## Part 1

<https://towardsdatascience.com/exploring-activation-functions-for-neural-networks-73498da59b02>

using this link along with the lecture slides to understand the layers and activations, I utilized relu, sigmoid and softmax to increase my accuracy. Relu’s gradient is ALWAYS equal to 1, allowing me to pass the maximum amount of error through the network. Softmax assigns decimal probabilities to each class that MNIST contains, which adds up to 1.0. This constraint helps training converge faster than normal.

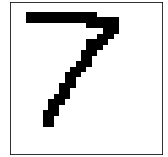
From the slides, we increase the number of epochs to decrease underfitting and tend closer toward optimal curve, while not increasing it too high to overfit

Got 99.8% accuracy with 0.01 loss on the training model

Got 98.2% accuracy with 0.07 loss on the test model

To run, simply go through each code snippet and press play.

## Part 2

I load my model using tf.keras.models.load\_model(sys.argv[2]). I create a prediction using model.predict(img). I determine what the predicted label is using np.argmax(prediction). I used MNIST to identify the number 7, which it incorrectly identifies as 2. I then changed my model to use elu, relu, and softmax, changing the dropout rate and also the dense unit. Doing so, it went from incorrectly identifying 100% 7 as 2, to incorrectly identifying 99% 6.

To run predict.ipynb, upload MNIST\_fix.h5, image.png, and notMNIST.h5.

Run the arguments with dataset: MNIST, mode: MNIST\_fix.h5, image: image.png, class index: 7. Then run the rest of the code.

## Part 3