For this step towards a more complex input, I first modified my backpropagation code to generalize it to more cases than the XOR problem. I made use of classes in Python to persist some the parameters such as alpha throughout the neural network. The code can be found here in this directory.

The parts of this exercise I was most interested in were:

* Interpreting the hidden activations as features.
* Comparing the number of iterations needed to train models with different numbers of hidden nodes.
* Observing possible dead neurons.

I fit a neural network using six hidden nodes. It was interesting to observe how N and M share similar outputs for the third, fifth, and sixth activations but differ in the others. This could indicate those features in the images that differentiate the two. The letters A and H share similar shapes and as a result similar hidden node output for the first activation.

Next, I trained a neural network with ten hidden nodes. I first noticed that while the first network tool approximately 600 epochs to train, this more complicated form took only approximately 300. This seems to show that more complicated neural networks will fit to the training data more quickly.

In observing the hidden node activations for the more complicated network, I noticed that the tenth activation shows similar values for all of the letters. This appears to be a dead node that does not add value. Perhaps this indicates for as simple a problem this is, ten nodes may be overdoing it.