Hidden Node Activation Values:

A functional examination of artificial neural networks

Christopher J. Rico

Northwestern University

Introduction and Problem Statement

Deep Neural Networks (DNN) are a cutting-edge approach to machine learning and artificial intelligence, consisting of a layer of hidden nodes sandwiched between a layer of input and output nodes. As a network is trained on a particular dataset, these hidden nodes perform undirected feature extraction by reacting to particular combinations of input values. In effect, they are doing rudimentary pattern recognition. Teasing apart the input data features that a given network is reacting to remains a difficult task, although one which has great value to those who wish to understand the inner workings of DNNs.

This paper details an experiment wherein several DNNs with various topologies are trained on the basic task of optical character recognition. Activation values of each hidden node are examined to gain better insight into how DNN hidden nodes encode data and extract features.

Dataset

The dataset consists of 45 capital letters. Each letter is encoded as a Python list of 81 binary integers, wherein values of 1 or 0 are used to create a 9x9 bitmap representation of a letter. Each letter is considered to be its own class; therefore, this dataset contains 26 classes.

Some letters also have alternate representations wherein several pixels have been changed to create a slightly different shape. For example, two different representations of the letter “F” (which are categorized as a single class) are as follows:

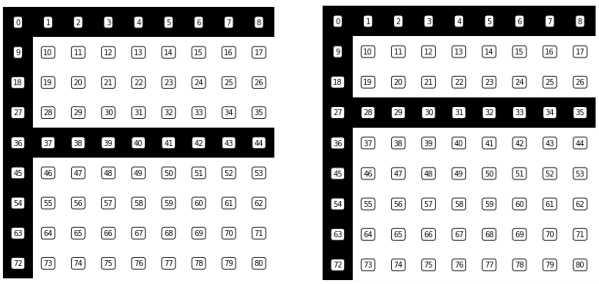

Fig 1 – 9x9 bitmap representation of two alternate forms of letter F training data

A complete list of each letter in the dataset can be found in the appendix.

Research Design and Methods

This experiment was performed using the provided code, executed in Google Colaboratory (a cloud-based Jupyter notebook) running a Python 3 interpreter. Before beginning the experiment, different hyperparameter values were tested to determine which would allow for a convergent network with epsilon ≤ 0.01. Eventually, hyperparameter values were assigned as follows: alpha=1.0; eta=0.2; epsilon=0.01.

This experiment endeavors to explore how DNNs with various numbers of hidden nodes encode information about the training set. Also of interest is the effect of randomly introduced noise during training. Noise introduction is implemented by "bit-flipping" random pixels in the training dataset at a given frequency, which is a way to create variation in the input data. I hypothesize noise introduction will force the network to recognize and generalize about broad patterns of input data instead of recognizing specific pixel combinations.

To explore the effects of both of these variables on hidden node information encoding, a fully crossed 4x4 experimental design will be employed. 4 different networks will be built and trained with either 6 or 10 hidden nodes, and 0 or 15% introduced noise. Qualitative comparisons of hidden node activation values will be made using a method proposed by a classmate, Nick Hallmark. His method of generating 9x9 mappings of the weighted average node activation for each hidden node should allow insight into which pixel combinations are most common in the letters activated by each node. I tried and failed several times to develop a similar visualization to calculate and plot correlations between pixel input values and hidden node activation values. In the end, Nick's visualizations ended up being very helpful.

Results and Analysis

Training Iterations

Number of hidden nodes appeared to have the strongest effect on the training speed, with noise introduction having a smaller, yet noticeable effect on training speed for the network with fewer hidden nodes.

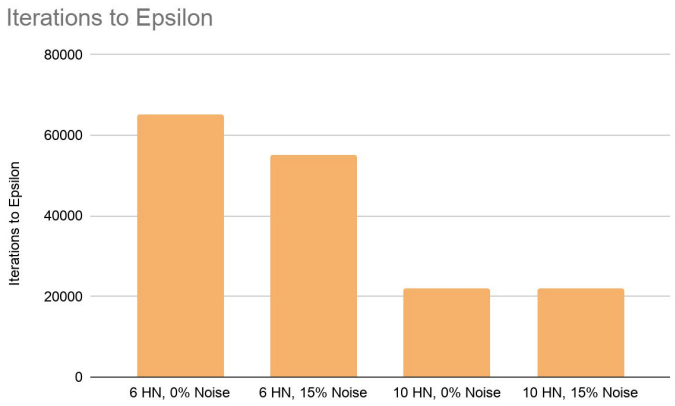

Fig. 2 – Weighted average hidden node activation for 6 hidden nodes, with 0 and 15% noise

Feature Extraction

Taking a look at Nick's 9x9 weighted heatmaps, it becomes clear that certain combinations of "on" or "off" pixels are important to classifying different letters. In other words, the hidden nodes perform rudimentary pattern recognition, and abstract away different features from the inputs. Using these heatmaps, we can determine which combinations of input pixels contribute to hidden node activation values, and gain insight into which features each node reacts to.

6 Hidden Nodes

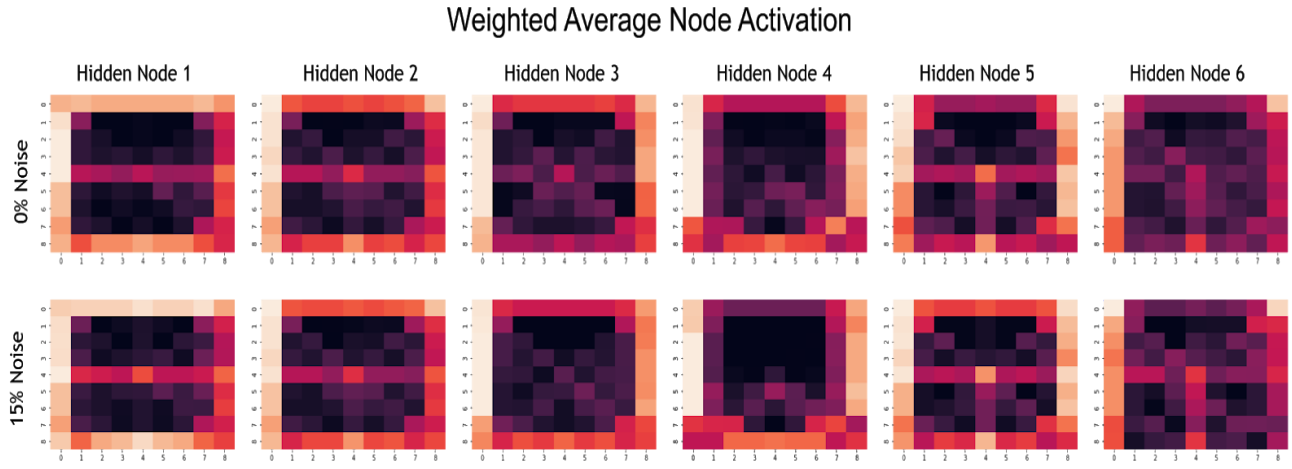

Fig. 3 – Weighted average hidden node activation for 6-HN network, with 0 and 15% noise

Node	Letters	Features Extracted
1	E, G, J, U, W	Bars on top, left, bottom, middle. G, E show these the strongest.
2	All but A, J, V, X, K, Y	Top corner pixels, bar on left, horizontal bar through middle.
3	D, G	Vertical bar along left.
4	F, N, R, Q, U	Vertical bars on sides, diagonal from top left to bottom right (R, N, Q).
5	H, M, N, V, W	Vertical bars on sides, horizontal bar across middle, middle pixel (H, H, M, W) and bottom middle pixel of diagonal (W, M, N.)
6	T, V, X, Y	Top corner pixels (all letters), bottom middle pixel(T, V, Y), middle vertical bar through bottom half (T, Y).

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

6 hidden nodes was the minimum number of hidden nodes to achieve network convergence. That said, this topology lacks a lot of the refinement of networks with a greater number of hidden nodes. It took roughly 60,000 training iterations for both networks with 6 hidden nodes to converge, likely due to the hidden nodes having to make big generalizations to differentiate between classes. The addition of 15% noise appears to dampen the effect of individual pixels on activation values, pushing the activation patterns towards bigger patterns of pixels as opposed to small features.

10 Hidden Nodes

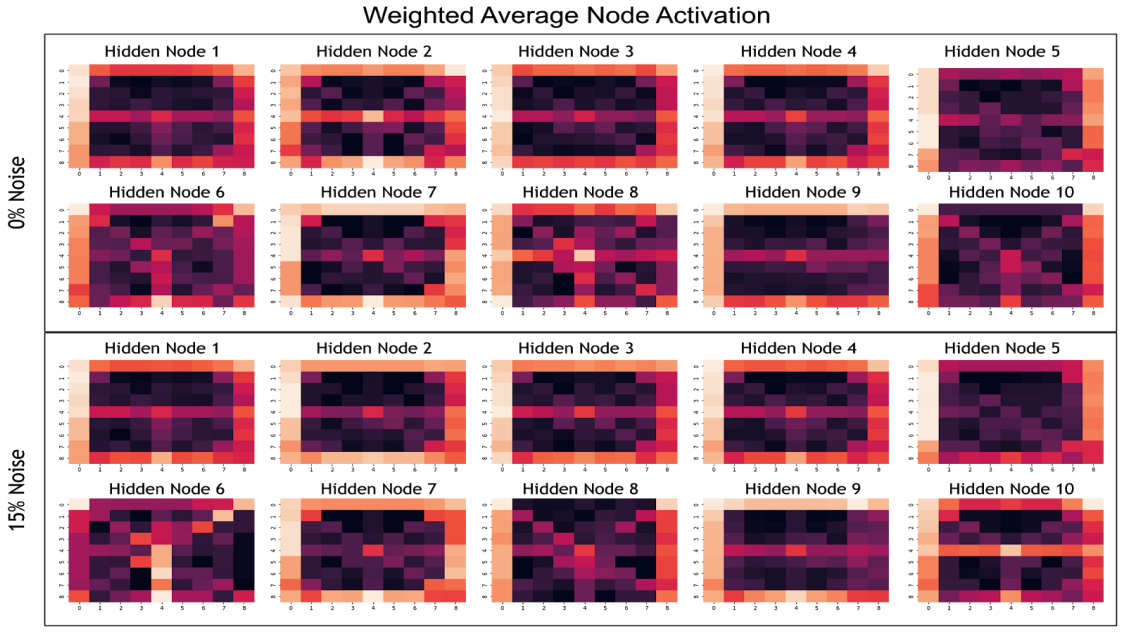

Fig. 4 – Weighted average hidden node activation for 10-HN network, with 0 and 15% noise.

Node	Letters	Features Extracted
1	All but J, U, V, Y	Bar on left, horizontal bar across middle, pixels in top two corners
2	I, J, V, X, Y, Z	Bars on top, bottom (Z), bottom middle pixel (I, J, V, Y), diagonals (X, Y)
3	All but C, E, F, G, K, S	Hard to discern
4	K, V, W, Y	Vertical bar on left (K, W), top corner pixels, middle pixel (K, W, Y), bottom middle pixel (V, Y)
5	All but M, V	Hard to discern.
6	A, B, E H, I M, Q, R, S, T, V, Y, Z	Vertical middle bar in lower portion (T, I, Y), diagonal from top corners to middle (Y, V, Z, M)
7	All but L, Q, W	Hard to discern. Maybe a general O shape?
8	B, C, E, G, L, T, Z	Vertical bars on right and left
9	B, C, D, I, J, L, M, N, O, Q, S, W, Z	Hard to discern. Maybe a general C shape?
10	B, C, D, E, G, I, J, Q, S, U, Z	Horizontal bar across middle (B, E, G, S), pixels in top corners (B, C, D, E, G, Q, S, U, Z)

Table 2. – Analysis of feature extractions from each hidden node in 10-HN network

In this network, the hidden nodes begin to react to smaller, more specific combinations of pixels, as the greater number of hidden nodes gives the network more degrees of freedom. The effects of introduced noise are much clearer in this topology: with 15% noise introduced, large combinations of pixels have a stronger effect on activation values, while smaller combinations are dampened. This is a great example of how introduced noise forces hidden nodes to generalize about large features, instead of overfitting to the training set and "memorizing" small combinations of pixels. That said, there are a few nodes (3, 5, 7, 9) which are activated strongly by every letter. This may indicate that there is too much freedom within the network for hidden nodes to drop out. By training and analyzing these four networks, I was able to analyze how DNN hidden nodes encode training data, and see very clearly the effects that noise introduction has on hidden nodes' ability to perform feature extraction.

Appendix

"Flattened" representation of each letter in the dataset: