NORTHWESTERN UNIVERSITY | SCHOOL OF PROFESSIONAL STUDIES

The Role of Hidden Nodes in Classification Neural Network

Yining Feng

Northwestern University

MSDS-458 Assignment 1: Classifier Neural Networks

Abstract

Artificial neural networks (ANNs) are powerful tools for data analysis and are particularly suitable for pattern recognition and classification. Nevertheless, their utility has been critically limited because the interpretation of the "black box" model is difficult. To address this important shortcoming of neural network modeling methods, this study analyzes the behavior of the hidden layer nodes in order to understand how the hidden layer in a feedforward single-hidden layer network has learned to represent features within the input data. This investigation into the role of hidden nodes can further offer preliminary insights into the underlying mechanisms of deep learning networks.

1. Introduction & Problem Statement

Artificial neural networks (ANNs) are highly parameterized, non-linear models with sets of processing units called neurons that can be used to approximate the relationship between input and output signals of a complex network. While ANNs can be used as powerful predicting tools compared to more conventional models (e.g., linear regression), they are also criticized as "black boxes". Compared to linear methods, ANN models are very difficult to interpret and it is challenging to identify the function of each neuron and how these neurons are connected to generate the output. The hyper-parameterized structure of neural networks creates complex functions from the input that can approximate observed outcomes with minimal error.

Recognizing the issue, I propose an examination of the role of each hidden node in a feedforward single-hidden layer network in order to understand the underlying mechanisms of ANN. The classification capability of a neural network will be demonstrated using a total of 26 distinct alphabet letters, providing an overview of how hidden nodes respond to each different class of input data. The Python programming language (version 3.7.4) is used in this study.

2. Research Design and Modeling Method

Data

The study is based on a collection of binary data which represents 26 alphabet letters. Each letter is made up of 81 binary inputs that form a 9×9 matrix of the letter. Besides the 81 binary inputs, the training data consists of one variable before and four list variables after the 81-element input grid.

As Figure 1 in the Appendix shows, the 1st element is the "number" of the training dataset, starting at 1 and going up to the total number of training instances used. The element right after the 81-element list represents the "number" of the corresponding letter in the alphabet for this training data set, starting at 1 and going up to the total number of training instances used. The following element stands for the "letter" of the alphabet corresponding to this particular training data set. The next element refers to the "number" corresponding to the "big shape" class used to classify different letters into groups based on their skeletal shapes. The last element is the corresponding string literal for the "big shape" class. There are 8 "big shape" classes in total.

Neural Network

The feedforward neural network in this study has 81 binary inputs and one hidden layer of 6 hidden nodes fully connected to the input layer. The output layer of the neural network consists of 8 nodes, with one node for each "big shape" alphabetic class classification. The classification by the network of an alphabet letter is taken to be the node with the highest activation value, which represents the network's confidence classifying that letter into a certain "big shape" class.

In this neural network, the forward pass involves computing the predicted output, which is a function of the weighted sum of the inputs given to the neurons. A sigmoid transfer function with alpha set to one is used to calculate the output of the hidden and output layers. Based on the comparison of the predicted output with the expected output, the weights for both the hidden and output layers are changed using backpropagation. Backpropagation is performed using the gradient descent algorithm by taking the sum of squared errors (across the output layer) and using the chain rule (partial derivatives) to determine how each weight depends on that particular error. The neural network is then trained with a learning rate of 0.5 under a limit of 5,000 epochs using sampling without replacement for each epoch. The resulting total sum of squared errors reaches the set epsilon threshold of 0.01. The weights to the hidden and output layers, as well as the bias for each node, are initialized to random values. The connection weights are optimized through backpropagation to minimize the sum of squared errors (SSE).

Experiment

The experiment was devised to understand how tuning different parameters (e.g. alpha, *eta*, epochs, etc.) can impact the classification accuracy of a neural network. In addition, the size of the hidden layer was varied during the creation and training of single-hidden layer neural networks. Each neural network tested had the same base structure. The input layer consisted of 81 nodes receiving input from the preprocessing module. These nodes corresponded to the 9×9 binary matrix produced by the preprocessing module. The output layer consisted 10 nodes, the output of which is ideally high at the output node corresponding to the appropriate "big shape" class, and low at every other output node. The hidden layer structure, however, is architecture dependent. The number of hidden nodes was changed and six values were tested for the number of nodes in the hidden layer of each neural network architecture: 6, 8, 10, 15, 20, and 30. These values were obtained based on a review of discussion board results in order to provide a diverse group of values without creating excessive computation. If the neural network is successfully structured and trained, it must be able to classify letters at a better rate than chance guessing. Random guessing would result in a $\frac{1}{26}$ or 3.85% chance of correct classification.

3. Results

Overall Result

The sum of squared error (SSE) reflecting the performance of the neural network is calculated based on the classification of 8 output classes instead of each alphabet letter. Having 6 hidden nodes performed the best overall because the neural network yielded a relatively low squared error for the classification of all alphabet letters, and performed the best in this experiment with 10 letters including “B”, “D”, “F”, “I”, “J”, “N”, “O”, “Q”, “S”, “Z”. These letters belong to output classes 1, 2, 4, 5, and 6, respectively. The accuracy of a feedforward single-hidden layer neural network with 6 hidden nodes is 88.46% with SSE value of 4.64 as the best result, whereas the accuracy of the neural network with 30 hidden nodes is 73.08% with SSE value of 15.40 as the worst result. Increasing the number of hidden nodes is not helpful for improving the overall performance of the neural network. It is interesting to note that the lowest error demonstrated for each letter (the number of nodes that performed the best for a particular letter) varied across all alphabet letters. Essentially there was not a specific number of nodes that performed consistently best for classifying every letter into the right “big shape” class. Based on Table 1 in the Appendix, the neural network has consistently struggled with classifying alphabet letters such as “A”, “H”, and “U” into the right “big shape” classes respectively throughout the experiment. With 6 hidden nodes, the neural network is capable of correctly classifying all the alphabet letters except for “E”, “H”, and “U”. As illustrated by Table 2 in the Appendix, the neural network mistakenly classified these three alphabet letters into output classes 2, 6, and 2 instead of their correct output classes 1, 3, and 7. Increasing the number of epochs seems to only increase the accuracy of classifying the letters into their corresponding “big shape” classes. Decreasing the learning rate (*eta*) also did not have any impact on more accurately classifying “E”, “H” or “U”. Increasing the alpha value essentially polarized the classification accuracy more, and the model became extremely confident for certain letters but less confident for classifying most of the other letters, so ultimately resulting in a poorly performing model. However, interestingly enough, the letter “H” was even more confidently classified into the output class “M”. Increasing the number of nodes in the hidden layer from 6 to 15 and even 30 did not help either, with the accuracy varying more or less insignificantly. After training with noise of 0.1%, as shown by Figure 2 in the Appendix, there was an observable trend of the neural network classifying the majority of alphabet letters into output classes 2 and 6, which correspond to the “big shape” classes “C” and “M” respectively. In contrast, as shown by Figure 3 in the Appendix, the neural network inclined to classify most of the alphabet letters into the output class 4, which corresponds to the “big shape” class “I” during the pre-training stage.

As illustrated by Figure 4 in the Appendix, within the hidden layer, the hidden node 0 is activated most frequently for output classes “A”, “B”, “J”; the hidden node 1 is activated most frequently for output classes “C”, “I”, “M”; the hidden node 2 is activated most frequently for output classes “C”, “H”, “M”, “U”; the hidden node 3 is activated most frequently for output classes “C”, “H”, “M”; the hidden node 4 is activated most frequently for output classes “A”, “B”, “C”, “H”, “I”, “M”; the hidden node 5 is activated most frequently for output classes “M” and “C”. Overall, the hidden node 4 is the most activated hidden node as it can be activated by a wide range of alphabet letters, whereas the hidden node 5 is the least activated hidden node as it can only be activated by two specific “big shape” classes.

Another observation from Figure 4 is that the output classes “A” and “B” can activate both hidden nodes 0 and 4; the output class “C” can activate all hidden nodes except for hidden node 0; the output class “H” can activate hidden nodes 2, 3, 4; the output class “I” can activate hidden nodes 1 and 4; the output class “J” can only activate the hidden node 0; the output class “M” can activate hidden nodes 1, 2, 3, 5; the output class “U” can only activate the hidden node 2. In a nutshell, the output classes “C”, “H”, and “M” can trigger similar activation among hidden nodes, whereas the output classes “A” and “B” can initiate the activation of the same hidden nodes.

Analysis and Interpretation

The results insinuate the heuristic nature of a neural network and the evident impact of the number of hidden nodes on the performance of a neural network. The fact that certain alphabet letters can activate the same hidden nodes implies that each alphabet letter is split into a number of segments or features, and these features can be recognized and clustered into groups by different hidden nodes. As illustrated by Figure 5 in the Appendix, summing up pixels for each alphabet letter highlights the similarity and differences between features of each letter. For instance, letters “C”, “E”, “F”, “G”, “P”, “Q”, “R”, and “S” all share the same length of the highlighted first segment of the visualization. One common feature of these letters is that each of them has a stroke crossing the top row. According to Figure 4, these letters haven been classified into the “big shape” classes “B” and “C”. In addition, it is known that both of these “big shape” classes can activate hidden node 4. Therefore, one plausible explanation is that the hidden node 4 can be activated by letters with pixels in the top row. Conversely, letters “J”, “U”, and “W” which cannot activate the hidden node 4 all have pixels distributed sparsely across all segments in the visualization.

In order to understand the role of hidden nodes, I revisited the backpropagation structure. Technically, the backpropagation algorithm is a method for training the weights in a multilayer feed-forward neural network. I attempted examining the weights assigned to each input or output node and comparing the values and signs of these weights to interpret the role of hidden nodes.

Each of the 81 input nodes, corresponding to the training data that collectively represent a letter, is multiplied by the connection weight w_i between the input layer and the first (and only) hidden layer, with the addition of bias ($b_1 = 0.0648$), to estimate the activation value for the 1st hidden node illustrated by Figure 6 in the Appendix. In Figure 6, the connection weights for each node is indicated right above the connection between nodes. The activation value is then put through an activation, or transfer, function, in this case the sigmoid transfer function, to obtain the value h_1. This illustrated process occurs for every node in the hidden layer. The h_1 value then becomes the hidden layer’s output value, which is then multiplied by each connection weight and then passed to the output layer. As Figure 6 depicts, even 1 node, h_1, contains so much information passed to and from the node. Once the output layer goes through the same process of taking the sum of the hidden layer nodes multiplied by the connection weights, plus the bias, that activation value is again passed through the sigmoid transfer function and then an output o_i is given. In this neural network, each output node represents a letter, and the predicted output figure is how likely the output node correlates to the classification letter.

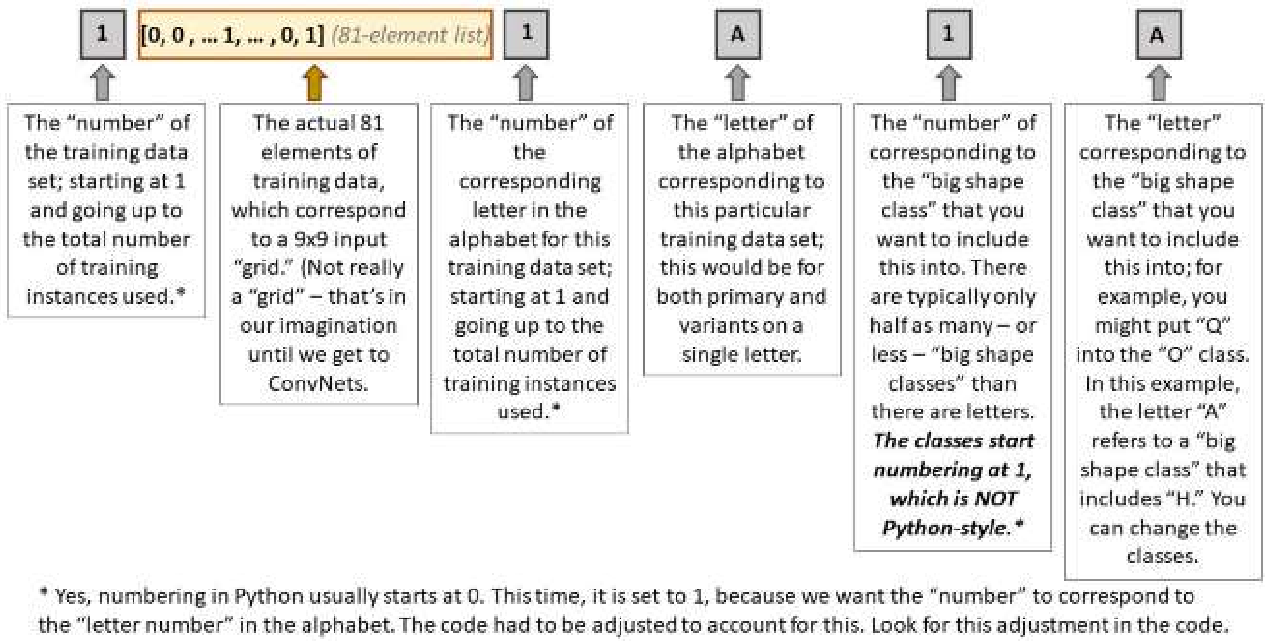
4. Conclusions

The top neural network for classification is a feedforward single-hidden layer network with an input layer of size 81, a single hidden layer with 6 hidden nodes, and an output layer of size 10. As indicated, this neural network performed the best overall with correct classification of 23 alphabet letters, with the best classification of 10 alphabet letters overall (as evidenced by the smallest SSE). Nevertheless, the neural network consistently struggled with classifying alphabet letters such as “A”, “H”, and “U” into the right “big shape” classes respectively throughout the experiment. However, an evaluation of the activation of each hidden node by different letters provided some insights into the role of certain hidden nodes. The fact that certain alphabet letters can activate the same hidden nodes implies that each alphabet letter is split into a number of segments or features, and these features can be recognized and clustered into groups by different hidden nodes. Findings from this study about the role of hidden nodes have provided preliminary insights into the underlying mechanisms of deep learning networks.

Appendix

Figures

Figure 1. The breakdown of training dataset elements created by Dr. Alianna J. Maren



The "number" of the training data set; starting at 1 and going up to the total number of training instances used.*

The actual 81 elements of training data, which correspond to a 9x9 input "grid." (Not really a "grid" – that's in our imagination until we get to ConvNets.

The "number" of the corresponding letter in the alphabet for this training data set; starting at 1 and going up to the total number of training instances used.*

The "letter" of the alphabet corresponding to this particular training data set; this would be for both primary and variants on a single letter.

The "number" of corresponding to the "big shape class" that you want to include this into. There are typically only half as many – or less – "big shape classes" than there are letters. ***The classes start numbering at 1, which is NOT Python-style.****

The "letter" corresponding to the "big shape class" that you want to include this into; for example, you might put "Q" into the "O" class. In this example, the letter "A" refers to a "big shape class" that includes "H." You can change the classes.

* Yes, numbering in Python usually starts at 0. This time, it is set to 1, because we want the "number" to correspond to the "letter number" in the alphabet. The code had to be adjusted to account for this. Look for this adjustment in the code.

Figure 2. Heatmap for predicted output class classification after training

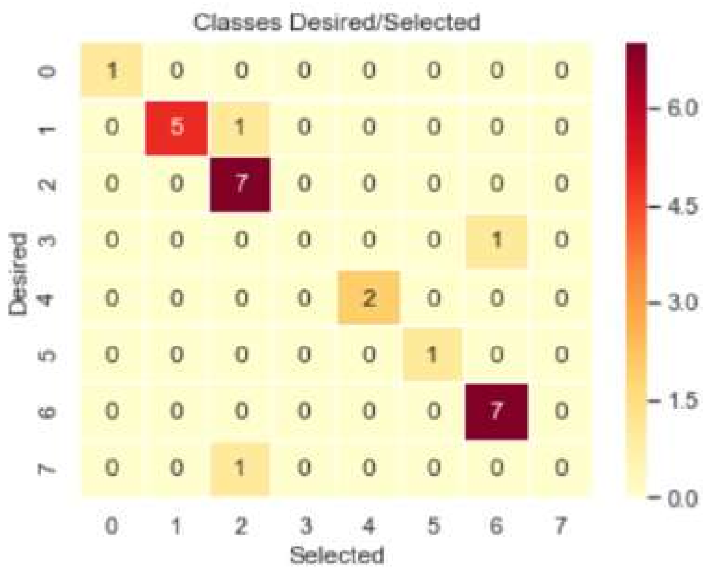
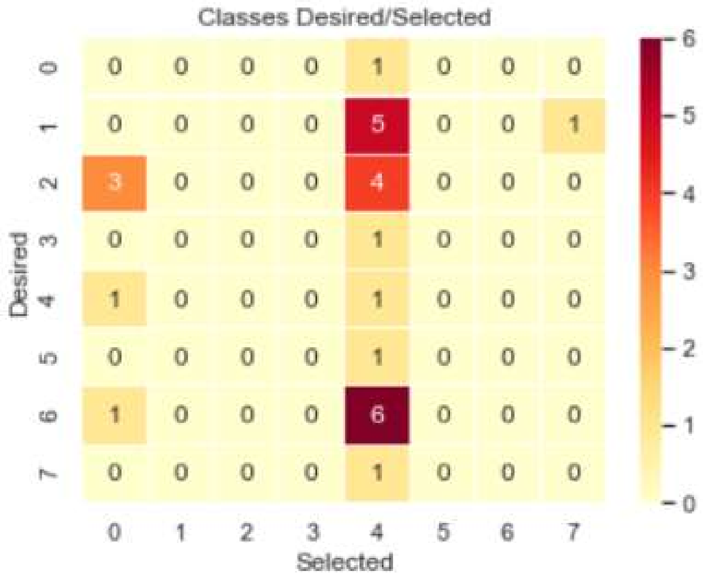


Figure 3. Heatmap for predicted output class classification before training



Desired \ Selected	0	1	2	3	4	5	6	7
0	0	0	0	0	1	0	0	0
1	0	0	0	0	5	0	0	1
2	3	0	0	0	4	0	0	0
3	0	0	0	0	1	0	0	0
4	1	0	0	0	1	0	0	0
5	0	0	0	0	1	0	0	0
6	1	0	0	0	6	0	0	0
7	0	0	0	0	1	0	0	0

Figure 4. Heatmap for hidden node activation associated with "big shape" class across 26 alphabet letters

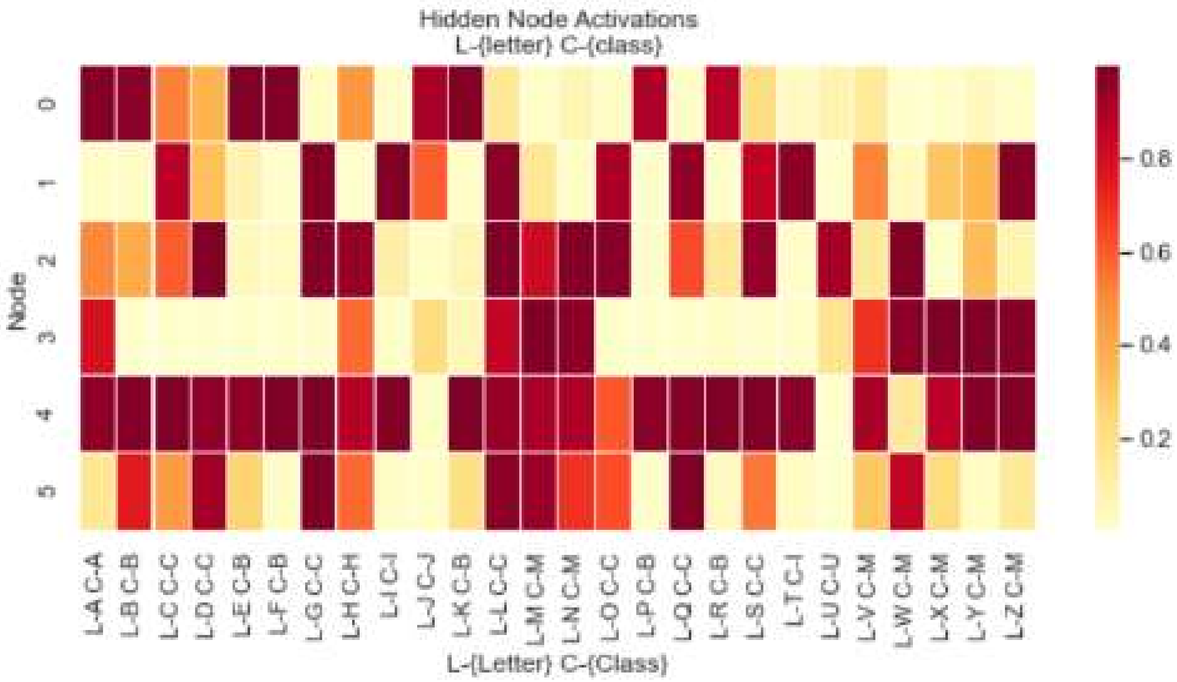


Figure 5. The visualization of total sum of pixels across all alphabet letters

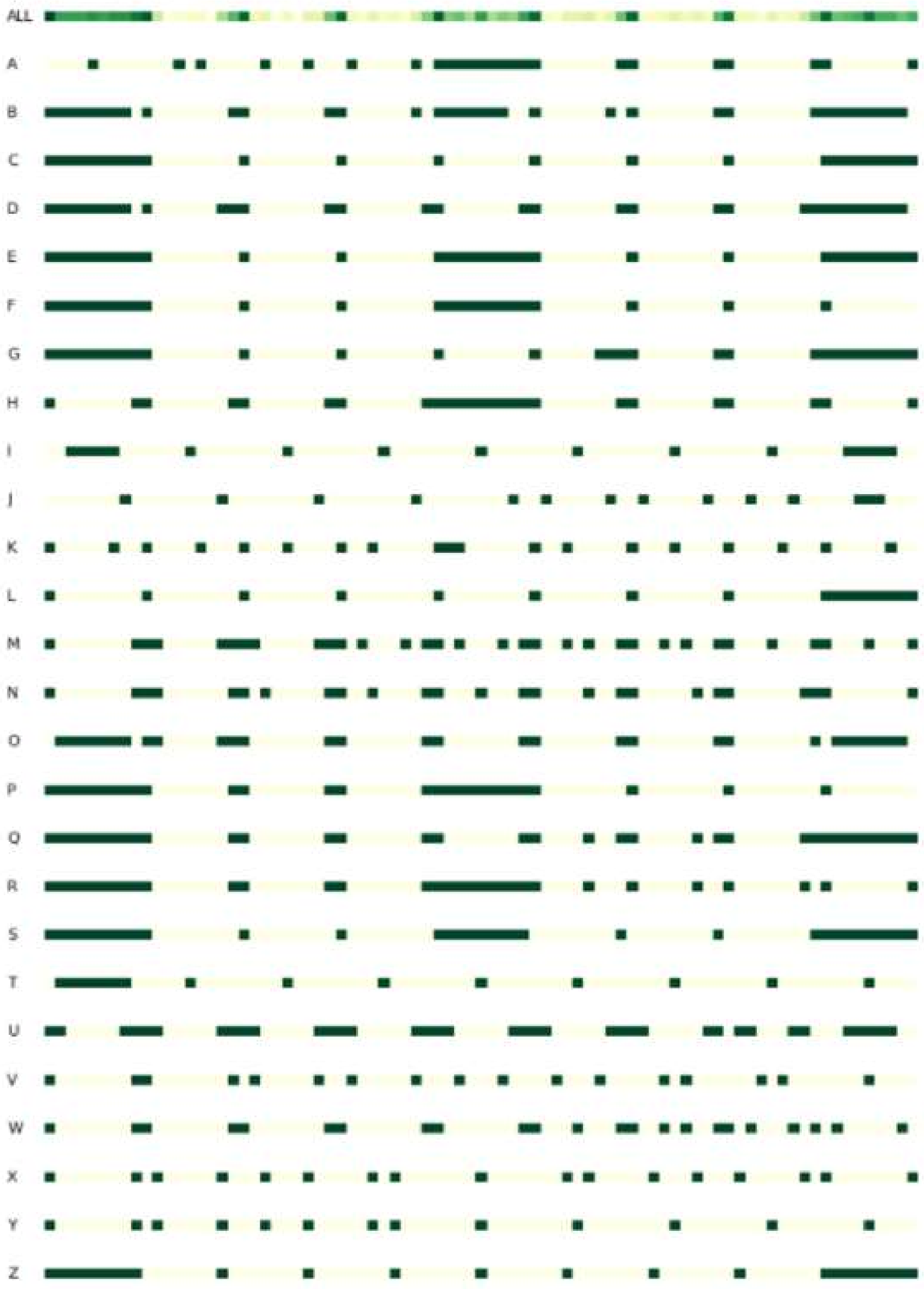
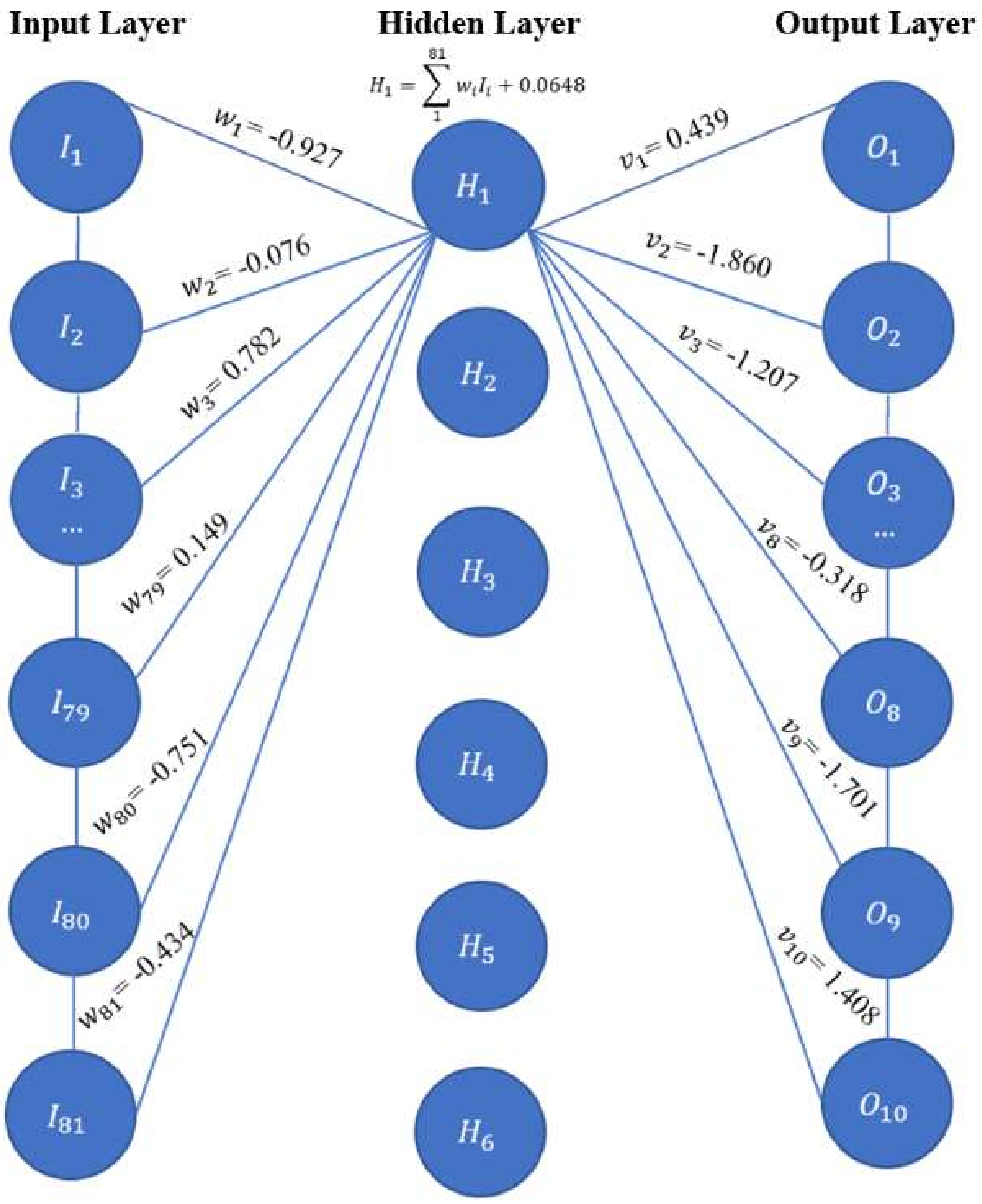


Figure 6. The framework of a neural network with an input layer of size 81, a single hidden layer with 6 hidden nodes, and an output layer of size 10.

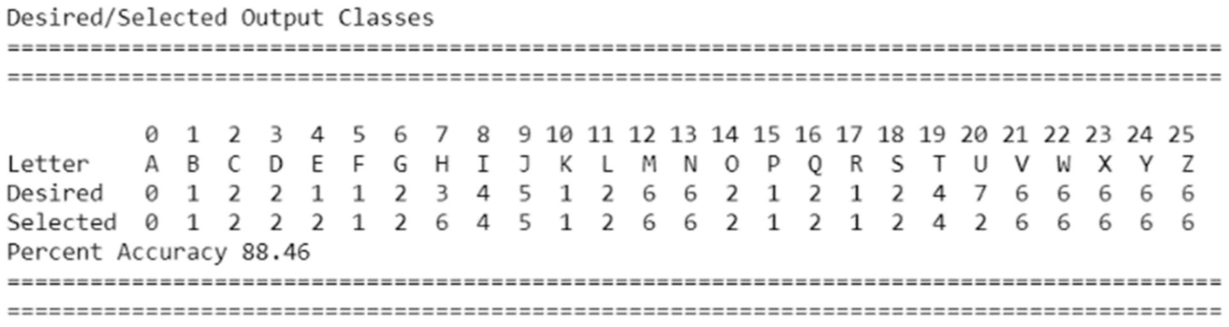


Tables

Table 1. Summary of performance of single-hidden layer classification neural networks with different sizes of hidden layer.

SSE of Each Letter after Training	**Number of Hidden Nodes**					
	6	8	10	15	20	30
A	0.957	1.029	0.811	0.888	1.092	1.388
B	0.044	0.098	0.092	0.241	0.228	1.335
C	0.058	0.023	0.050	0.137	0.108	0.209
D	0.015	0.330	0.064	0.076	0.087	0.019
E	0.045	0.181	0.132	0.319	0.620	1.171
F	0.022	0.031	0.061	0.159	0.135	0.484
G	0.024	0.035	0.012	0.057	0.046	0.009
H	1.267	1.635	0.981	1.150	0.971	1.037
I	0.120	0.250	0.169	0.201	0.270	1.264
J	0.189	1.096	0.889	1.143	0.777	1.248
K	0.034	0.034	0.019	0.098	0.252	1.103
L	0.038	0.071	0.019	0.054	0.079	0.298
M	0.050	0.067	0.022	0.072	0.030	0.436
N	0.051	0.941	0.086	0.077	0.186	0.283
O	0.016	0.053	0.030	0.037	0.137	0.034
P	0.030	0.022	0.050	0.086	0.157	0.623
Q	0.016	0.032	0.170	0.092	0.019	0.042
R	0.042	0.023	0.026	0.294	0.085	0.662
S	0.020	0.027	0.402	0.245	0.058	0.184
T	0.109	0.186	0.183	0.077	0.254	1.297
U	1.091	0.683	0.969	1.133	1.015	0.643
V	0.058	0.022	0.196	0.009	0.026	0.049
W	0.255	0.033	0.193	0.101	0.075	0.144
X	0.021	0.040	0.014	0.005	0.039	0.025
Y	0.025	0.133	0.031	0.020	0.209	0.130
Z	0.040	0.132	0.192	0.425	0.133	1.281
Average SSE	0.343	0.534	0.434	0.533	0.525	1.141
Total SSE	4.637	7.207	5.863	7.196	7.088	15.40

Table 2. The performance summary of a single-hidden layer neural network with 6 hidden nodes in the hidden layer.



```
Desired/Selected Output Classes
========================================================================================
========================================================================================

          0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Letter    A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z
Desired   0  1  2  2  1  1  2  3  4  5  1  2  6  6  2  1  2  1  2  4  7  6  6  6  6  6
Selected  0  1  2  2  2  1  2  6  4  5  1  2  6  6  2  1  2  1  2  4  2  6  6  6  6  6
Percent Accuracy 88.46
========================================================================================
========================================================================================
```