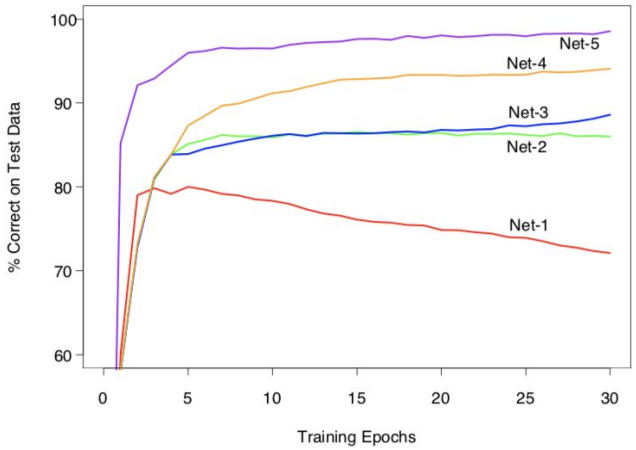

FIGURE 11.11. *Test performance curves, as a function of the number of training epochs, for the five networks of Table 11.1 applied to the ZIP code data. (Le Cun, 1989)*

T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd Edition, Springer, 2017, Figure 11.11

Problem Setting:

The inputs that make up this figure are inputs for each Network based on the number of input pixels (256 = 16x16 for Net-1). There are 10 outputs in this model for each of the digits classes from an image (class k, for k = 0,1,2,....,9). The desired function to learn is the predicted value $\hat{f}_k(x)$.

Data Sources:

The U.S. Postal service uses images that are scanned, resulting in 16x16 greyscale images that need to be classified. With each network, there are misclassification rates around 4.5%. This is the real problem that this figure solves. This graph represents the correctness of test data, over the number of training iterations. There were five different networks that were examined. The first network overfits very quick, where as the second network overfits slightly over the course of the iterations. Net-5 shows the best classification results, based on the highest percent of correct data. I plan to get this data from the Elements of Statistical Learning website (https://web.stanford.edu/~hastie/ElemStatLearn/). There are 7291 training observations(X inputs) and 2007 test observations(Y outputs). Each of these data sets are split into rows/lines. 7291 rows in the training data and 2007 rows in the testing data.

First two rows of data:

6.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.6310 0.8620 -0.1670
-1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000
-1.0000 -0.9920 0.2970 1.0000 0.3070 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000
-1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.4100 1.0000 0.9860 -0.5650 -1.0000
-1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.6830
0.8250 1.0000 0.5620 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000

-1.0000 -1.0000 -1.0000 -0.9380 0.5400 1.0000 0.7780 -0.7150 -1.0000 -1.0000 -1.0000
-1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 0.1000 1.0000 0.9220
-0.4390 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -1.0000
-1.0000 -0.2570 0.9500 1.0000 -0.1620 -1.0000 -1.0000 -1.0000 -0.9870 -0.7140 -0.8320
-1.0000 -1.0000 -1.0000 -1.0000 -1.0000 -0.7970 0.9090 1.0000 0.3000 -0.9610 -1.0000
-1.0000 -0.5500 0.4850 0.9960 0.8670 0.0920

Algorithm(Baselines):

I'm going to have a for loop through every network, every row in the training iterations, and every row in the data for the correctness(%) of the test data. Each network will be plotted, one by one. The predicted values for each network will be calculated then plotted. If I chose this figure, I will learn how to create a neural network for character recognition. I plan on using PyTorch or TensorFlow for this project, as this is what I am most familiar with. I should be able to find packages for these within the python regression package that will allow me to utilize the predicted value function.