Letters of the Alphabet: How hidden layer values in a single-hidden layer network: represent features within the input data

Ezana N. Beyenne

MSDS 458 Artificial Intelligence and Deep Learning
Dr. A.J. Maren Ph.D.

10/5/2020

Abstract

We will investigate the behavior of hidden layer nodes to gain insight into how generative methods work. By doing so, we will gain insight into how deep neural networks operate. The purpose of this research paper is to explain how hidden layers that exist in a single hidden layer neural network have learned to represent features within the input data using the backpropagation learning method. We will use a neural network with 81 input nodes and attempt to classify the input into one of the 26 output classes representing the letters of the alphabet. The critical focus is not on classification accuracy. Instead, it looks at how the hidden activation nodes identify for each different class of input data and what the hidden nodes respond to. Understanding hidden node activations is essential because deep learning networks use a generative method to determine what features should be identified and represented at each hidden layer, followed by a discriminative approach such as backpropagation to smooth out the connection weights, top-to-bottom, once the various hidden layers have been approximately trained.

Introduction & Problem Statement

This paper's primary goal is to gain practical knowledge in interpreting the impact of hyperparameters and going a step further and understanding how the hidden nodes in a simple neural network learn to represent features within the input. Once we know how hidden nodes work using the backpropagation learning method, we will set ourselves up for understanding how deep learning works.

Deep learning neural networks consist of multiple hidden layers and use an entirely different approach to figure out features that should be first identified and represented at each of the hidden nodes. The deep learning neural network will then use a discriminative method like backpropagation, going from top to bottom, smoothing out the connection weights once the various hidden layers have been approximately trained.

This paper details an experiment where we will use different value settings of the hidden node count hyperparameter to understand and interpret the hidden node activations as features without focusing on training a successful neural network.

Literature Review

Neural networks have usually been considered a black box.[1] With input going into the black box, learning happens in the hidden layers to find a representation of input data and then have some expectations on the output.[1] In understanding how hidden nodes in a single hidden layer network learn to represent features within the input data, and how the number of nodes within this hidden layer can affect generalization, we achieve the following:

1. The neural network becomes a little more transparent
2. We gain a more in-depth insight into how multi-layer neural networks operate.

[1] Jani Turunen, "The Black Box Problem – What It Is and Why You Should Worry about It," Solita, March 23, 2020, https://www.solita.fi/en/blogs/the-black-box-problem-what-it-is-and-why-you-should-worry-about-it/.

Data

The data used in this paper's analysis consists of the 26 letters of the English alphabet. Each letter comprises 81 input nodes, and it corresponds to having a nine by nine grid with the filled-in blocks working together to create an image of the letter. The 81-node input layer is a one-dimensional array (e.g., an 81 by one format). The nine by nine grid is shown so that it is easier to interpret for human eyes. Figure 1 gives an example of the letters A, B, Y, and Z. All the letters display a level of consistency in that they take up the entire nine by nine grid in their representations. For the network to derive features that can be represented, consistency is essential.

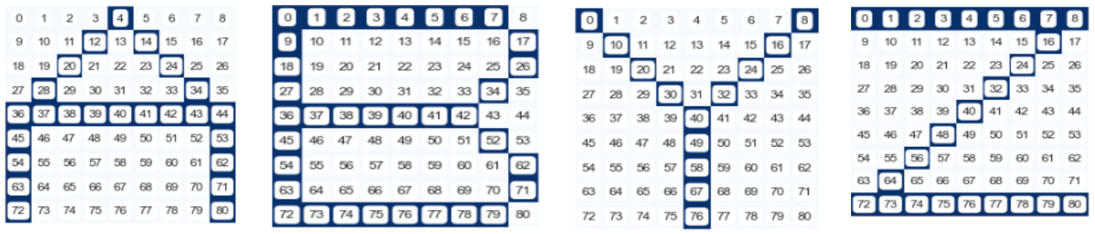


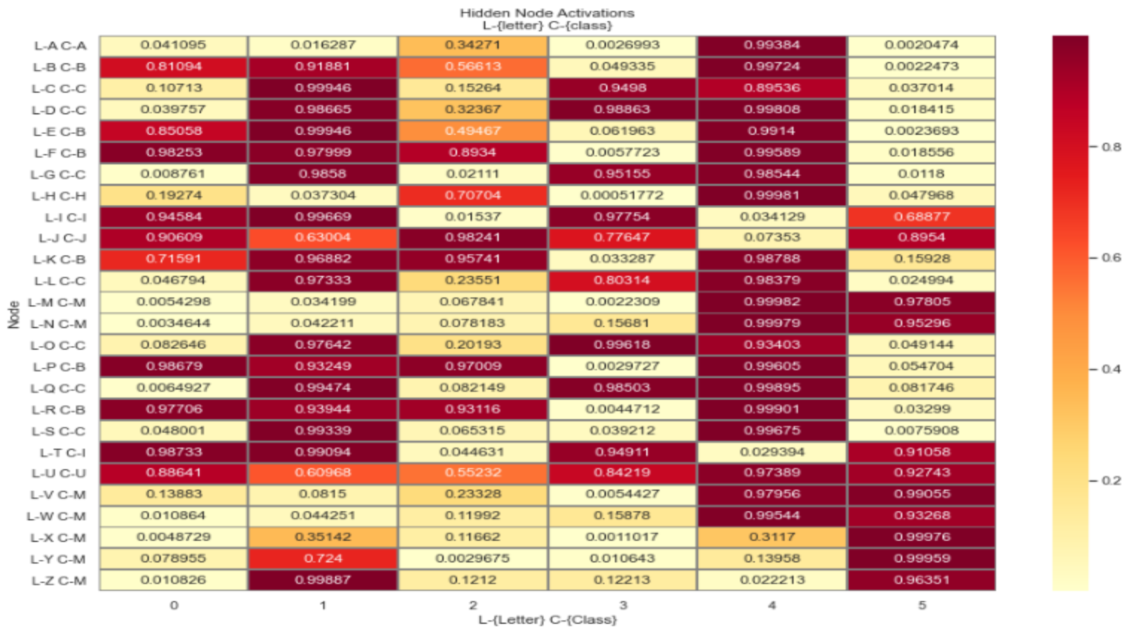
Figure 1. 9 by 9 Letter representations of the letters A, B, Y, and Z

Research Design and Modeling Method(s)

The neural network comprises an input node of size 81, a single fully connected hidden layer with six hidden nodes, and an output layer of size 26 to represent the possible letter outputs. I used a sigmoid transfer function with alpha set to one to calculate the hidden layers' output. For the learning method, I used the backpropagation algorithm to train the neural network with a learning rate (eta) set to 0.5 that iterates over 5,000 epochs. The resulting total sum of squared errors reaches the set epsilon threshold of 0.01. I initially started with ten nodes for the hidden layer, then compared it with an eight-layer node and finally chose six nodes for the hidden layer. It seems that the neural network with six nodes had better generalization.

Results

The first step is to visualize the weighted matrix of the hidden node activations in the form of a heatmap. The hidden nodes perform simple pattern recognition and abstract away different features from the inputs and biases. By looking at the heatmaps, we can determine which combinations of the input data contributed to the hidden node activation values and therefore see which features each node picks. The red areas in the heatmap indicate parts of the matrix with an increased chance of the input causing an activation, whereas the yellow areas indicate a decreased chance. Figure 2 shows a hidden node activation heat map for six hidden nodes and the chances of those nodes being activated for each of the 26 letters of the alphabet.



Node	0	1	2	3	4	5
L-A C-A	0.041095	0.016287	0.34271	0.0026993	0.99384	0.0020474
L-B C-B	0.81094	0.91881	0.56613	0.049335	0.99724	0.0022473
L-C C-C	0.10713	0.99946	0.15264	0.9498	0.89536	0.037014
L-D C-C	0.039757	0.98665	0.32367	0.98863	0.99808	0.018415
L-E C-B	0.85058	0.99946	0.49467	0.061963	0.9914	0.0023693
L-F C-B	0.98253	0.97999	0.8934	0.0057723	0.99589	0.018556
L-G C-C	0.008761	0.9858	0.02111	0.95155	0.98544	0.0118
L-H C-H	0.19274	0.037304	0.70704	0.00051772	0.99981	0.047968
L-I C-I	0.94584	0.99669	0.01537	0.97754	0.034129	0.68877
L-J C-J	0.90609	0.63004	0.98241	0.77647	0.07353	0.8954
L-K C-B	0.71591	0.96882	0.95741	0.033287	0.98788	0.15928
L-L C-C	0.046794	0.97333	0.23551	0.80314	0.98379	0.024994
L-M C-M	0.0054298	0.034199	0.067841	0.0022309	0.99982	0.97805
L-N C-M	0.0034644	0.042211	0.078183	0.15681	0.99979	0.95296
L-O C-C	0.082646	0.97642	0.20193	0.99618	0.93403	0.049144
L-P C-B	0.98679	0.93249	0.97009	0.0029727	0.99605	0.054704
L-Q C-C	0.0064927	0.99474	0.082149	0.98503	0.99895	0.081746
L-R C-B	0.97706	0.93944	0.93116	0.0044712	0.99901	0.03299
L-S C-C	0.048001	0.99339	0.065315	0.039212	0.99675	0.0075908
L-T C-I	0.98733	0.99094	0.044631	0.94911	0.029394	0.91058
L-U C-U	0.88641	0.60968	0.55232	0.84219	0.97389	0.92743
L-V C-M	0.13883	0.0815	0.23328	0.0054427	0.97956	0.99055
L-W C-M	0.010864	0.044251	0.11992	0.15878	0.99544	0.93268
L-X C-M	0.0048729	0.35142	0.11662	0.0011017	0.3117	0.99976
L-Y C-M	0.078955	0.724	0.0029675	0.010643	0.13958	0.99959
L-Z C-M	0.010826	0.99887	0.1212	0.12213	0.022213	0.96351

Figure 2. Hidden node activations for 6 hidden nodes

Table 1 contains a summary of the hidden node activation heatmap from Figure 2. Hopefully, this will show a strong correlation with the nine by nine weighted heatmaps for each hidden node in Figure 3, where we will get a clearer picture of the extracted features

Node	**Letters**
0	Letters with the strongest features are P, F, T, R, I, J and the letters B, E, U show strong features
1	Z, T, S, R, Q, P, O, L, K, I, G, F, E, D, C, B

2	J, K, P, R
3	C, D, G, I, O, Q. U has strong features but not as strong as the others
4	Every letter but I, J, T, X, Y, Z
5	T, U, V, W, X, Y, Z, M, N, J

Table 1. Hidden node activation heatmap showing each node showing strongest features

Analysis and Interpretation

In the nine by nine weighted heatmap for hidden node activations 0 through 5, we get a better representation of the features being extracted for the letters. Table 2 shows the letters that activate each node and the features being extracted for the six hidden nodes.

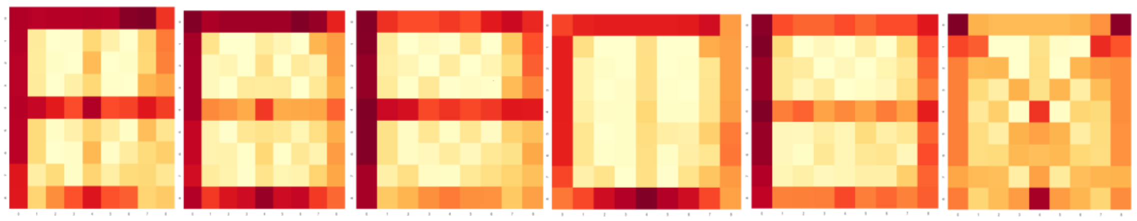


Figure 3. Hidden node activations 0, 1, 2, 3, 4, 5 in a nine by nine pixel grid

Nodes	Letters	Features extracted
0	P, F, E, R, B, I, U, T	Vertical bar along the left and somewhat middle, horizontal bar through the top, middle, somewhat the bottom, right vertical bar through the top half. Fails to turn the diagonals. Strongest for P, F, E.
1	O, P, E, F, D, B, G, L, C, S, Z, I, Q, R, Y	Vertical bar on the left and right, the top, left, bottom, somewhat middle horizontal bar, middle pixel, shows a weak turn on the diagonals for a faded Y. Strong showing for O, E, P, G.
2	P, R,	Vertical bar along the left, horizontal bar through the top, middle and somewhat the bottom, right vertical bar through the top half.
3	C, D, G, O, I, Q, Q, U	Vertical left and somewhat right bars, horizontal top and bottom bars, middle vertical bar although this is not as strong.
4	Every letter but I, T, X, Y	Vertical bars on the left and right, horizontal bars on the top, bottom and middle. Shows a faint diagonal.
5	T, V, W, X, M, N, Z	Top left and right pixels, along with bottom middle pixel, left and right and somewhat middle vertical bars, somewhat top horizontal bar.

Table 2. Analysis of feature extractions from each hidden node in a 6-hidden node network

As we have seen, nodes 1 and 4 are activated strongly by most of the letters. This may indicate too much freedom within the network for the hidden nodes to drop out.

Design and Implementation Considerations

Running the given code and doing some hyperparameter tuning with the number of hidden nodes, the learning rate, number of epochs, and epsilon was time consuming. I think using Azure’s or Google’s cloud services and running the different variations in their virtual machine in parallel would have save considerable time. Additionally, using Keras would have allowed me to take advantage of my laptop’s GPU card. It would have been good to introduce noise to the input values and seen how it would have affected the weighted hidden node activation values. Would the effects of introducing noise have dampened small combinations of pixels, while the extensive combinations of pixels have strongly affected the activation values? I think introducing noise would have forced the hidden nodes to generalize more.

Conclusion

By analyzing the hidden node activations and their relationship to the input and trained weights, we see using heatmaps how each node segments the output space and shows how it represents the letters' features. The nodes generate different segmentations that, when combined, work with the weights going from the hidden layer to the output layer to identify the correct letter ultimately. This paper’s main goal was to interpret the hidden node output and understand how the input data represents features important to classifying the letters of the alphabet. Secondly, by gaining a better understanding, the neural network is no longer a black box. I do not think this would have been possible if we had used datasets like the MNIST dataset. As the neural network structure gets more complicated, it will be interesting to see how the interpretation of hidden node activations will vary and affect generalization.

Works Cited

Turunen, Jani. “The Black Box Problem – What It Is and Why You Should Worry about It.” Solita, March 23, 2020. https://www.solita.fi/en/blogs/the-black-box-problem-what-it-is-and-why-you-should-worry-about-it/