1. Interpreting the model

One of the main criticisms of convolutional neural networks is that they are "black boxes" and that even when they work very well, it is hard to understand why they work so well. Many efforts are being made to improve the interpretability of neural networks, and this field is likely to evolve rapidly in the next few years. One of the major thrusts of this evolution is that people are interested in visualizing what different parts of the network are doing. Here, I will show you how to take apart a trained convolutional network, select particular parts of the network and analyze their behavior.

2. Selecting layers

Once a model is constructed and compiled, it will store its layers in an attribute called "layers". This is a list of layer objects. For example, here is a network with 2 convolutional layers, followed by a flattening operation and readout with a dense layer.

3. Getting model weights

If we want to look at the first convolutional layer, we can pull it out by indexing the first item in this list. The weights for this layer are accessible through the get_weights() method. This method returns a list with two items. The first item in this list is an array that holds the values of the weights for the convolutional kernels for this layer. The kernels array has the shape 3 by 3 by 1 by 5. The first 2 dimensions denote the kernel size. This network was initialized with kernel size of 3. The third dimension denotes the number of channels in the kernels. This is one, because the network was looking at black and white data. The last dimension denotes the number of kernels in this layer: 5. To pull out the first kernel in this layer, we would use the index 0 into the last dimension. Because there is only one channel, we can also index on the channel dimension, to collapse that dimension. This would return the 3 by 3 array containing this convolutional kernel.

4. Visualizing the kernel

We can then visualize this kernel directly, but understanding what kinds of features this kernel is responding to may be hard just from direct observation.

5. Visualizing the kernel responses

To understand what this kernel does, it might sometimes be even more useful to convolve one of the images from our test set with this kernel and see what aspects of the image are emphasized by this kernel. Here, we pick the fourth image from the test_set, an image of a shoe.

6. Visualizing the kernel responses

We convolve it with the kernel using the function that we created previously, and create a filtered image that is the result of this convolution. This filter seems to like the external edges of this image on the left.

7. Visualizing the kernel responses

We can confirm this by running the convolution over another image. Here, we've picked the fifth image from the set, an image of a t-shirt.

8. Visualizing the kernel responses

We see that here as well, vertical edges on the left side of the object are emphasized and on the right side are de-emphasized.

9. Visualizing the kernel responses

We can do the same thing with another one of our kernels. Here, we select the second kernel, and convolve the t-shirt image with this kernel. This kernel seems to have learned to detect horizontal edges, particularly on the bottom of the object. Taken together, this gives as an intuition for the kinds of things that the first layer of the network has learned, and can help interpreting the results you see from using this network for a particular task.