mnist_unet

November 1, 2018

```
In [ ]: import tensorflow as tf
        import numpy as np

        import innvestigate
        import innvestigate.utils as iutils
        from innvestigate.utils.visualizations import heatmap

        from keras.models import Model
        from keras.layers import concatenate, Input, Conv2D, MaxPooling2D, Conv2DTranspose, Acti
        from keras.layers.normalization import BatchNormalization

        from tqdm import tqdm_notebook as tqdm

        import matplotlib.pyplot as plot
        import numpy as np
        import os
        %matplotlib inline

In [ ]: from tensorflow.examples.tutorials.mnist import input_data
        mnist = input_data.read_data_sets('MNIST_DATA', one_hot=True)

In [41]: def ConvBNRelu(input_tensor, filters):
             out = Conv2D(filters, (3, 3), activation=None, padding='valid', use_bias=False)(inp
             out = BatchNormalization(axis=-1)(out)
             return Activation('relu')(out)


         def get_model(weights=None, flatten_output=False):
             filters = 4
             inputs = Input((28, 28, 1), name="input")

             conv1 = ConvBNRelu(inputs, filters)
             conv2 = ConvBNRelu(conv1, filters)

             pool1 = MaxPooling2D(pool_size=(2, 2))(conv2)

             conv3 = ConvBNRelu(pool1, 2*filters)
             conv4 = ConvBNRelu(conv3, 2*filters)
```

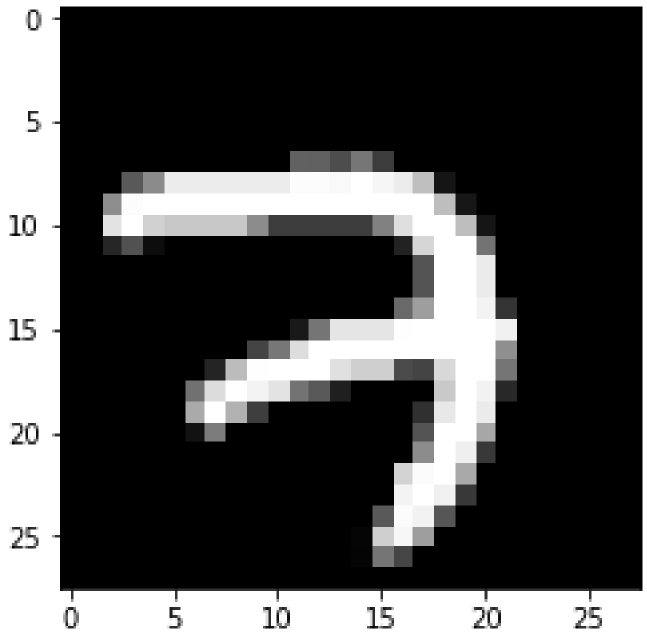
```
            up1 = Conv2DTranspose(filters, (2, 2), strides=(2, 2), activation='relu', padding='
            up1 = concatenate([up1, Cropping2D(cropping=(4, 4))(conv2)])

            conv5 = ConvBNRelu(up1, filters)
            conv6 = ConvBNRelu(conv5, filters)

            logits = Conv2D(1, (1, 1), activation="sigmoid")(conv6)

            if flatten_output:
                logits = Flatten()(logits)

            model = Model(inputs=[inputs], outputs=[logits])
            model.compile(loss="binary_crossentropy", optimizer="adam")
            if weights is not None:
                model.load_weights(weights)
            return model
```

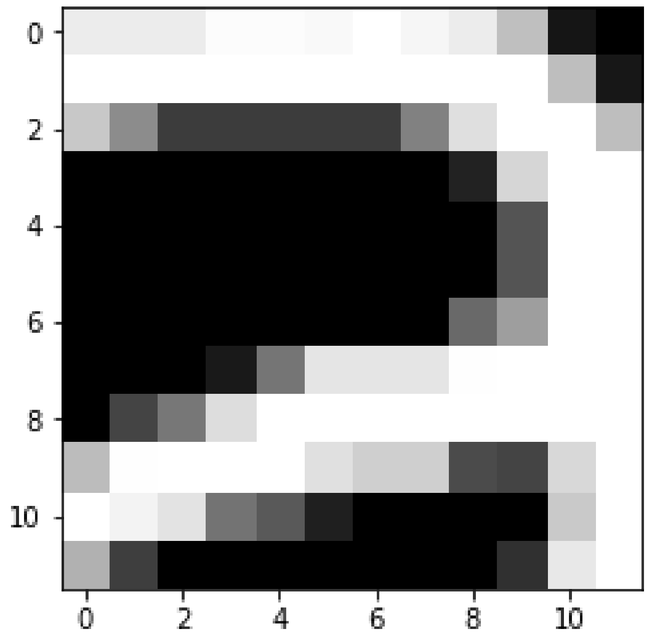
```
In [43]: model = get_model()


         batch_size = 32

         x = mnist.train.images
         count = len(x)
         x = np.reshape(x, (count, 28, 28, 1))
         y = np.where(x > 0.5, 1, 0)
         y = y[:, 8:-8, 8:-8, :]
         model.fit(x, y, batch_size=batch_size, epochs=2)
         model.save_weights("params")
```

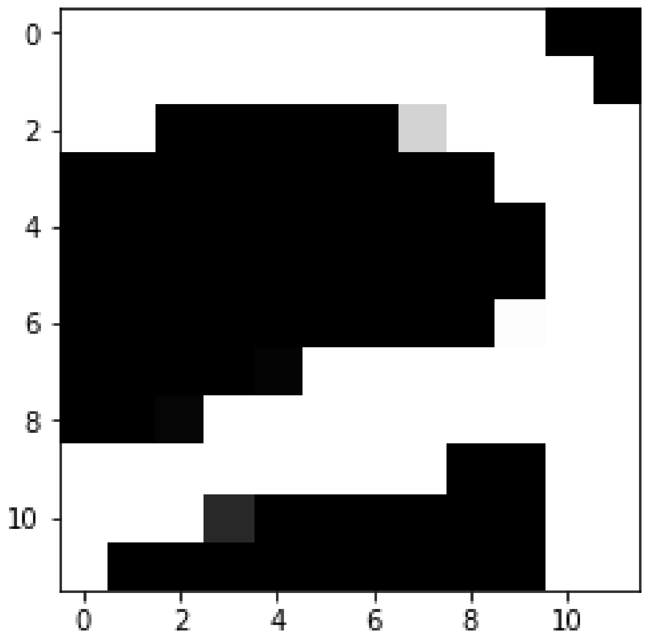
```
Epoch 1/2
55000/55000 [==============================] - 66s 1ms/step - loss: 0.1222
Epoch 2/2
55000/55000 [==============================] - 58s 1ms/step - loss: 0.0526
```

```
In [53]: model = get_model("params")
         sample = x[0:1]
         plot.imshow(sample.squeeze(), cmap="gray")
         plot.show()

         plot.imshow(sample[:, 8:-8, 8:-8].squeeze(), cmap="gray")
         plot.show()

         p = model.predict(sample)
         p = p.squeeze()
         #p = np.pad(p, (8, 8), mode="constant", constant_values=0)
```

```
i, j = np.unravel_index(np.argmax(p), p.shape)
plot.imshow(p.squeeze(), cmap="gray")
plot.show()
```

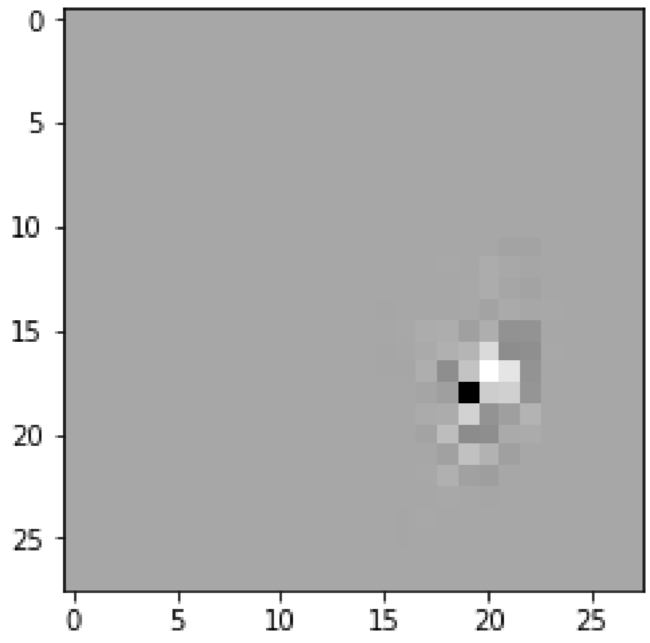




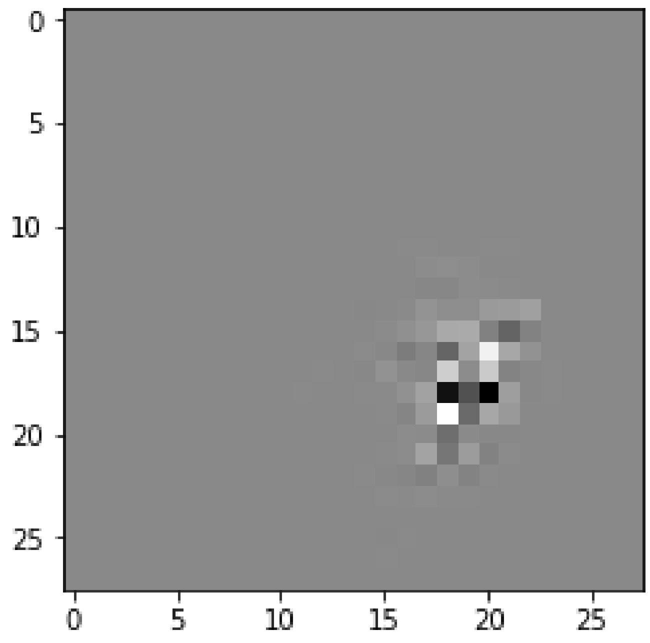


```
In [57]: pixel_x = 11
         pixel_y = 10
         model = get_model("params")
         p = model.predict(sample)
         print('Neuron output at (%d, %d) = ' % (pixel_y, pixel_x), p[0, pixel_y, pixel_x, 0])
         index = np.ravel_multi_index((0, pixel_y, pixel_x, 0), p.shape)
         flatten_model = get_model("params", flatten_output=True)
         flat_p = flatten_model.predict(sample)
         print('Flattened model neuron output at (%d, %d) = ' % (pixel_y, pixel_x), flat_p[0, in
         for _ in range(3):
             analyzer = innvestigate.analyzer.LRPEpsilon(flatten_model, input_layer_rule=(0, 1),
             analysis = analyzer.analyze(sample, index)
             print("relevance at index = ", analysis[0, pixel_y+8, pixel_x+8, 0])
             print("analysis.sum() = ", analysis.sum())
             #analysis = heatmap(analysis)
             plot.imshow(analysis.squeeze(), cmap='gray', interpolation='nearest')
             plot.show()
```

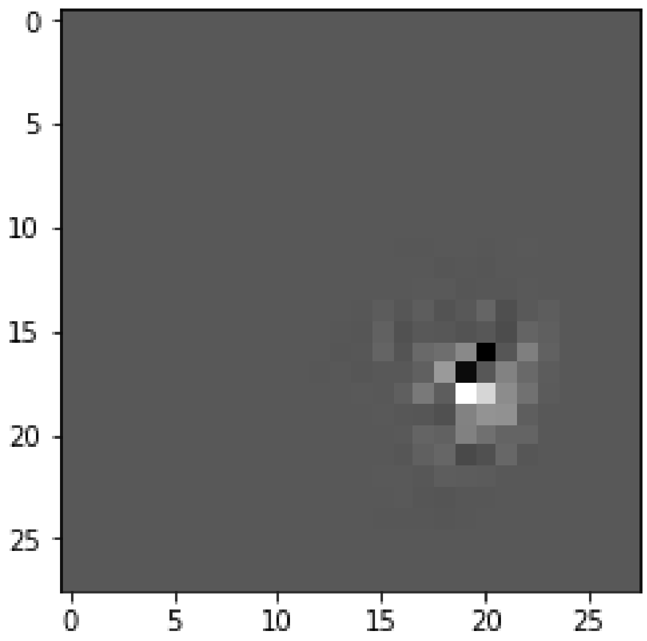
```
Neuron output at (10, 11) =  0.99998176
Flattened model neuron output at (10, 11) =  0.99998176
relevance at index =  -8.912141
analysis.sum() =  3.5663242
```



```
relevance at index =  -0.8532734
analysis.sum() =  3.5663233
```



```
relevance at index =  0.7234432
analysis.sum() =  3.5663233
```



```
In [ ]:
```