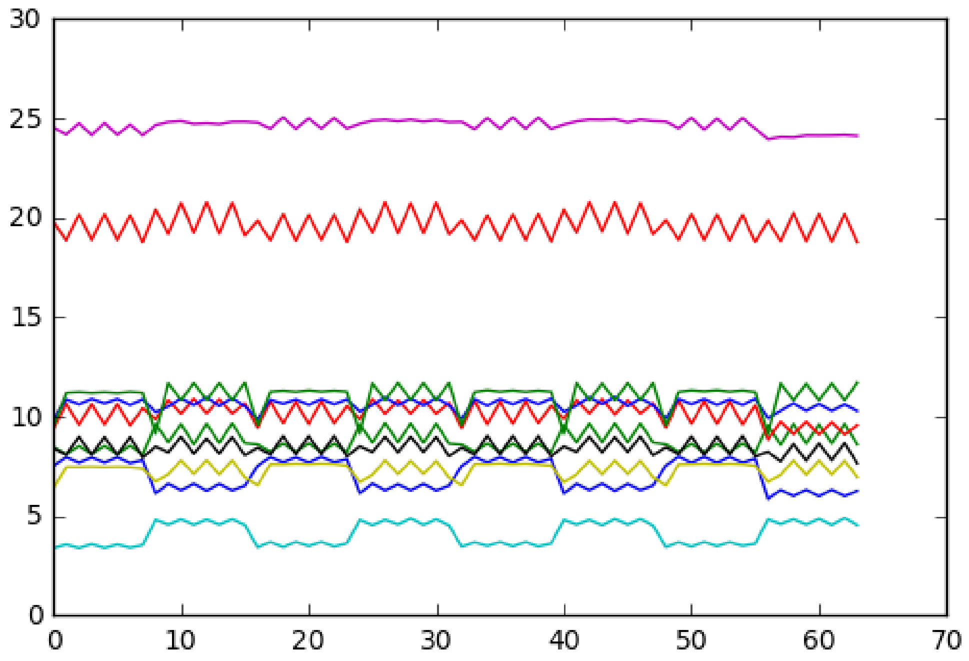
```
--------------------------------------------------------------------------------
prediction is : 1 with 18.319
```

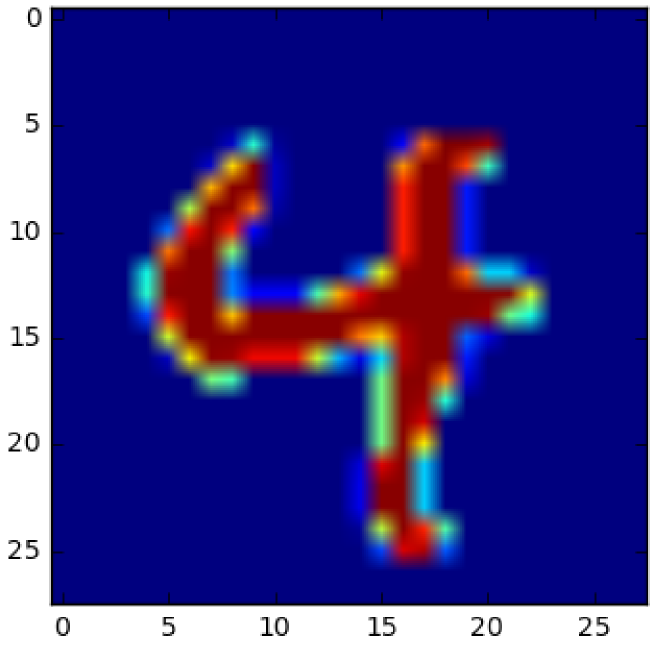


```
In [24]: accuracies = []
         for _ in range(100):
             digit,lbl   = utils.get_batch('test', 1, .1).next()
             digit       = digit[0].reshape(28,28)
             img         = np.random.random((100,100))*.1
             height      = img.shape[0]
             width       = img.shape[1]
             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])
```

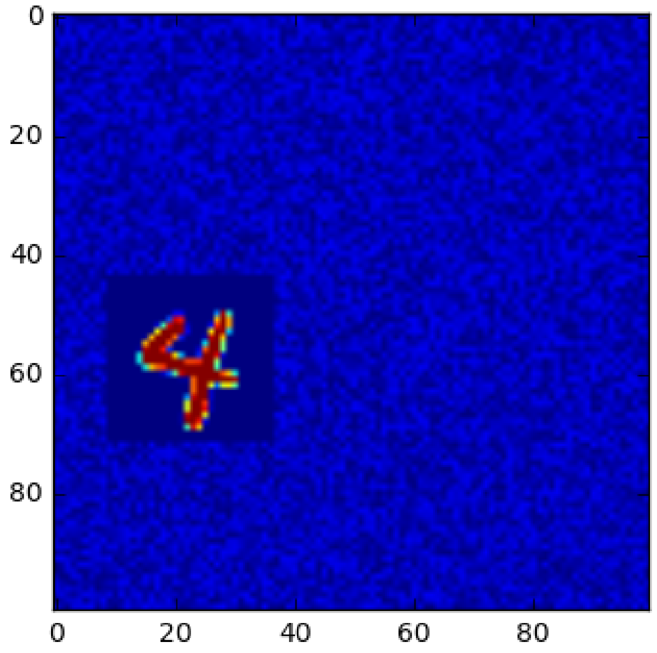
```
            accuracy = sum(preds.argmax(axis=1) == lbl) / float(len(preds))
            accuracies.append(accuracy)

        print sum(accuracies)/len(accuracies)

0.934375
```

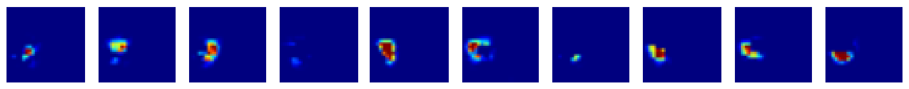
```
In [25]: digit,lbl   = utils.get_batch('test', 1, .1).next()
         digit       = digit[0].reshape(28,28)
         img         = np.random.random((100,100))*.1
         height      = img.shape[0]
         width       = img.shape[1]
         box_size    = 28
         step_size   = 10
         n_x_boxes   = (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()

[4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4]
100.0
```

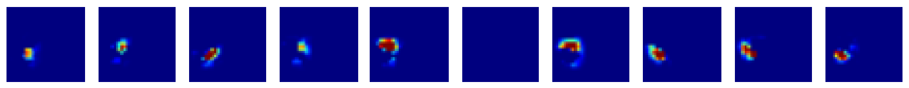


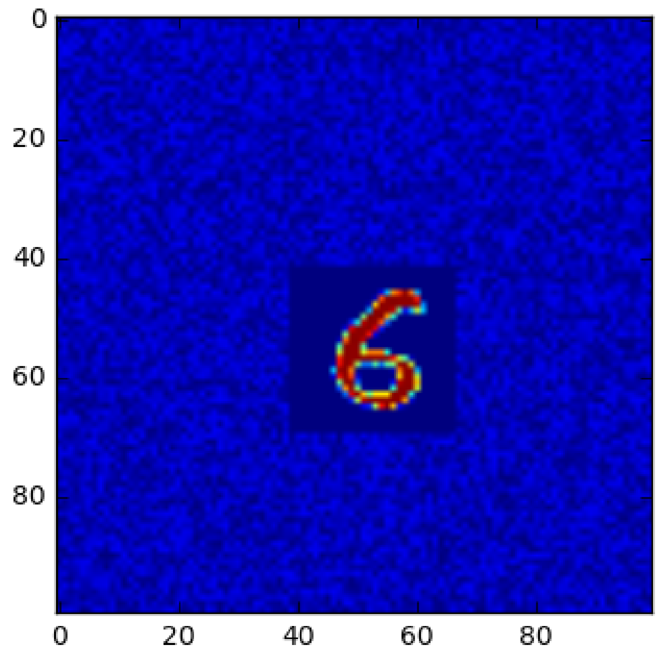


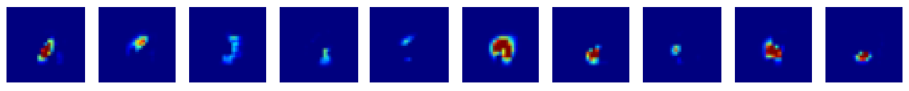
```
In [26]: nb_CCN = 2
         for _ in range(10):
             simple_model.plot_classes_maps(100, 2)
```

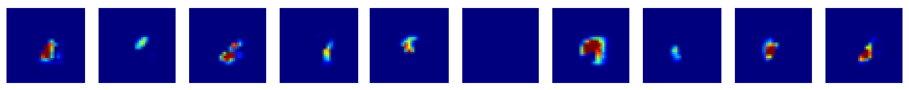

0
20
40
60
80
0
20
40
60
80

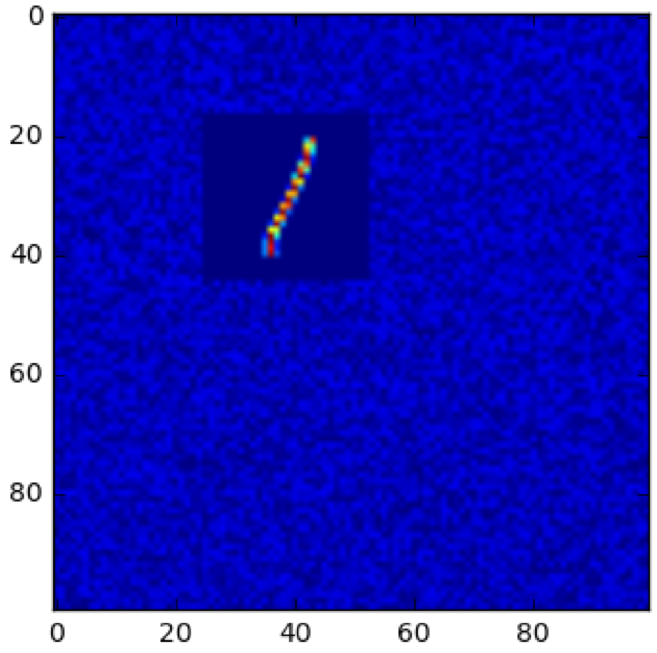


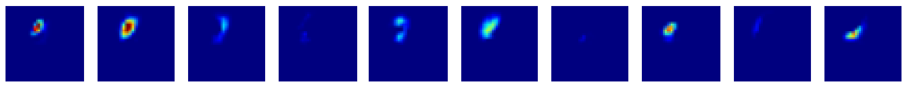


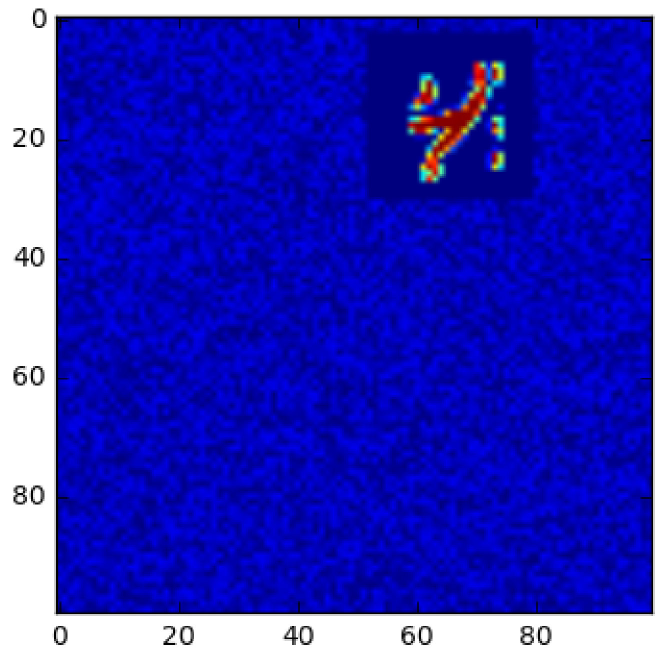

0
20
40
60
80
0
20
40
60
80

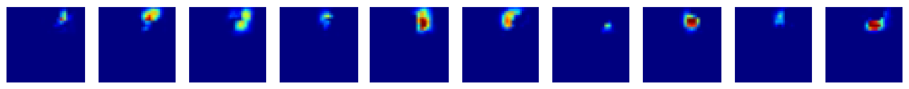


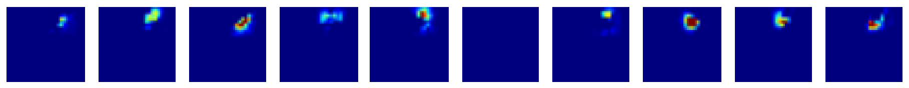


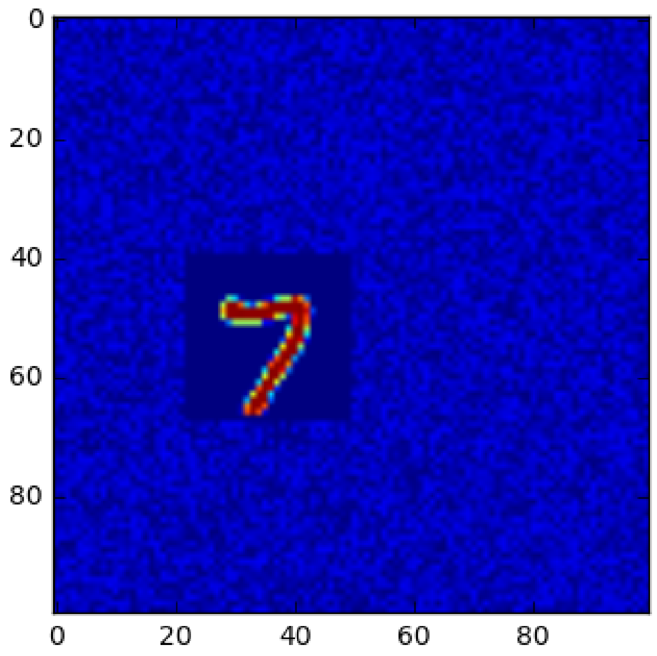

0
20
40
60
80
0
20
40
60
80

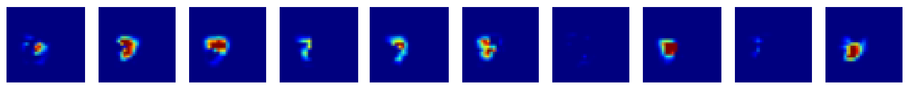


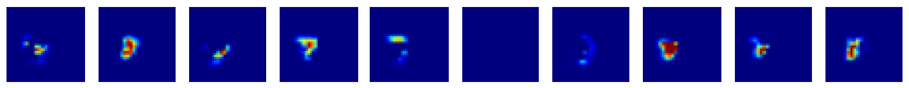

0
20
40
60
80
0
20
40
60
80

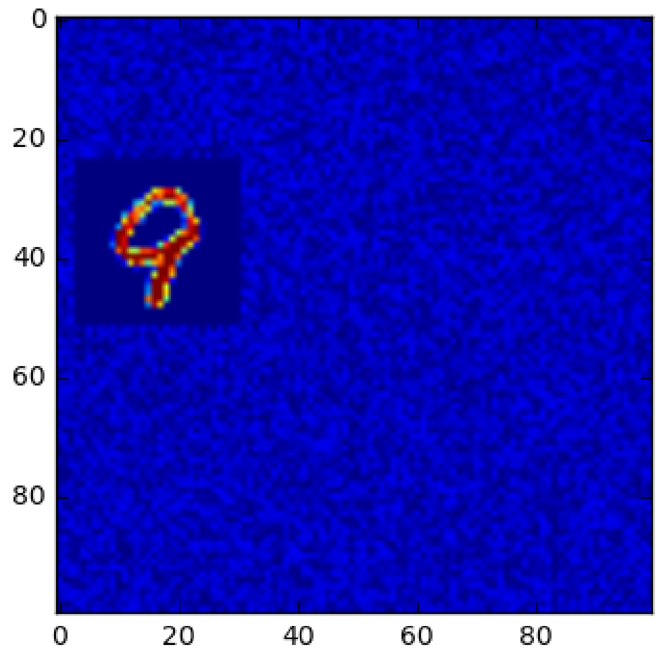


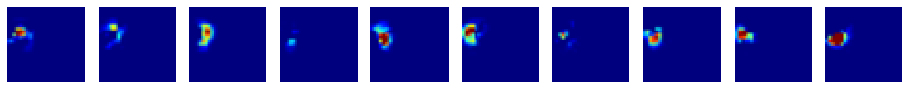


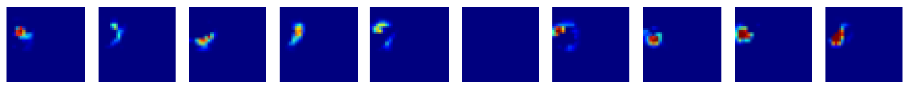

0
20
40
60
80
0
20
40
60
80

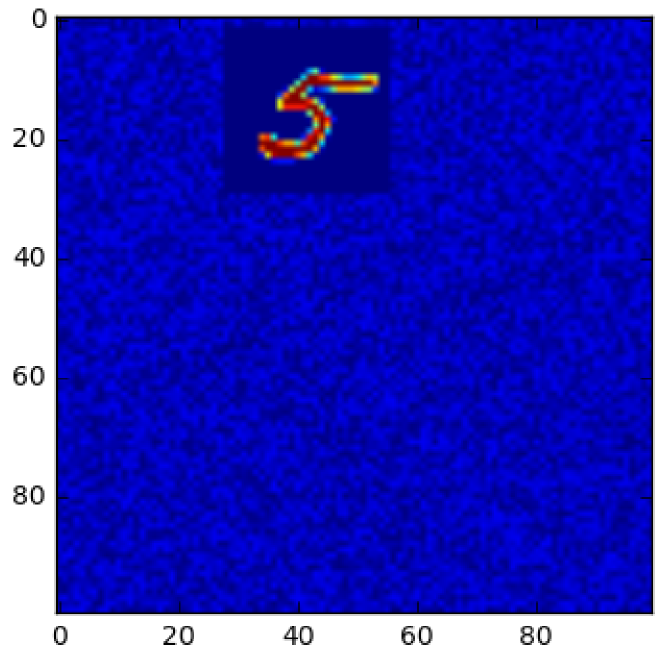


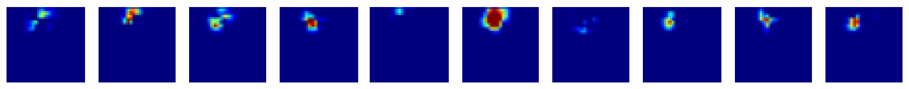


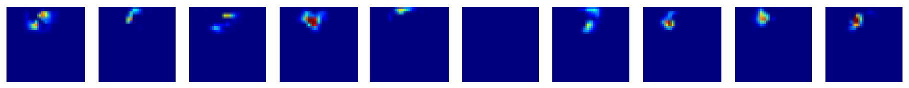

0
20
40
60
80
0
20
40
60
80

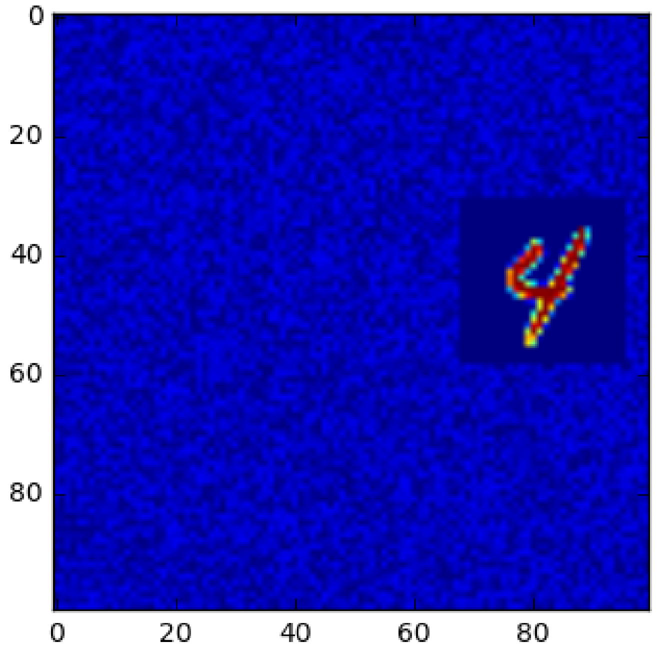


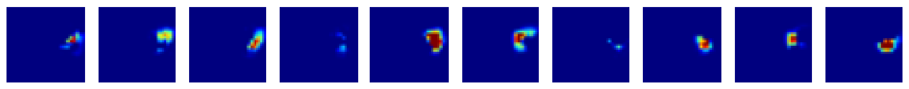


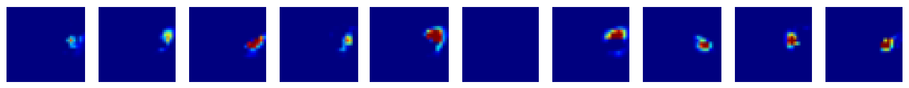

0
20
40
60
80
0
20
40
60
80

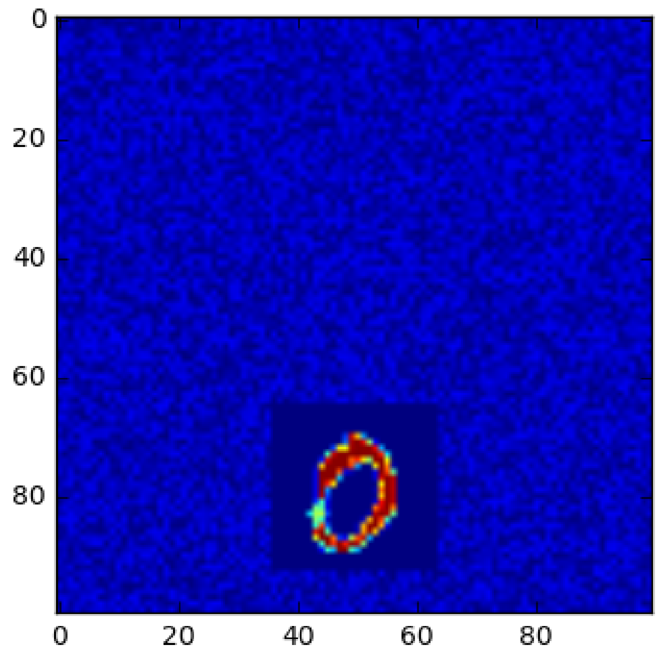




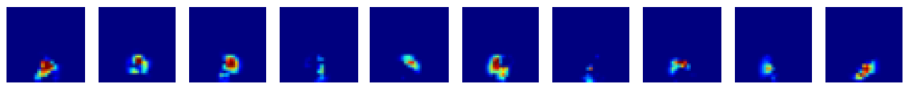

0
20
40
60
80
0
20
40
60
80

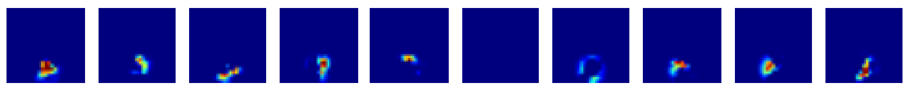


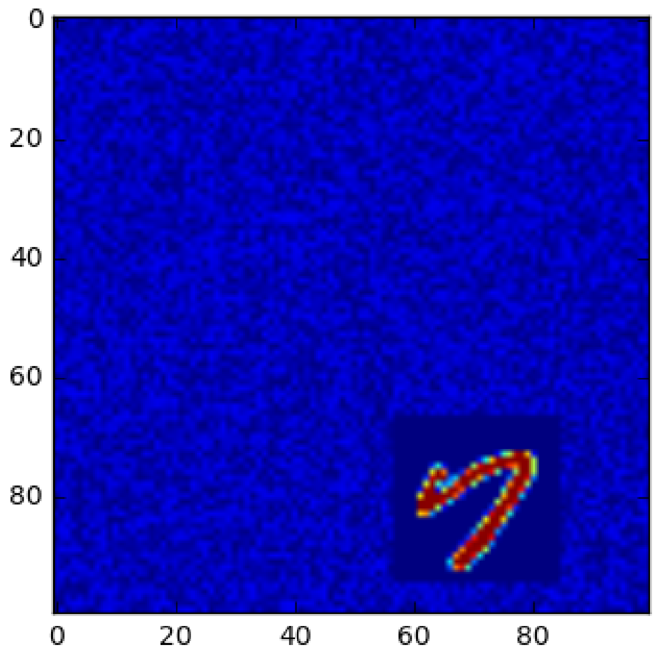


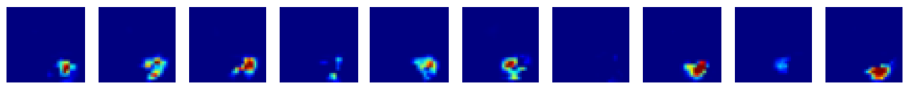


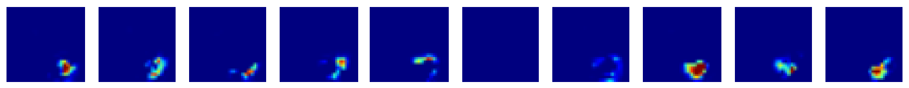
0
20
40
60
80
0
20
40
60
80











```
In [ ]:
```