

Notes about the code:

The code builds off of Izhikevich's model with two major modifications. First is the stimulus input system. In addition to the random "thalamic input", there is also stimulation of the neurons in the word by setting its input value I to 30. This however, does not ensure that the neuron will fire. If the neuron has already fired, simple input of that strength will not be enough to build it up to firing potential. However, a neuron at resting state will be guaranteed enough input to overcome firing potential.

The words are stimulated in order up to $\text{words}=5$, the end of the sentence (called *song* in the code), in seconds 1-600, 603 and 606. 1-600 are the first ten minutes of the network, and the training period. 603 is the first presentation of noiseless sentence to determine exactly what fires off a pure sentence. 606 is the first trial of the sentence.

Second 607 sees the activation only of the first word in the sentence to see how far the synfire chain goes in terms of predictor neurons.

The second major addition is the system of predictor neurons. Predictor neurons are simply the neurons that have a higher-than-average chance of fire during training that also matches those that fire during a noiseless sentence. In order to achieve that, at the end of every second, all the firings are recorded in the form of timing-neuron pairs. It is not enough that a neuron has fired within that second, but it must fire at a specific time within the second. Every pair has a counter which is incremented $1/600$, so that if a pair fires 100% consistently through training, it will have a counter with 1.0.

It must be noted that no pair has ever been observed at or near 1.0, even the input neurons. This indicates that the neurons fire at different times given that the input is at the same time. I do not know why this is done, but programmatically, the contributions of v , the neuron's current voltage, and u , the neuron's baseline recovery variable, can shift the timing of the firing.

Pairs with their counter above a certain threshold determined graphically. The reason that I used a predetermined threshold instead of a statistical procedure here is that because of the inclusion of the input neurons, the mean calculation would be extremely skewed. While removal of those values from the stability matrix would give me a better data set to work with, because of the timing issues, those values are often spread over 3 or 4 pairs. Other more sophisticated measures were considered, but implementation time and run time increases were deemed not worth it for this investigation.

The pairs that made it above the threshold were recorded and matched with pairs that fired during noiseless sentence presentation. Since the input pairs were not removed in either and we have no interest in them as predictor neurons, we need to remove them from the list of predictor neurons. This entire process was done using a points system. The pairs that made the threshold gained one point; the pairs that fired during the sentence gained one point; the pairs that are part of the input lost two points. By the end of the process, pairs that had two points left were our predictor neurons.

After the predictor neurons were recorded, the trials using single word and full sentences were done, comparing the number of predictor neuron hits in either circumstance. I should note that the trial presented in the paper was unique in that I picked the timings in such a way that the sentence activated less predictor neurons than just the word. However, it had the nice illustrative last predictor neuron and

the last predictor neuron being off mark by just a bit. In most circumstances, the sentence does activate more. I thought I had made note of that in the paper, but upon looking at it again, it appears I made a mistake and did not include that note. This also bears out with the trials presented with noise. One run had 20 predictor neurons, 15 active for sentence without noise, 5 active for word without noise. With noise, the sentence had 5 active, the word had 0 active. Examining the data revealed that that run also had a predictor neuron after presentations, and that during word with noise, the predictor was not active around the time it was expected to, but during sentence with noise, the predictor was active within 1ms. However, I had foolishly lost that data.

One last programmatic thing to note is that the random number algorithm used in the code is not truly random, in that multiple runs of the same program will always choose the same neurons. I had not even suspected that until too late, when inspecting the neurons led me to observe that the same neurons keep cropping up time after time when they were supposed to be random. I tried to hunt down where in my code this could have happened with no result and much frustration. Now this may not have any effects within the context of a single simulation, but it does have ramifications across simulations. It may be that we are lucky and the neurons chosen are just the right ones for this effect, or that we are unlucky and neurons chosen are just the wrong ones. However, that the effect was observed indicates the former to be the more likely.

For the future, one of the most important things to do is to find a mechanism of determining Predictor Neurons that have relative timing-independence within the range of 1-2 ms, perhaps. Just by looking at the same neuron, but a millisecond or two before or after, the Predictor Neurons would perform a lot better, given the current inexact nature of neuron firing timings. Another thing that would be beneficial to implement would be the ability to serialize and deserialize the variables so that one can test the network without having to re-run the simulation.