Introduction

In this paper, I discuss the implementation of a simple, single-hidden layer neural network for classification. Specifically, the model seeks to sort sixteen representations of English letters into nine distinct classes. My focus in this analysis will not be on the accuracy or efficiency of the model's performance; rather, I will focus on the relationships between the input, hidden, and output nodes after learning. By understanding how the input features affect the hidden nodes, and how the hidden nodes subsequently affect the final predicted classifications, I gain a deeper understanding of the internal processes of neural networks.

As mentioned, the model consists of a single hidden layer. The input layer has eighty-one nodes, the single hidden layer has six, and the output layer has nine. As I am chiefly concerned with hidden node behavior, the number of hidden nodes was set at six and not tested to see if it provided the best reduction in SSE.

My hypothesis for the hidden node interaction is that the hidden nodes will pick up not only clearly identifiable features of the inputs (e.g. long, straight, vertical lines), but also focus on patterns of *deactivated* inputs (e.g. letters with ample whitespace in the center region, such as D and O). Some of the hidden nodes will focus on activated input patterns, while some will focus on deactivated input patterns. Additionally, I believe that the output nodes will struggle to strongly predict across similar classes due to the relatively small number of hidden nodes-- class 1 (which contains the letters 'B' and 'P') and class 4 (containing "E" and "F") will be too similar for the model to confidently distinguish between, for example.

Data and Output Overview

Data Overview

The input data consists of sixteen unique letters, A-P. Each letter is represented by a 9x9 grid, with the “filled-in” blocks working together to create an image of the letter. Of course, the 81-node input layer only “sees” the ‘K’ represented as a one-dimensional array (i.e. an 81x1 format). While the dimensionality of the layout may be different, with the 9x9 grid being easier for humans to interpret, it is possible to mentally map the structure of the input array back to the visual representation of the letter. In Figure 1, we see a table showing each letter, ordered by its desired class number, with the blue blocks signaling activation for each of the first 18 input nodes. Nodes X0 through X8 correspond to the top, 9-element row in our 9x9 matrix, while the next nine nodes correspond to the second row.

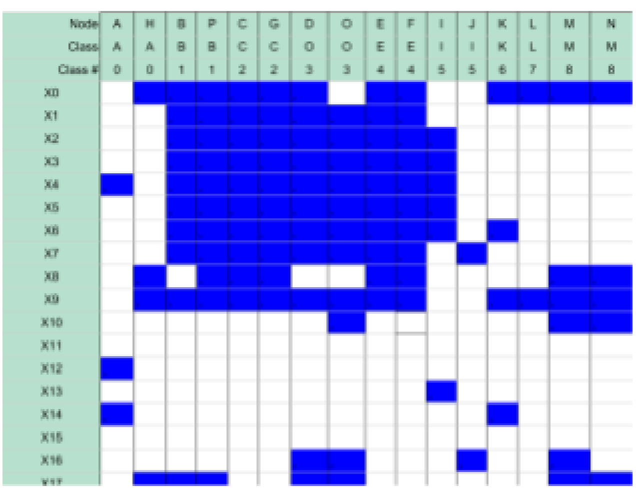

Figure 1

We can quickly see some patterns developing- letters that have a long, horizontal top row have nodes X0 through X8 activated with a one. We can also spot some potential challenges within classes- while ‘I’ and ‘J’ are both Class 5 representations, the letter I sees activations at nodes X2 through X6, while ‘J’ does not. These types of patterns continue throughout the 81 input nodes, but for length considerations, I will not include them all here.

Output Overview

While I am not overly concerned with the accuracy or efficiency of the model results for the purposes of this paper, it is helpful to review the outputs to see where the model may have run into some difficulty in assigning the correct class. Table 1 below shows us the desired outputs from the model. The blue fields indicate the desired output of one for each input letter and output node/class.

Output Node	Class Name	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
o0	A	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
o1	B	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
o2	C	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
o3	D	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
o4	E	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
o5	I	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
o6	K	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
o7	L	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
o8	M	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

Table 1

In Table 2, we see the actual outputs. The model did exceptionally well on most input letters--the correct class was identified for 15 of the 16 inputs. While the confidence is a bit lower for letters of the Class D (letters ‘O’ and ‘D’), it really only failed to make a correct choice for the sole member of Class C. We’ll investigate why identifying the letter ‘C’ might have caused some issues.

Output Node	Class Name	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
o0	A	0.91	0.01	0	0	0.04	0.04	0.01	0.91	0	0	0.03	0	0.05	0.08	0	0.02
o1	B	0.02	0.95	0.01	0.03	0.03	0.05	0.92	0.03	0.03	0.02	0.06	0	0.01	0.01	0.03	0.89
o2	C	0.03	0.05	0.03	0.03	0.02	0.03	0.05	0.03	0.07	0.07	0.03	0.03	0.04	0.04	0.03	0.04
o3	D	0.01	0.08	0.02	0.86	0	0	0.08	0.01	0.07	0.06	0.01	0.08	0.12	0.12	0.86	0.06
o4	E	0.05	0.04	0.04	0	0.93	0.92	0.04	0.05	0.03	0.03	0.02	0.07	0	0	0	0.04
o5	I	0	0.04	0.06	0.02	0.02	0.03	0.04	0	0.92	0.92	0	0.05	0.01	0.01	0.02	0.02
o6	K	0.03	0.03	0.01	0.01	0.01	0.01	0.03	0.03	0	0	0.89	0.07	0.04	0.01	0.01	0.04
o7	L	0.01	0	0.02	0.01	0.02	0.01	0	0.01	0.05	0.05	0.08	0.9	0.02	0.01	0.01	0
o8	M	0.06	0	0.01	0.07	0	0	0	0.07	0.01	0.01	0.06	0.01	0.92	0.9	0.07	0

Table 2