Run Away Synapses

Abstract

This article discusses synapse formation in the HTM algorithm, analyses a flaw in the algorithm, and proposes a solution. The flaw is caused by the competitive nature of neuronal activity, which if left unchecked will cause run-away feedback loops. The flaw is a potentially present in all neural systems, and indeed, other comparable algorithms employ similar measures.

David McDougall, 2019

Introduction

Hebbian Learning and Thresholds

Synapses in the cortex use Hebbian learning. Neurons in the cortex use an activation threshold to discriminate between cells which recognize their inputs and cells which do not. Naive models of the cortex incorporated just these two features and observed that Hebbian learning has two characteristic failure modes.

- If the threshold is too high then cells never activate. Inactive cells do not learn, and so they never form additional synapses with which they might overcome the activation threshold. These cells are stuck-off.
- If the threshold is too low then cells activate, learn and form new synapses, which makes them more likely to activate. This can lead to lead to run away activity, where all cells activate at the same time, or a cell activates in response to everything. These cells are stuck-on.

Inhibitory cells in the cortex facilitate a competition between neurons. The competition augments the activation threshold be raising it such that only a small fraction of the strongest neurons activate. The competition can raise the threshold as high or low as it needs to, allowing it to scale to any number of neurons with any number of inputs. Models which include a competition do not suffer from the same characteristic failures of the naive models. In general, Hebbian learning works better with a competition than with only a constant threshold.

However, there is a component of the HTM algorithm which uses Hebbian learning with a constant threshold: the synapses. The permanence value of a synapse is controlled by Hebbian learning. A simple constant threshold discriminates between potentially and actually connected synapses. Synapses suffer from failure modes which are characteristic of Hebbian learning combined with only a constant threshold.

Methods of Analysis

I trained a Spatial Pooler [1] to recognize the handwritten digits 0-9 in the MNIST dataset. All experiments scored between 95% and 96% accuracy, and changes in accuracy as a result of these experiments were insignificant.

I measured the number of connected synapses on each proximal dendrite segment. In this scenario: segments have at most 73 potentially connected synapses, meaning that regardless of any experimental modifications the maximum number of connected synapses is 73.

I measured the activation frequencies of each neuron. Activation frequencies are reported in graphs as a fraction between zero and one. The Spatial Pooler enforces a sparsity of 1% cell activations, which means that the average activation frequency across all cells in the Spatial Pooler will also be 1%.

I calculated the binary entropy of the activations, which is a measure of how much information the cells are transmitting. The entropy is reported as a percent of the theoretical maximum entropy of the system with the sparsity held constant. Higher entropy is better.

Through experimentation and analysis, I hope to better understand how to use each of these measurements to better understand the inner workings of the HTM algorithm.

[1] Cui Y, Ahmad S and Hawkins J (2017) The HTM Spatial Pooler — A Neocortical Algorithm for Online Sparse Distributed Coding. Front. Comput. Neurosci. 11:111. doi:10.3389/fncom.2017.00111

Run Away Hebbian Learning

In this section I demonstrate the failures which are characteristic of Hebbian learning combined with a constant threshold.

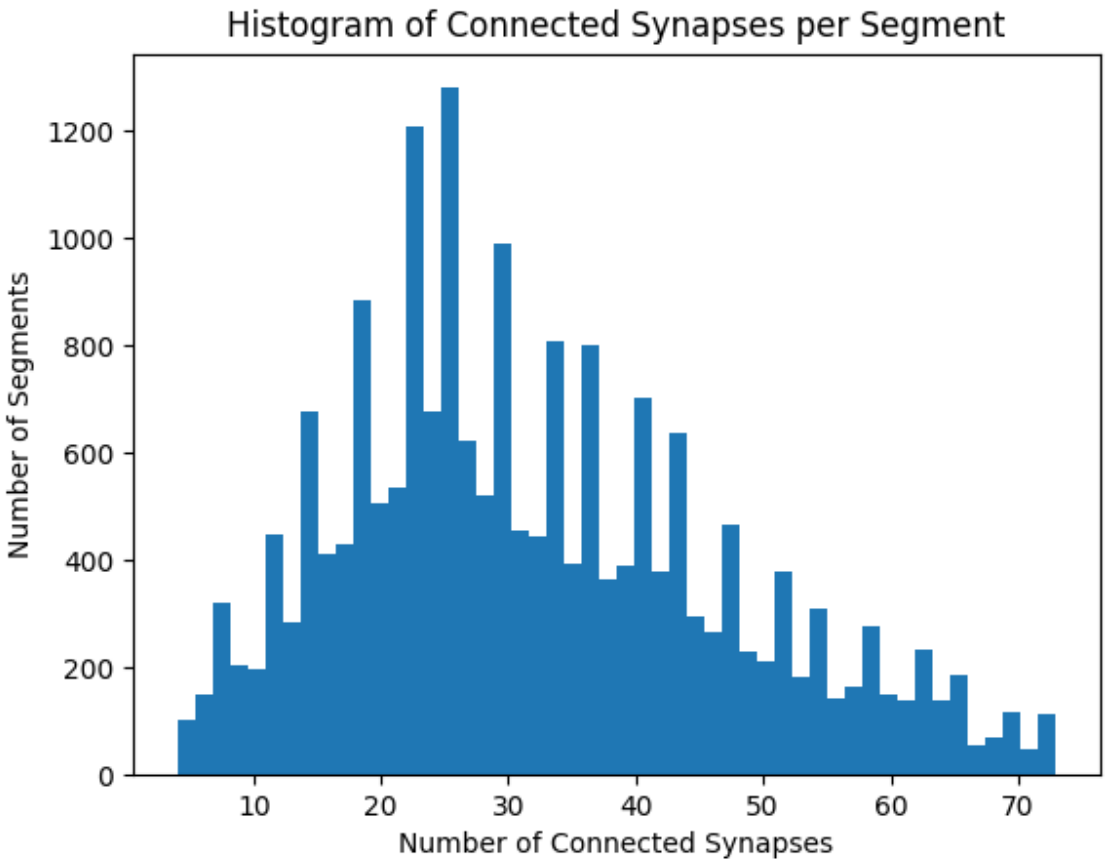


Figure 1: Histogram of the number of connected synapses per segment, in the original Spatial Pooler algorithm. Notice that some segments have very few synapses (4) and that some segments have connected every potential synapse (73).

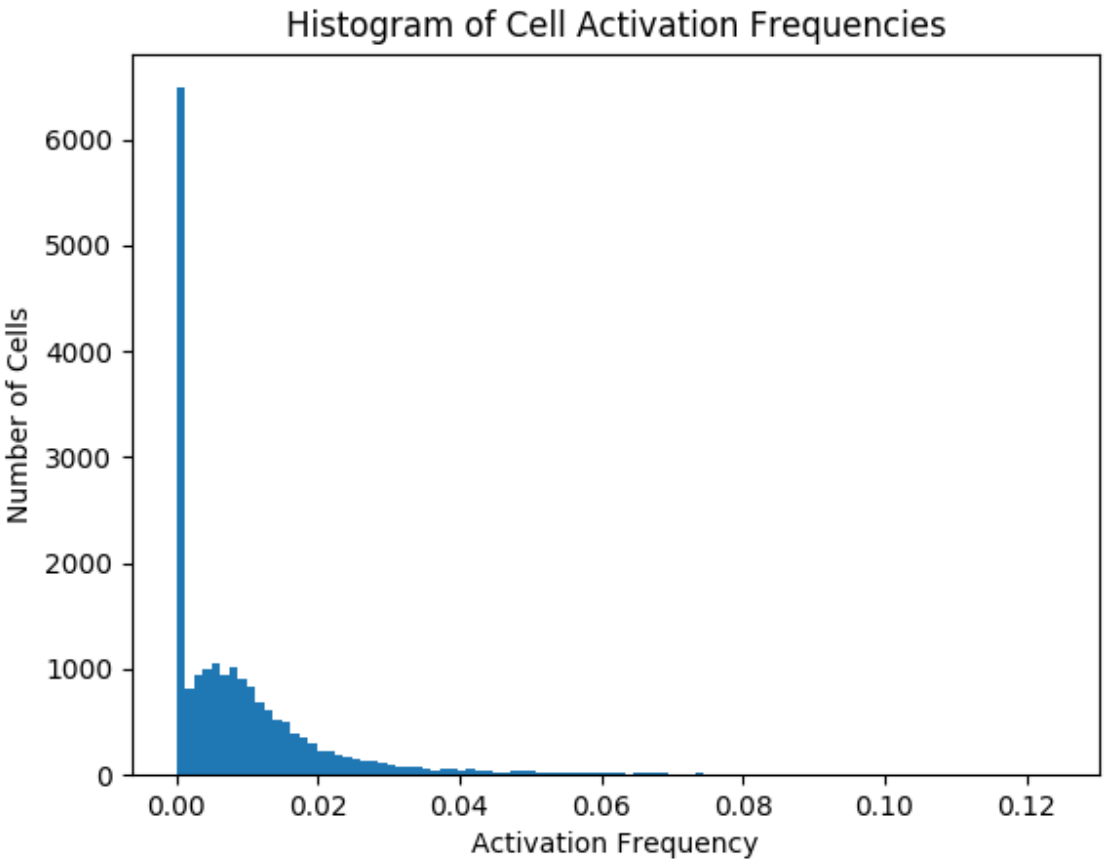


Figure 2: Histogram of cell activation frequencies, in a Spatial Pooler with no constraints on the number of connected synapses per segment. There are cells in almost every bin in this image, although many of them are invisible because there are too few cells in the bin to render on this low resolution image.

Notice that a significant number of segments are underutilized / stuck-off, having close to zero activations. A small minority of segments are overutilized / stuck-on, activating significantly more often than the average activation frequency of 1%.

Mathematical Analysis

In this section I demonstrate the perils of having too many synapses. I used Monte Carlo methods to measure the false positive error rates as a function of the number of connected synapses.

This analysis models dendritic segments as coincidence detectors. A simple threshold determines if a segment receives enough inputs to activate. I generate segments with random connectivity and measure the probability that the segments respond to a random inputs.

Size of Potential Pool	73 cells
Number of Active Inputs	10 cells, (14% sparsity)
Threshold of Detection	5 cells, (50% of active inputs)

Figure 3: Table of Spatial Pooler parameters used for this analysis. These values are typical of a proximal dendritic segment, and were chosen to match or be comparable to the rest of this article. I measured the average sparsity of the MNIST images to be approximately 15 percent.

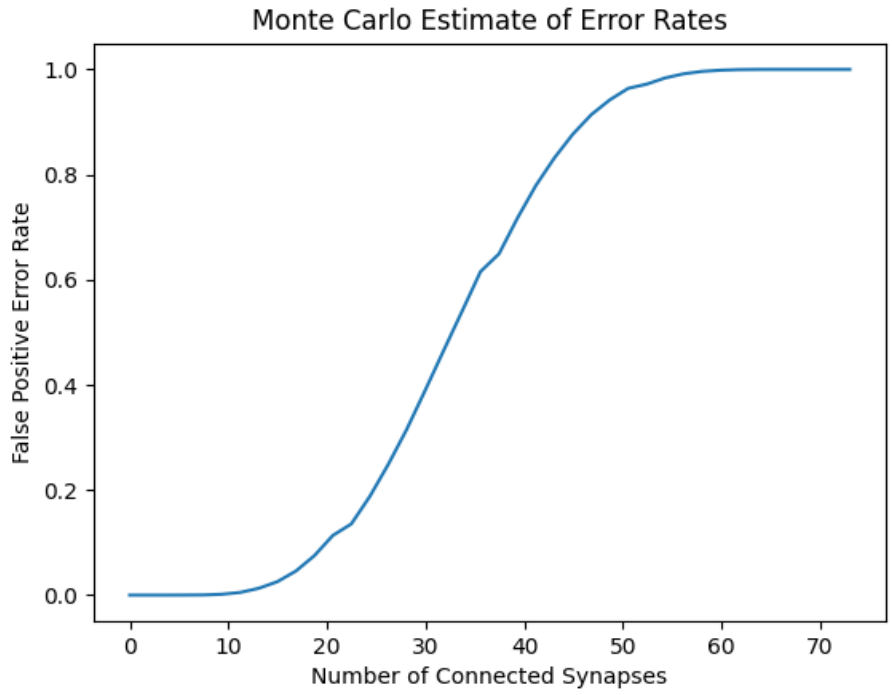


Figure 4: Clearly, too many synapses causes a total breakdown of segment's function.

Results

I developed two techniques for fixing this flaw. First I tried controlling the number of connected synapses on each segments, and then I tried scaling the post-synaptic effect of the connected synapses. The second technique performed best and I recommend using it.

Technique 1: Synapse Competition

This method controls the number of synapses on each proximal segment by adjusting the permanence values of the synapses. I modified a Spatial Pooler such that the number of connected synapses on each proximal segment is constrained. I introduce two new global parameters: minimumSynapsesPerSegment and maximumSynapsesPerSegment. Whenever a segment has too few or too many connected synapses the permanence values of all synapses on the segment are changed uniformly such that the segment has a valid number of connected synapses. The change is added to or subtracted from the permanence and it is calculated to be the smallest possible change which achieves the desired effect.

Results

I implemented the synapse competition by modifying the community fork of Nupic. I experimented with several different parameter sets, shown here:

Minimum Connected Synapses Per Segment	0 (No Limit)	15	20	30
Maximum Connected Synapses Per Segment	73 (No Limit)	50	40	35
Maximum Cell Activation Frequency	12 %	9.3 %	6.8 %	3.1 %
Binary Entropy	87 %	91 %	94 %	96 %

Figure 5: Data table of results. Notice that as the constraints of the synapse competition are tightened, the entropy increases.

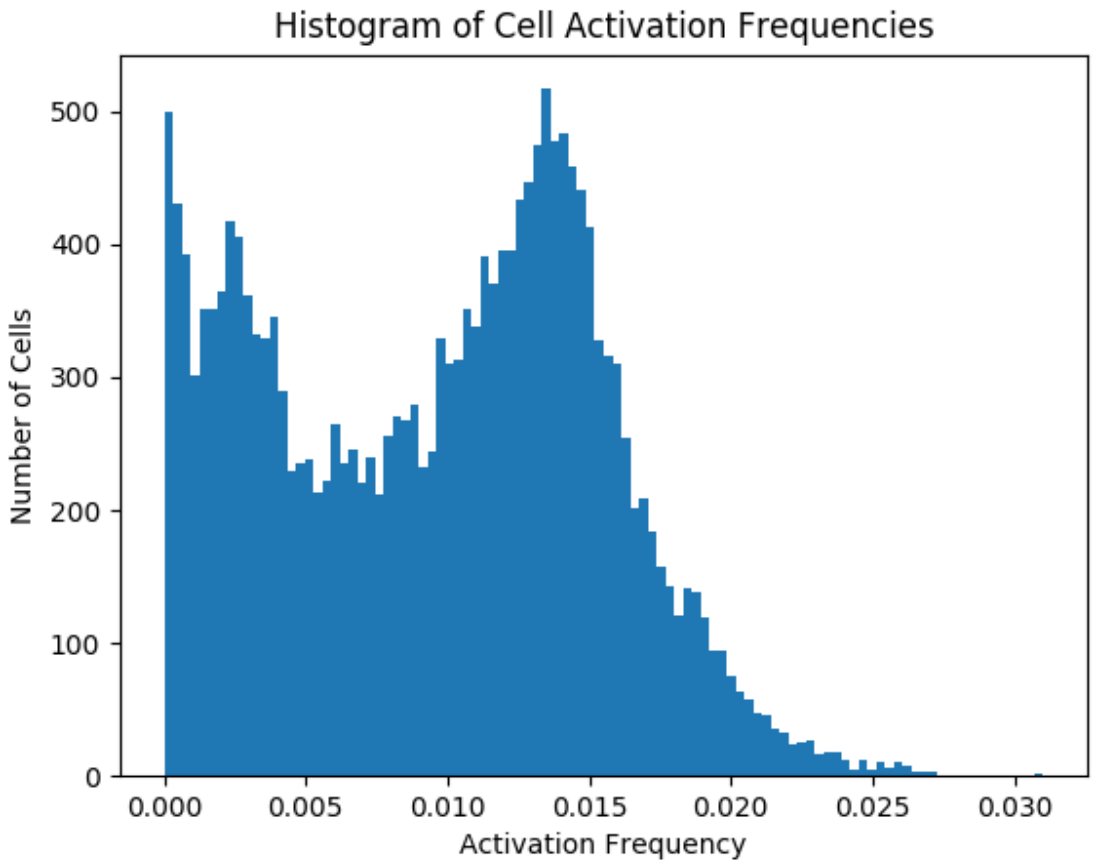


Figure 6: Histogram of cell activation frequencies, in a Spatial Pooler with the constraint that the number of connected synapses per segment is between 30 and 35. Notice that significantly fewer cells are underutilized and that no cells are overutilized.

Analysis of The Distribution of Permanence Values

The permanence values of approximately 50 % of the synapses have saturated to either zero or one. This is a normal aspect of the Spatial Pooler learning process, and will not be discussed further.

Notice the small bump which coincides with the connected threshold. This bump exists on both sides of the threshold. This bump is only present when the number of connected synapses per segment is constrained (evidence not shown). This bump is caused by synapses which lost their competition and are trying to cross the threshold, but which are being held back by the new competition rules. These synapses could be either trying to connect or disconnect, and in both cases they're unable to.

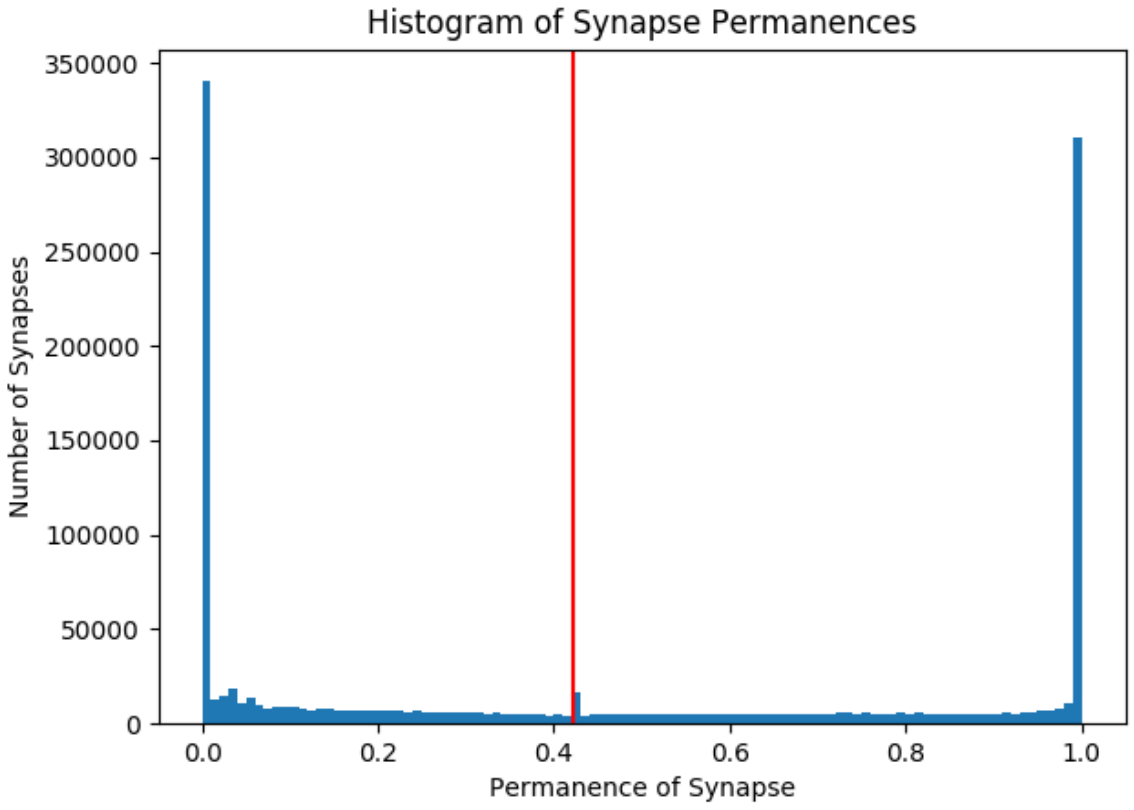


Figure 7: Histogram of synapse permanence values, in a Spatial Pooler with the constraint that the number of connected synapses per segment is between 30 and 35. The red line indicates the connected threshold for synapses. All synapses to the right of the red line are connected. All synapses to the left of the red line are disconnected.

Technique 2: Synapse Scaling

This method controls the net effect of synapses on proximal dendrites, and adjusts it according to the total number of connected synapses. Normally the synapses in the Spatial Pooler have a postsynaptic effect or "weight" of either zero or one, so dendrites are simply counting the number of the active presynapses. Instead, divide by the weight of each synapse by the total number of connected synapses on the postsynaptic dendrite. So instead of comparing cells based on the total number of synaptic inputs, this compares cells based on the fraction of their synaptic inputs which are active. Segment activity is now in measured the range [0, 1]. Cells can now compete to activate on a level playing field even if they have different numbers of connected synapses, whereas before cells with more synapses had an advantage over cells with fewer synapses.

Other similar neural network models also use this technique.

Results:

I implemented synapse scaling by modifying the community fork of Nupic, and using numenta's exponential boosting algorithm. I evaluated the technique on the MNIST dataset. It achieved 95% accuracy, as expected. The maximum activation frequency was 16%. The binary entropy of the cellular activity was 99% of the theoretic maximum!

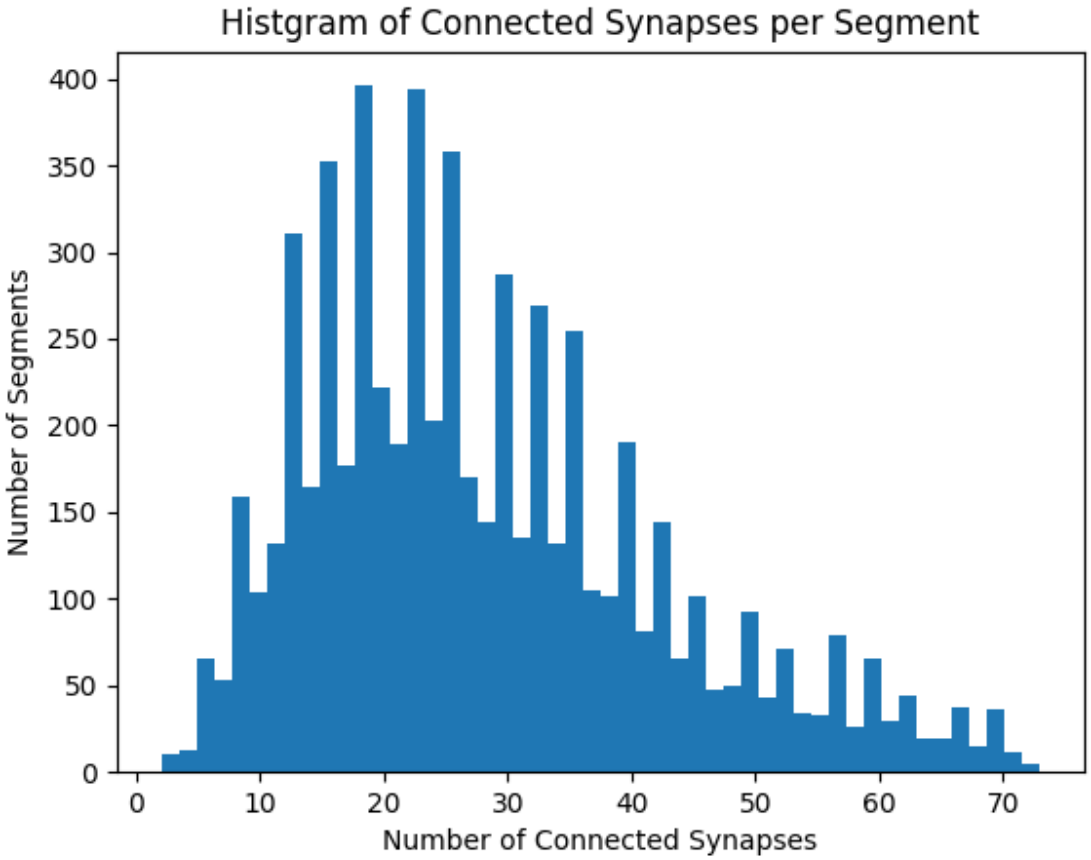


Figure 8: This method does not directly control the number of connected synapses, and as you can see from this histogram the Spatial Pooler still has cells which are connected to every possible input as well as cells with very few synapses.

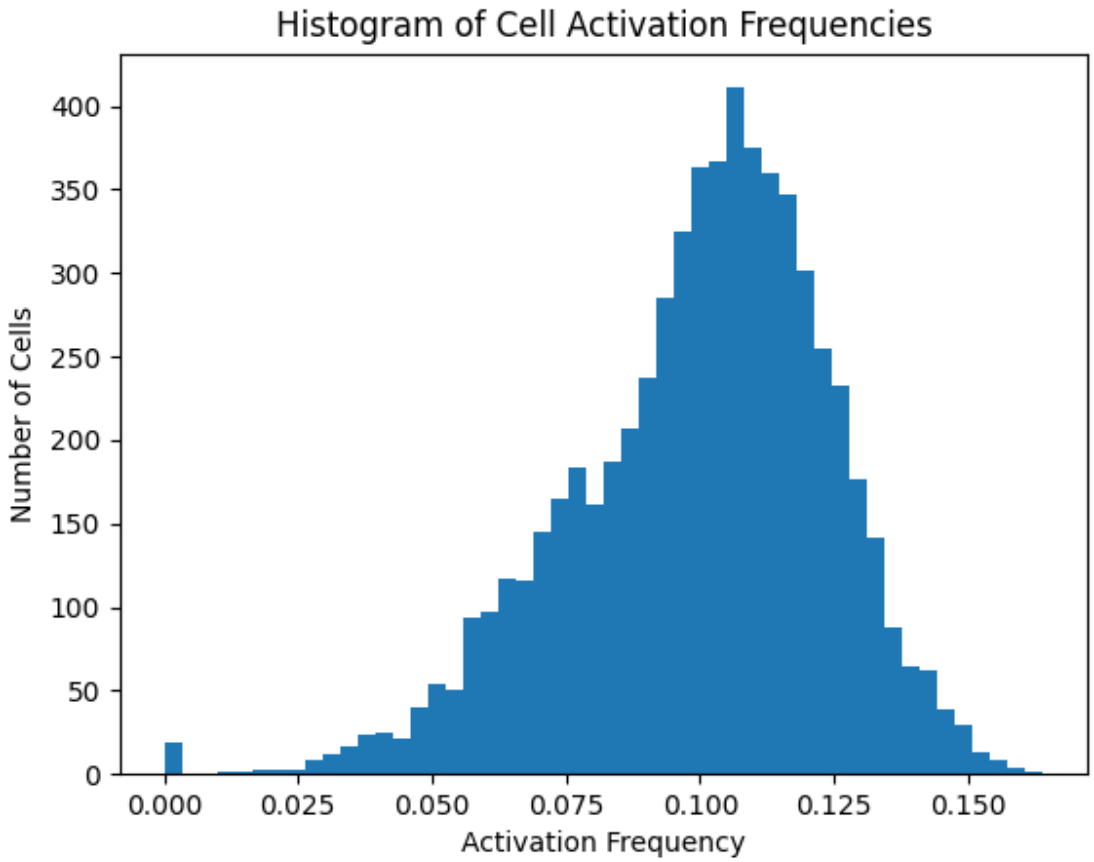


Figure 9: Despite having a wide range in the number of connected inputs, almost all of the cells are equally utilized. Almost no cells are stuck off, and no cells are stuck on.

Appendix A: Other Boosting Methods

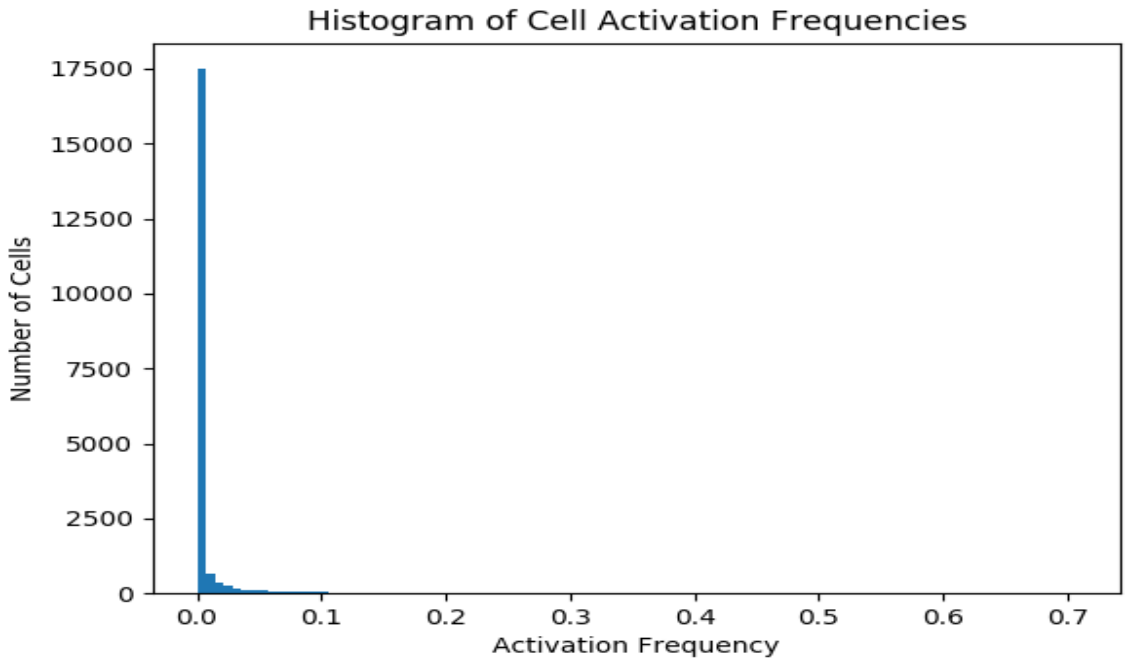
Boosting is a general term for methods which aim to control and normalize the activation frequencies of neurons. Here is a table of relevant findings, followed by a more detailed analysis of each method.

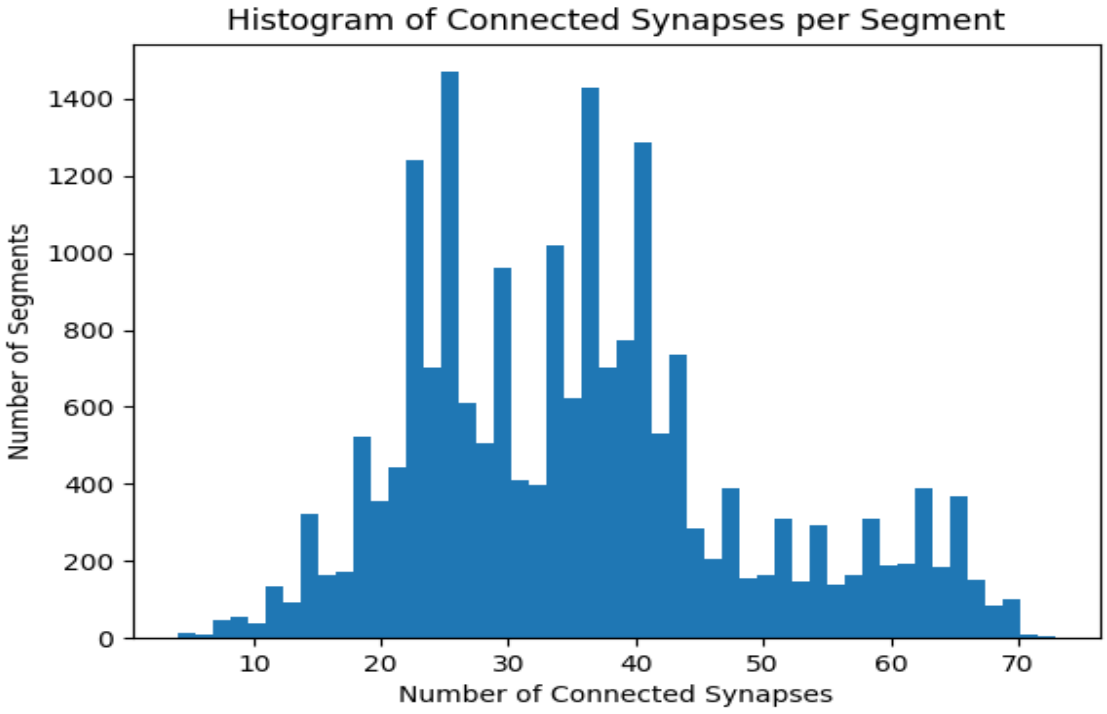
Method of Boosting	Maximum Cell Activation Frequency	Binary Entropy
No Boosting	71 %	51 %
Synapse Competition	67 %	55 %
Exponential Boosting	6 %	87 %
Logarithmic Boosting	12 %	87 %
Synapse Competition with Logarithmic Boosting	3 %	96 %
Synapse Scaling with Exponential Boosting	16 %	99 %

All methods which use an exponential moving average to track the activation frequency use a time scale of 1402.

No Boosting

For this experiment I disabled boosting to observe the extent and severity of the issues which boosting seeks to fix. This scored 93.88 % accuracy, making it the only experiment which scored less than 95 % accuracy.

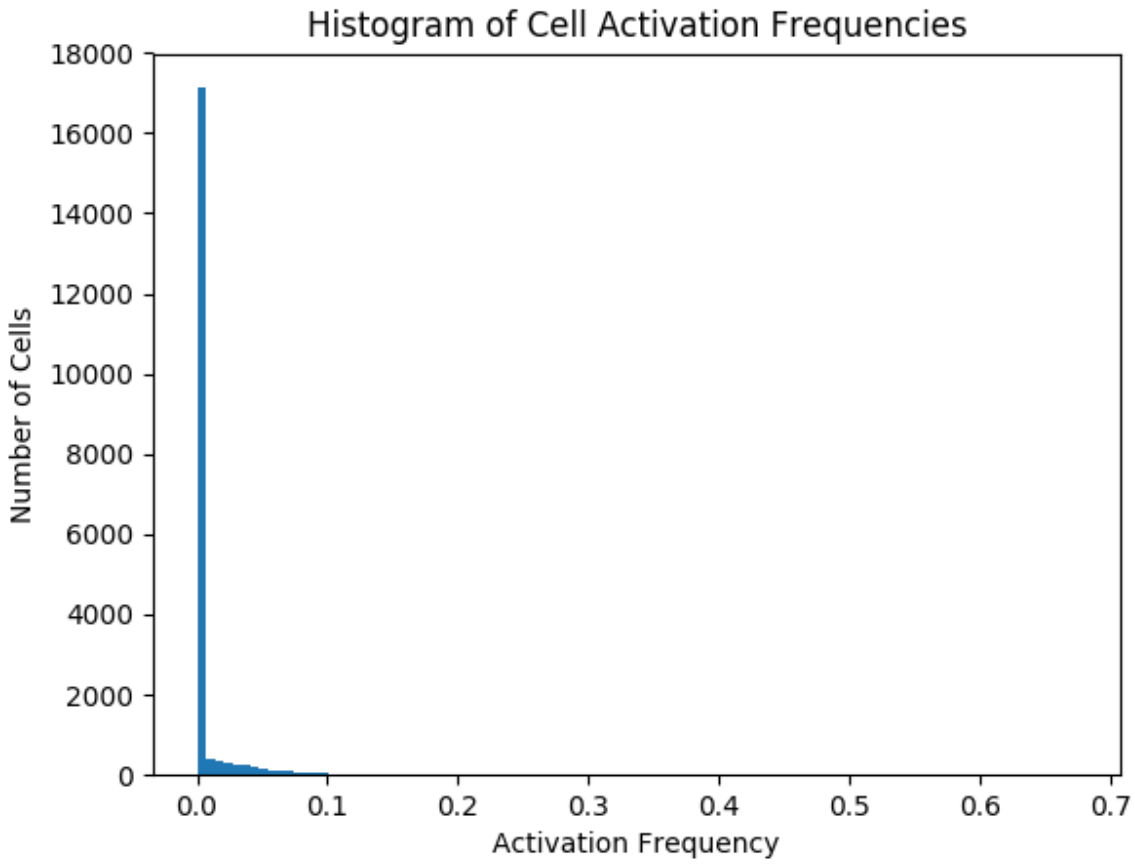




Synapse Competition without Boosting

For this experiment I disabled boosting but enabled the synapse competition, enforcing between 30 and 35 connected synapses per segment. The results show that synapse competition is not a replacement for boosting.

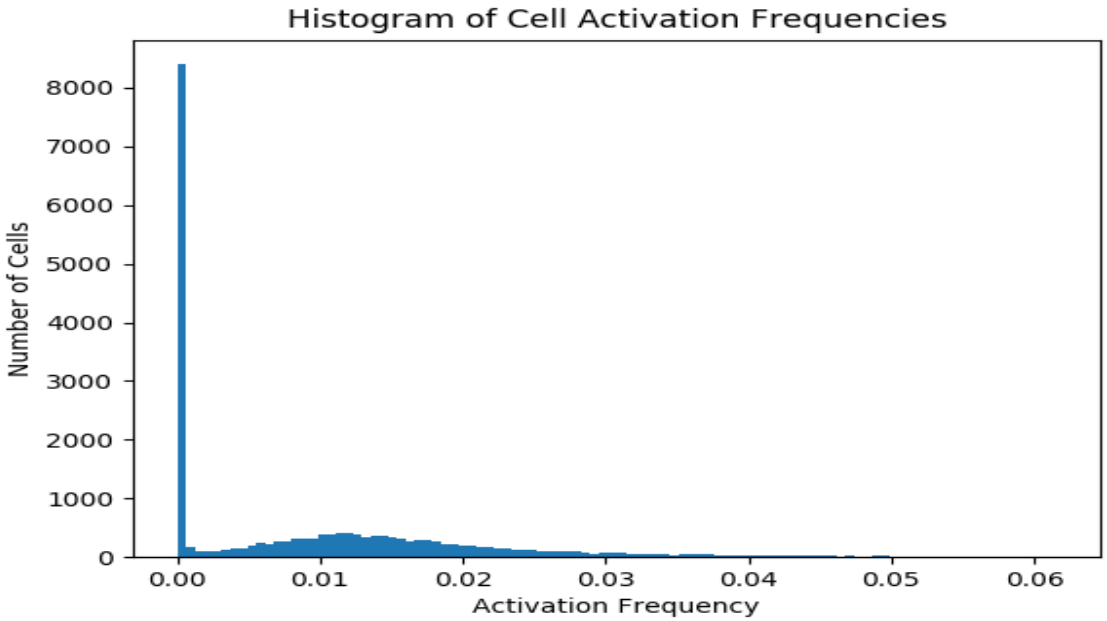
Without boosting the activation frequencies of the cells tend towards the extremes, although it does score above 95% accuracy and it does perform marginally better than with neither boosting nor synapse competition.

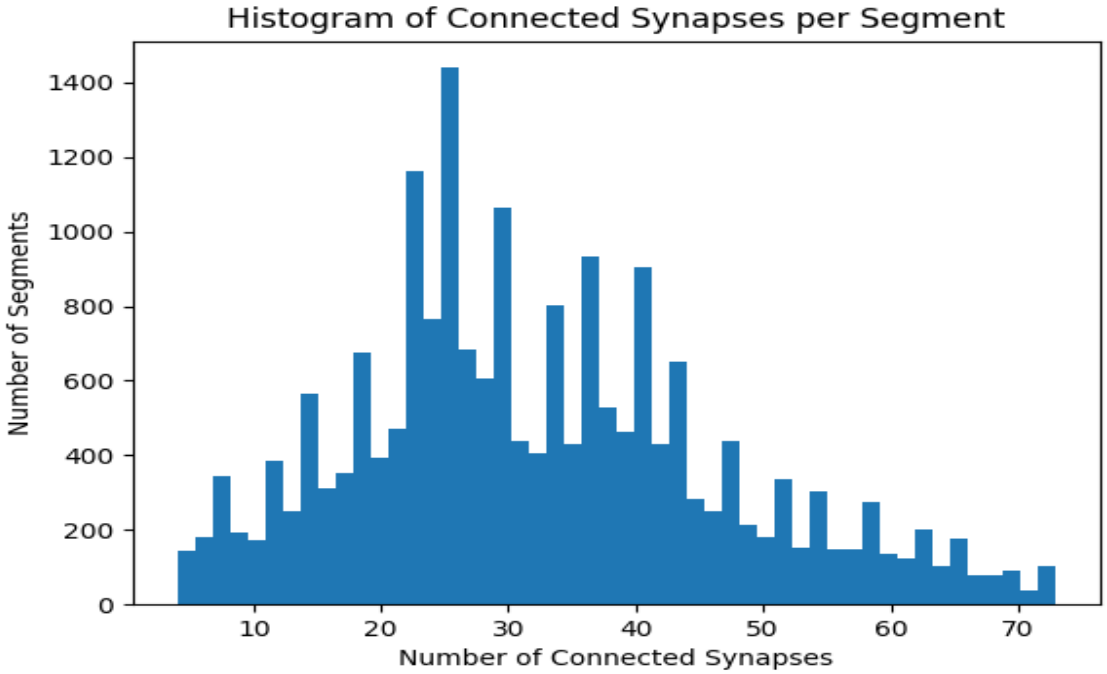


Exponential Boosting

For this experiment I used boosting without a synapse competition. Numenta came up with this boosting function, and it is currently the default for Nupic.

boost-factor = e ^ (boost-strength * (target-sparsity - activation-frequency))
boost-strength = 25.





Logarithmic Boosting

For this experiment I used boosting without a synapse competition. I came up with this boosting function:

boost-factor = log(activation-frequency) / log(target-sparsity)

This function has a zero-crossing at the activation-frequency of 100% and an asymptote to infinity at the activation-frequency of 0%. These properties give it stronger theoretical guarantees than the exponential boosting function. It also has no parameters, which makes it easier to use.

See figure 1 for histogram of connected synapses per segment.
See figure 2 for histogram of cell activation frequencies.

Synapse Competition and Logarithmic Boosting

For this experiment I used both logarithmic boosting as well as the synapse competition.

Boosting alone is unable to control the number of connected synapses. The synapse competition can control this. The fact that the synapse competition improves the performance of the boosting function is evidence of the underlying failures and that they are now fixed.

See figure 6 for histogram of cell activation frequencies.
See figure 7 for histogram of synapse permanence values.

Synapse Scaling and Exponential Boosting

See figure 8 for histogram of connected synapses per segment.
See figure 9 for histogram of cell activation frequencies.

Synapse scaling controls the net effect of the synapses in order to prevents the feedback loops from running away. This technique performs excellently under the metrics defined in this article. However synapse scaling does not attempt to control the number of connected synapses. It begs the question: what are the effects of the number of connected synapses on information processing? How can that be quantified and should it be controlled?