Simulation of Neural Network on Internet

Specialize in Memory/Sensory forming

Chao-Hsuan Shen, Kevin Widjaja, Harshini Chandrashekar

Acknowledgements

Special thanks to professor Ming-Hwa Wang. And thanks to our parents for giving birth to us.

Table of contents

List of tables and figures

Abstract

What motivate us is not to solve an existing engineering problem but a quest of a scientific question: How does our complex neural activities give rise to equally complex human behaviors. This quest drives us to build a software model to test and experiment our many hypotheses about “how does it happen?”

We propose that during the memory forming process, the firing pattern correspondent to its stimulus sensory input would converge and getting stable. This stabled firing pattern is directly relevant to memory.

During this project we use Scala programing language and Akka library to model neural network. The simulation result shows it is possible to have a stable firing pattern per different sensory stimuli even the neural network is dynamically changing during the process.

We believe this research provide some insights about how memory could be formed and distributed in dynamically changing neural network. The result could inspire new generation’s computer scientists that internet could be used not only as a mechanism to transfer data and messages, but the connection itself could be used to computed and store data.

1. Introduction

1.1. What is the question ?

We are trying to know how the neural network works. The human brain is structurally and functionally organized into complex neural networks. The functionalities of the brain extend from learning to teaching. In our Project we are focusing on the dynamic connection of the neurons. How the neural system work together to produce integrated behaviours.

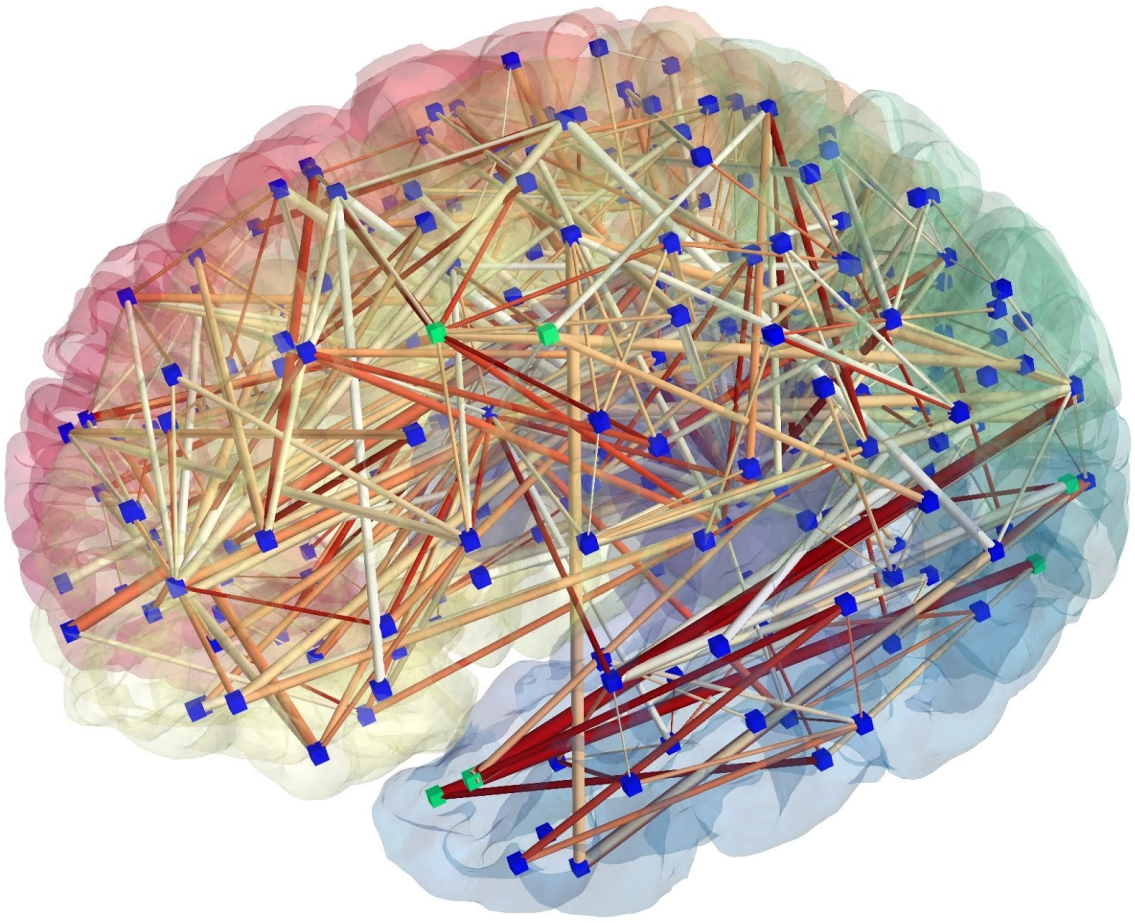


Figure 1.1: Dynamic view of the neuron Connections.

1.2. Why this is a project related the this class

Neural network simulators are software applications that are used to simulate the behaviour of artificial or biological neural networks. Simulators usually provide the visualisation of

the process. In computer science we have networks on which all our computers are connected. In neural networks neurons are interconnected and the information flow through Synapse.

1.3. Why other approach is no good

In the study of neural networks, simulation is the only option for the betterment of solution. But, Few of the scientific environments uses component based simulation, where the neural network is constructed by connecting adaptive filter components in a pipe filter flow. this allows a combination of adaptive and non- adaptive components to work together but main drawback of this paradigm is, component- based environments is more complex than simulators. They require more learning to fully operate and are more complicated to develop.

Therefore Simulation of neurons would provide better understanding of the neural network behaviour.

IBM team unveiled the basic building block of a brain inspired chip architecture based on a scalable, interconnected, configurable network of "**neurosynaptic cores**" that brought memory, processors and communication into close proximity.

1.4. Why you think your approach is better

The companies like IBM, SPAUN and Blue Brian are working on the the study of the neurons connections and trying to model the Human Brain on a silicon chip. IBM's project of SYNAPSE which is building of 'Neurosynaptic Chip' system with ten billion neurons and hundred million synapses. They are building a model, inspired by the brain functions and efficiency.

Simulation of neural network gives an analysis and visualization of the neurons.

1.5. Scope of investigation

Neural networks have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including computer vision and speech recognition. For example, in a neural network for handwriting recognition, a set of inputs neurons may be activated by the pixels of an input image representing a letter or digit. The activations of these neurons are then passed on, weighted and transformed by some function determined by the network's designer, to other neurons, etc. until finally an output neuron is activated that determines which character was read.

2. Theoretical bases and literature review

2.1. Theoretical background of the problem

There are lot of research trying to understand of how human brain works. It revolved around our primary functional unit of nervous system, “Neuron.” Neuron is an excitable cell that processes and transmits information through electrical and chemical signals. These signal between neurons occur via synapses which connects to other neurons. Neurons can connect to each other to form neural networks. It has structures as shown in **Figure 2.1**

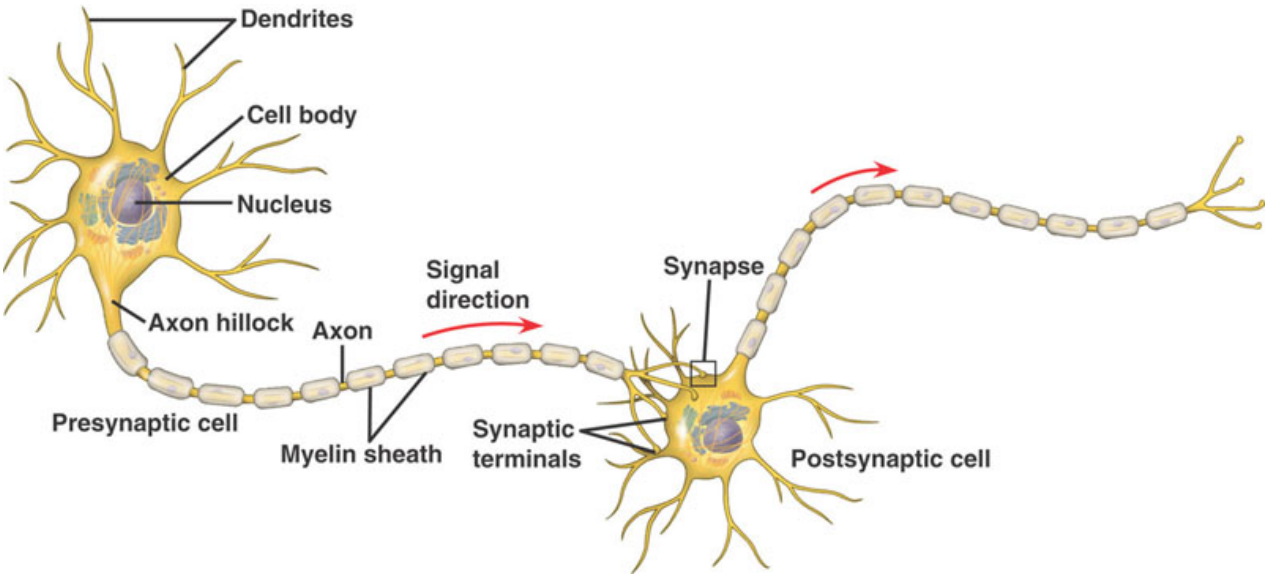


Figure 2.1: Two neurons connected to each other.

Then we have a connectome which basically the comprehensive map of neural connections in our brain. In other word, it is sort of ‘wiring diagram’ that include all neural connection within an organism’s nervous system.

As of today, It still too hard for us to understand human’s brain; some people said simulating brain is “too hard for science.” Even then, we try to make our best effort by decomposing it to smaller parts and use different entry points and perspective to understand it. Projects such as Spaun, BlueBrain, and SyNAPSE are good examples of it. In this paper we try to assess “how” complex neural activities give rise to also complex human behaviors.

2.2. Related research to solve the problem

2.2.1. Spaun

Spaun created by Chris Eliasmith and colleagues at the Center for Theoretical Neuroscience (CTN) at University of Waterloo in Canada, as of December 2012; the world's largest functional brain model. It is a simulated brain that contains 2.5 millions Neurons which is far fewer than the 100 billions in the human brain; Spaun able to recognize lists of numbers, do simple arithmetic, and solve reasoning problems. Spaun is biologically realistic in its simulation of spiking neurons and and neurotransmitters. As a result, it reproduces many quirks of human behaviour, such as tendency to remember items at the start and end of a list better than those in the middle.

2.2.2. BlueBrain

BlueBrain attempt to reverse engineer the human brain and to recreate it in molecular level computer simulation. Founded and led by Henry Markram, the goals of the project are to gain a complete understanding of the brain and to enable better and faster development of brain disease treatments. It involves studying living brain tissue and collecting data about many different neuron types. The collected data later used to construct biologically realistic models of neurons and networks of neurons in cerebral cortex; The simulation are carried out on a Blue Gene supercomputer built by IBM with software based around Michael Hines's NEURON, together with other custom-built component.

2.2.3. SyNAPSE

SyNAPSE (Systems of Neuromorphic Adaptive Plastic Scalable Electronics) project by IBM researchers led by Dharmendra S. Modha develop electronic neuromorphic machine technology that scales to biological levels. In other words, it attempts to build a

new kind of computer with similar form and function to the mammalian brain; this technology would be used to build robot whose intelligence matches that of mice and cats.

2.3. Advantage/disadvantage of those research

I will say that there are no specific advantage or disadvantage for each one of them since we have a very complex problems. It's inevitable that we try to focus to certain parts we interested in and make assumption of certain parts of it. For example, we have Spaun projects that tries to be as biologically realistic as possible in its simulation of spiking neurons and neurotransmitters. In contrast with BlueBrain that attempts to reverse engineer the human brain and recreate it at the cellular level(Structural). And also SyNAPSE which produces brain-inspired chips to simulate brain cognitive ability. All of these researches ignored connectome, which is a comprehensive map of neural connections in the brain.

2.4. Your solution to solve this problem

We are trying to run simulation of neurons in the internet by constructing a model of neural connectivities that considers some of the dynamic nature of it such as reweighting, reconnecting to test a more fundamental hypothesis about memory forming.

2.5. Where your solution different from others

We put the focus on the dynamics of this neural connectivity(Connectome) preceding complex human behaviour. Most of related research presented do not include this idea of dynamic connectivity(example: reconnection).

2.6. Why your solution is better

We try to make it more accessible for people with regular computers with decent computing power to run the simulation instead of the need to use supercomputers. Furthermore, we able to test the process of how to maintain the stability of certain memory in a dynamically changing environment.

3. Hypothesis and Goals

Build upon the theoretical foundations above, we try to model a dynamically changing neural network. In the simulation, the network takes a predefined sensory input, then it fires according to voting system with corresponding weights. During the firing cascading through the neural network, each neurons within firing pattern will change its internal states. The aggregation of all neurons' internal states would be the overall configuration of this neural graph.

We propose if we configure the initial state of the neural network right, by repeatedly firing the same sensory inputs, the corresponding firing pattern will stabilize even though overall wirings of the neural network are changing.

And our goal is to design a model to test this hypothesis.

4. Methodology

4.1. How to generate/collect input data

In this project, the input data would be the sensory stimulus. The way we approach stimulus is to purposefully define them. Here are two major reasons:

- **Ignore unnecessary details**
- **Focus on memory forming and graph dynamics**

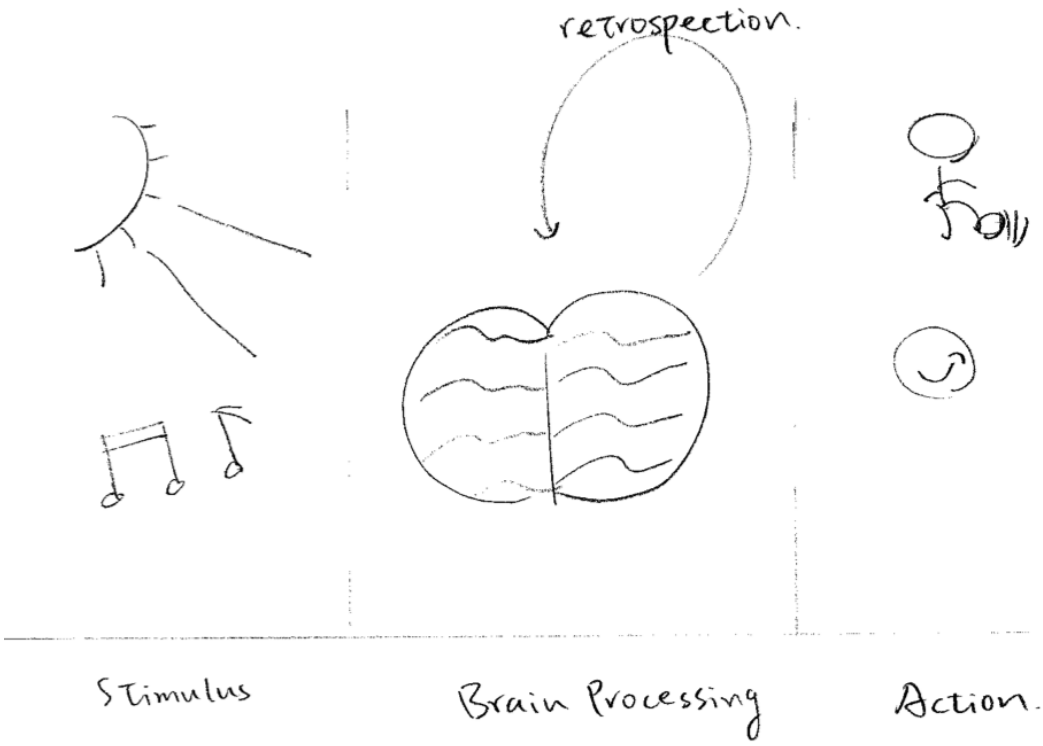


Figure 4.1

In this project, we want to focus on the pursuit of knowing how neural activities could possibly relate to memory forming. So by predefining sensory inputs, we purposefully ignore details and complexities about how human receive sensations.

4.2. Algorithm design

- **Initial state of the graph**
    - Neurons are randomly connected
    - Initial weight of each synapse == 1
    - predefined threshold per neuron
- **Voting firing mechanism**

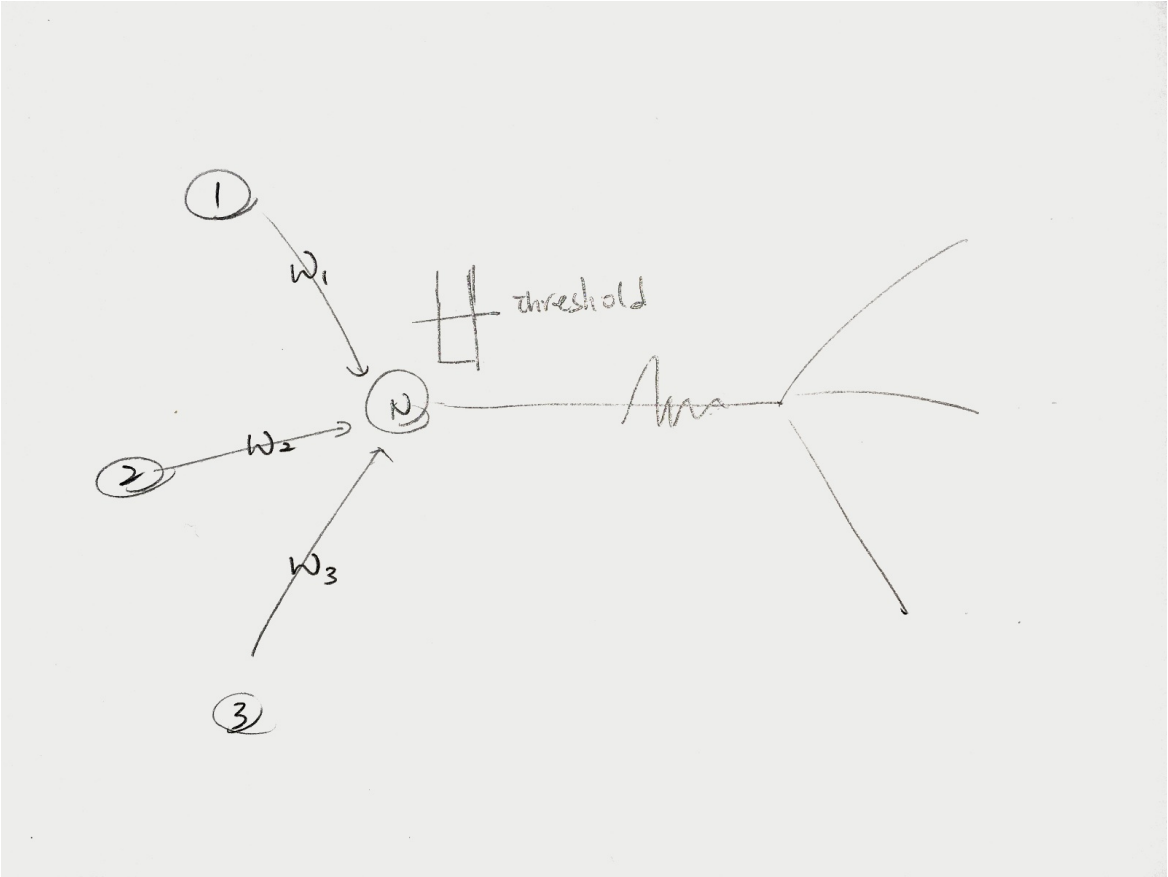


Figure 4.2

- A neuron have to make a decision: **“To fire, or not to fire, that is the question”** Neuroscientist figure out that neurons use a **weighted voting system** to make the decision.
- As the picture above, neuron N is connected to neuron 1, 2, 3, each one of them has its weight to the connection with N. Assume each one of $N_{1, 2, 3}$ fires to N.

If $W_1 + W_2 + W_3$ >= threshold

then N fire

- **Reweight mechanism**
  - In the initial state, the weight for each connection is 1. In the process of neurons firing with each other, the weight of connections would change under one principle:

    “ If A neuron fires to B and then B fires, the weight of A→ B will increase. The contrary condition will decrease the weight.

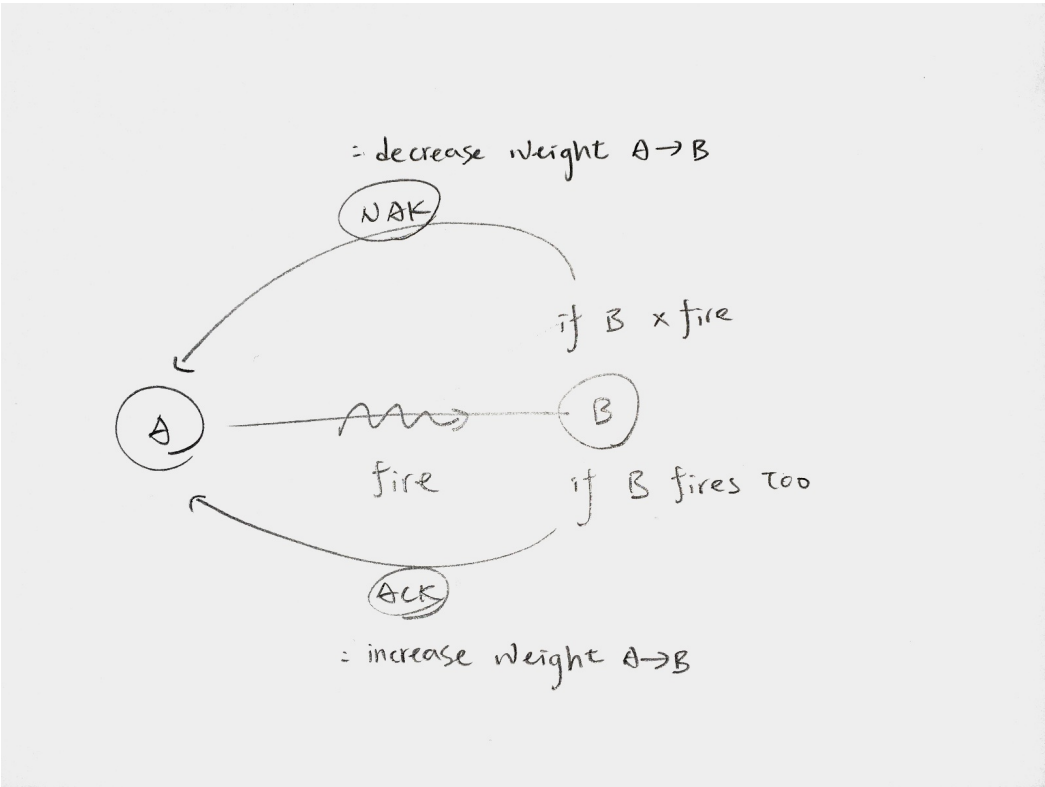


Figure 4.3

- **Reconnect mechanism**
  - For example, neuron A has it’s linked list of connected neighbors. And those neighbor neurons with 0 weight means they are no longer connected with A.
  - Reconnect means, we randomly increase those 0s to 1, which means those disconnected synapses could be reconnected later.

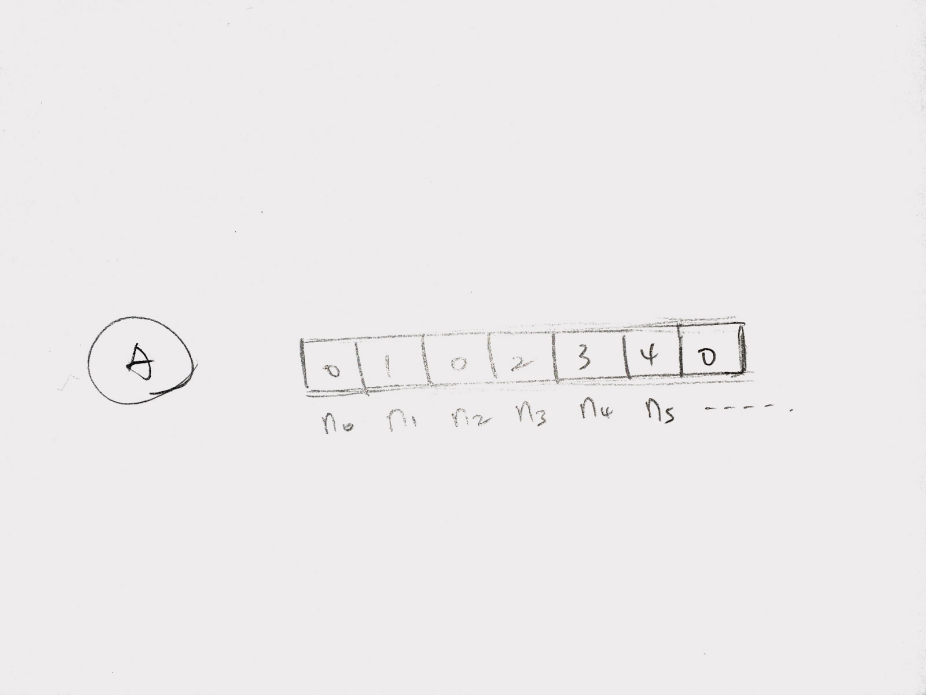


Figure 4.4

4.3. Language used

We choose to use **Scala** because as following reasons:

- **Scalable language**
  - Feel like scripting language. The syntax is concise and low ceremony.
  - Static type, but the compiler could infer them.
- **Object-Oriented paradigm**

  The language supports the advanced component architecture through classes and traits.

- **Functional paradigm**
  - Discourage side effects
  - First class functions
  - Immutable data structures
- **Interoperation with Java**
  - Runs on JVM

- Java libraries, frameworks and tools are all available.

4.4. Tools used

We choose to use **Akka** library as the following reasons:

- **Actors**

  This is what really matters. Actors are very lightweight concurrent entities. They process messages asynchronously using an event-driven receive loop. Pattern matching against messages is a convenient way to express an actor's behavior. They raise the abstraction level and make it much easier to write, test, understand and maintain concurrent and/or distributed systems. **In such a way we could focus on workflow—how the messages flow in the system and the responding behaviours —instead of low level primitives like threads, locks and socket IO.**

  In our model of neural networks, each neurons are actors, and the behaviours of neurons would be message dependent. This abstractions level is just right to our purpose in the project.

4.5. How to generate output

One observer, and one being observed.

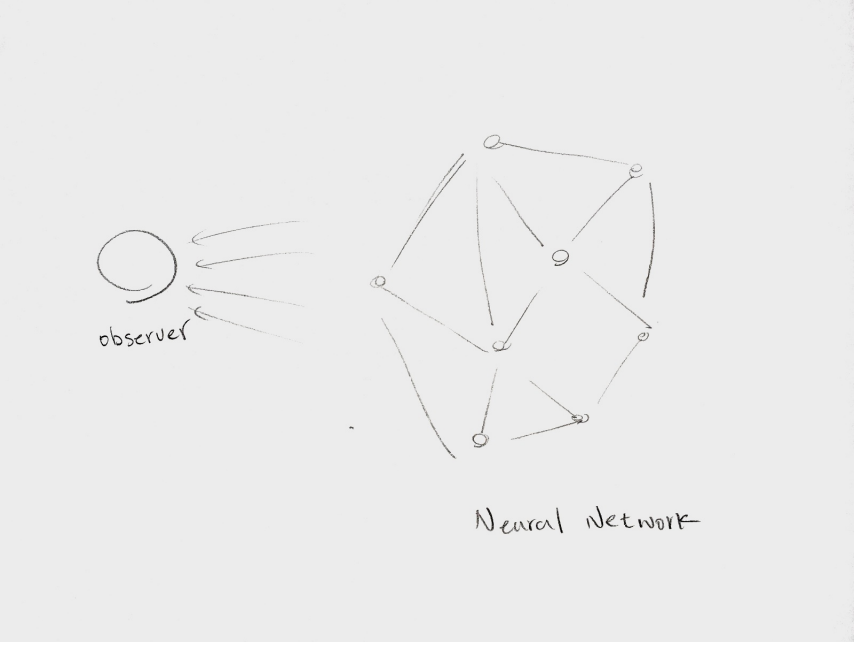


Figure 4.5

Each time a neuron choose to fire, it will send out firing signal to its neighbors and at the same time, send out a log message to observer actor. This actor will collect logging messages so the firing pattern would be preserved.

4.6. How to test your hypothesis

In the observer, logging data are represented as a sequence of neurons that participate in one firing phase. Then we use the “longest common sequence” algorithm to compare and evaluate how much two firing patterns are similar with each other. By doing so, we could evaluate if the firing patterns would converge per corresponding input stimulus.

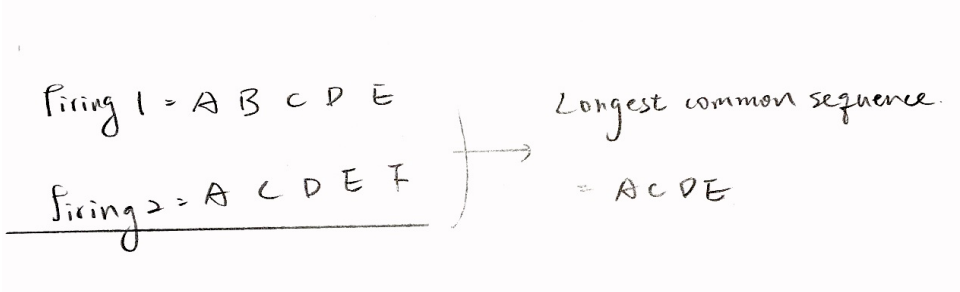


Figure 4.6

5. Implementation

5.1. Code architecture

5.1.1. **Main object file(main.scala)**:This the starting point from command line terminal. The program takes 2 arguments: server/supervisor , supervisor number server/supervisor option is designed for remote execution, which allows program running on different machine across internet. Supervisor number is for convenience of server to initiate the process.

5.1.2. **Message file(message.scala):** This file aggregate every message in the program. In akka actor model, threads interact with each other by message passing.

5.1.3. Actor file * 4:

- **Server.scala:** This file define the main control mechanism of the program. In order to coordinate the program running, server actor serves 2 major functions:
    - Aggregate full map of neurons: each neuron does not necessary need to know the full map, but in order to “randomly” create its neighbors, full map is needed to perform such randomness.
    - Check point for first input fire: Calculation of neighbor map is done by supervisor, and this computation and message circulation may take a while. So server here plays a check point. Only if each neuron knows it’s neighbors, the first neuron fire could be really fired.
- **Supervisor.scala:** This is the control unit for group of neurons. Each remote machine need a supervisor the manage it’s neurons. Supervisor performs 2 major functions:
    - Creating neurons: Each supervisor in different machine has to report to server at the beginning. The report includes how many neurons it is going to create. Then server will initiate the order to each supervisors. With server’s permission, supervisor could start to create neuron actors.
    - Computing and distributing it’s children neuron’s neighbor map: After server aggregate the full map, it wii distribute such map to each supervisors. Then supervisor will compute unique neighbor map for each children neurons.
- **Observer.scala:** 3 major functions:
    - Recorde neuron firing pattern.

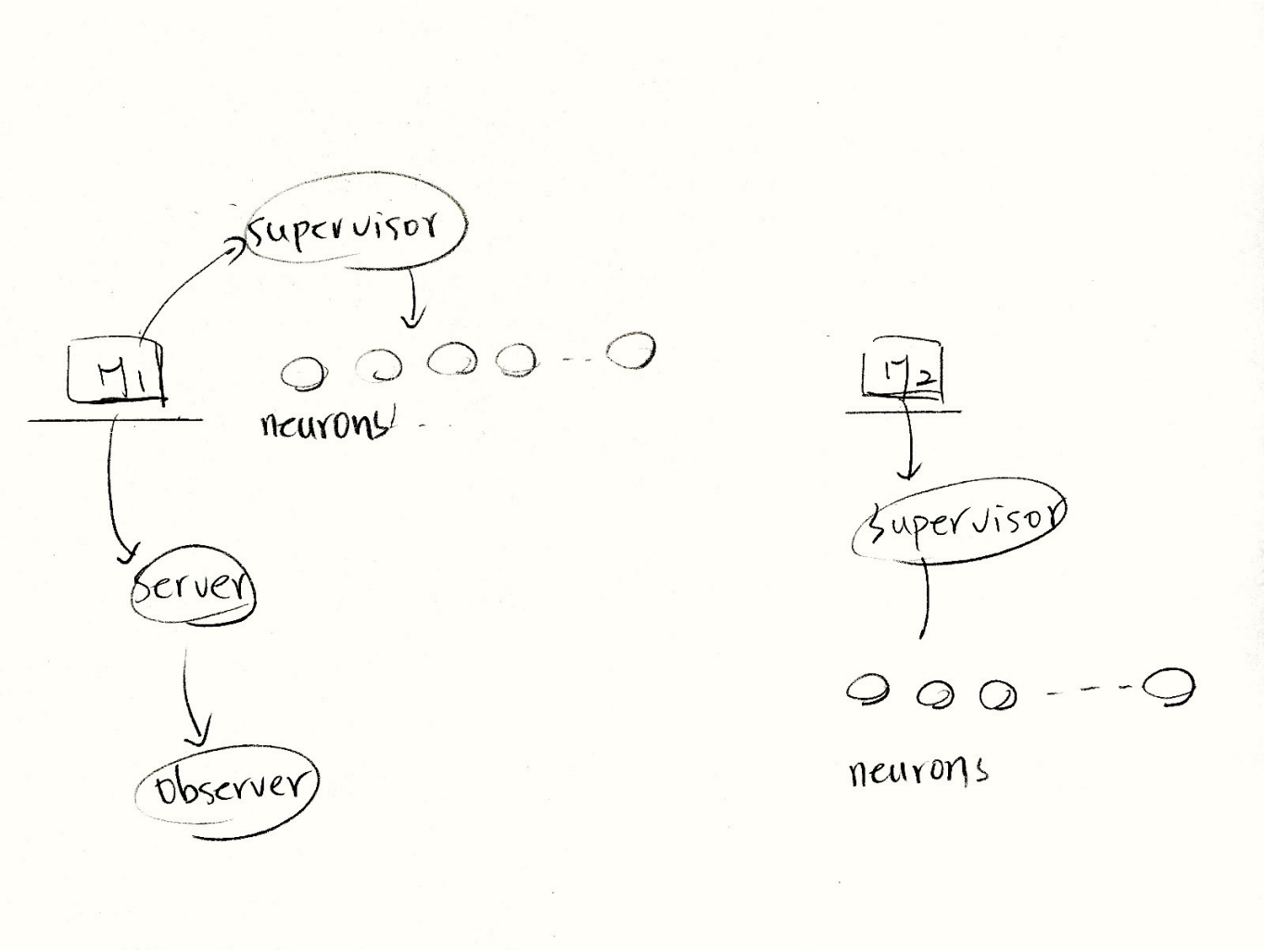
        - Each firing neuron will send a Log message to observer so that we could have complete logging information about who has been fired. Using this information, we could observe the neural network and see if it converge or not.
    - Compute the input fire.
    - Initiate the periodical input fire into the neural network.
        - In order to train the neural network, we have to fire the same input to the neural network repeatedly.
- **neuron.scala:** Neuron's function is very simple: decide to fire or not + ack or nak to neurons fired at you. Ack and Nak is used to changing the weight.

5.1.4. Configuration files *2 (server.config, supervisor.config)

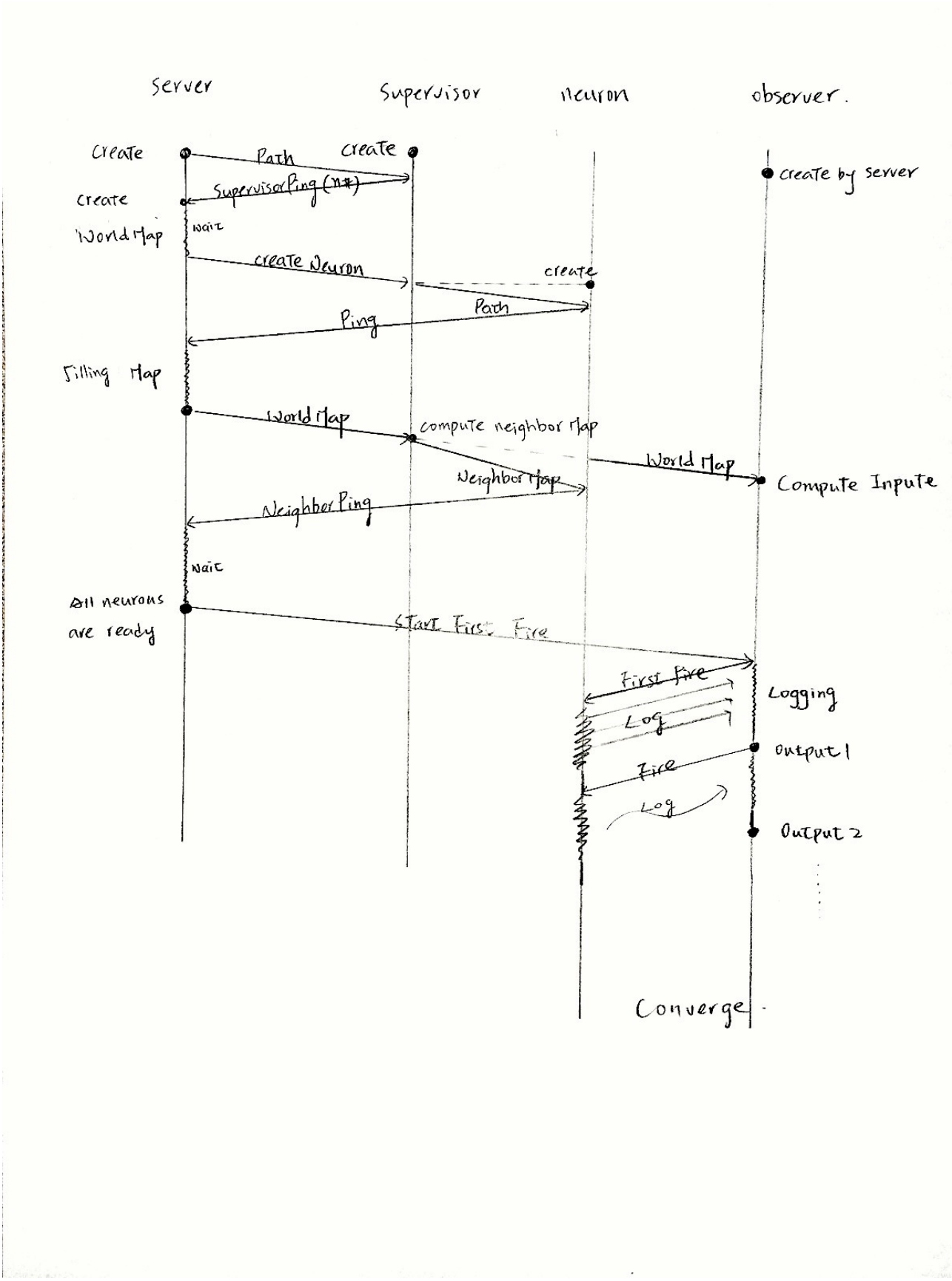
- Because server and supervisor could be separated on different machine, so they need different configuration files.

5.2. Design Flowchart

5.2.1. Physical Architecture:


supervisor
M1
neurons
server
observer
M2
supervisor
neurons

5.2.2. Message passing timeline



6. Data analysis and discussion

6.1. Output generation

In our implementation, we input parameters as shown below:

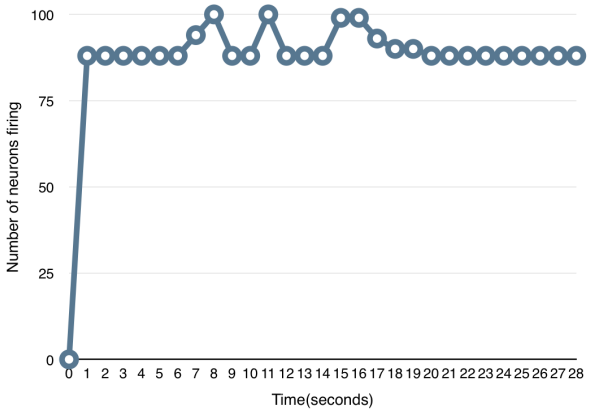
	Parameters	Trial #1	Trial #2	Trial #3	Trial #4	Trial #5	Trial #6	Trial #7	Trial #8
Neuron	threshold	25	18	18	25	25	25	18	25
	timeout	2s	2s	2s	2s	2s	2s	2s	2s
Observer	timeout	1s	1s	1s	1s	1s	1s	1s	1s
	inputRatio	0.1	0.04	0.04	0.04	0.04	0.01	0.04	0.05
Supervisor	neuronPerNode	100	250	400	400	500	1000	400	400
	neighborRatio	0.15	0.08	0.05	0.05	0.05	0.04	0.02	0.05
MACHINE		Macbook Pro	Macbook Pro	Macbook Pro	Macbook Pro	Macbook Pro	Macbook Pro	Macbook Pro	Macbook Pro

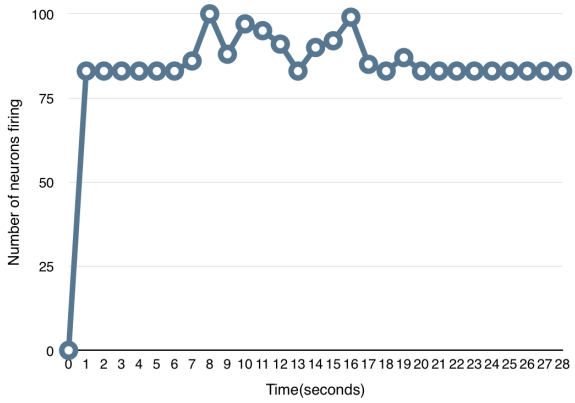
The sample output for trial #5 is shown below in appendix(9.2), for every fire triggered, the program show the number of neurons that fires and print out neurons identity that take part in this event.

6.2. Output analysis & Statistical regression

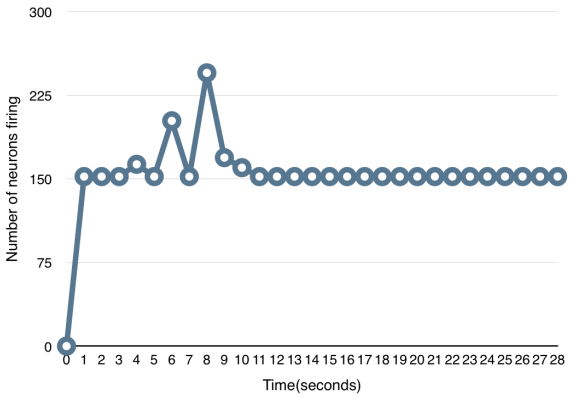
For every set of parameters we chose previously in our data generation, we executed it two times(for each trial) and plotted graphs for every trial we run. This are graph plotted Time(seconds) vs number of neurons:

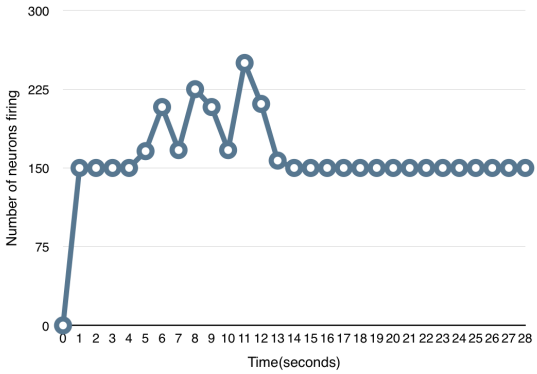
Trial #1:



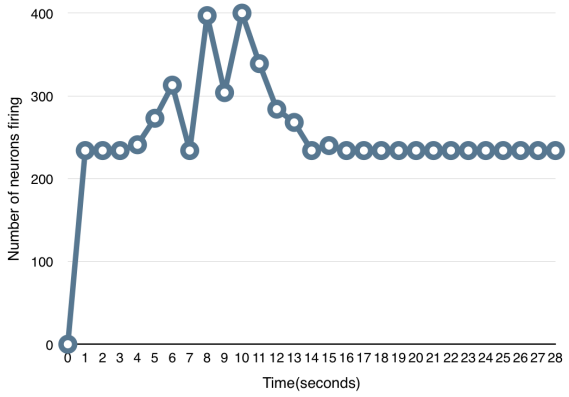


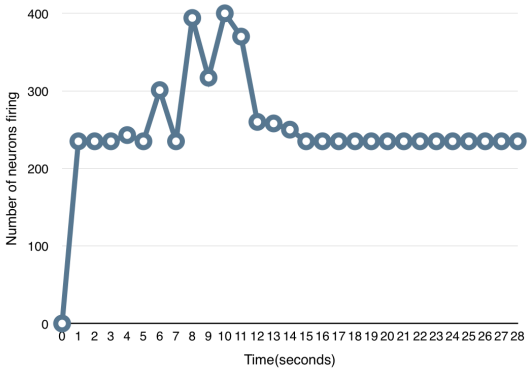
Trial #2:



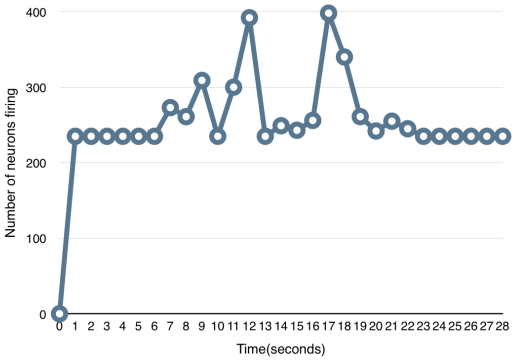


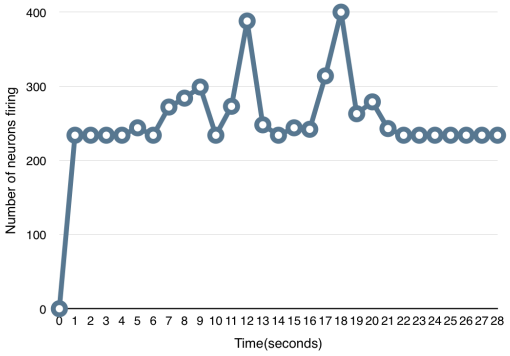
Trial #3:



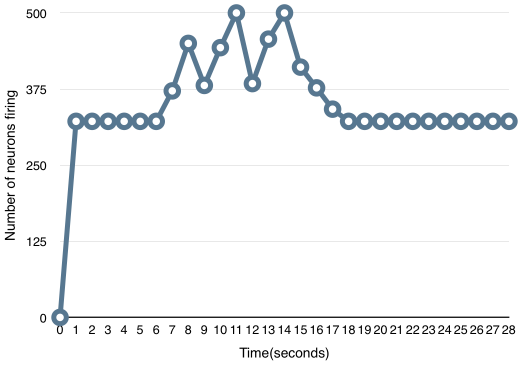


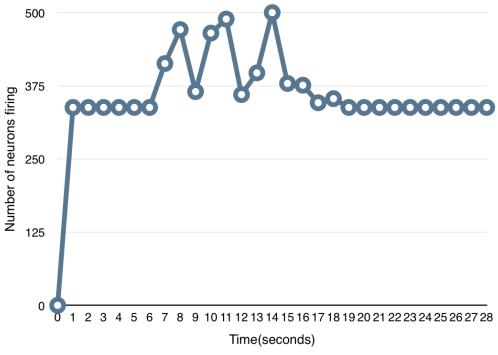
Trial #4:



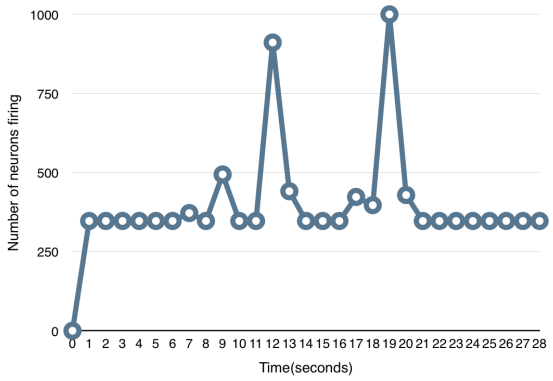


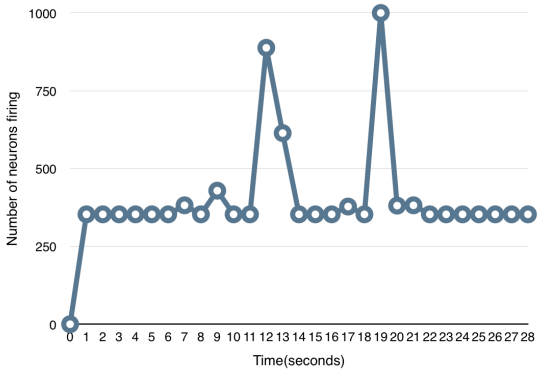
Trial #5:



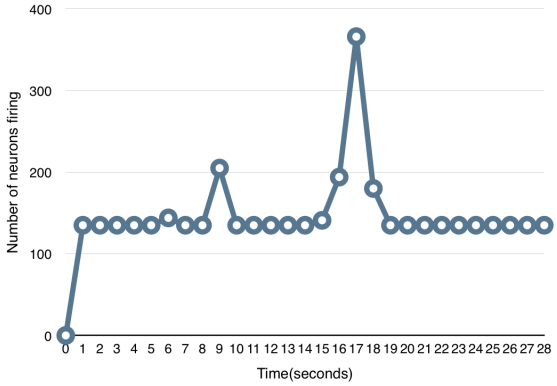


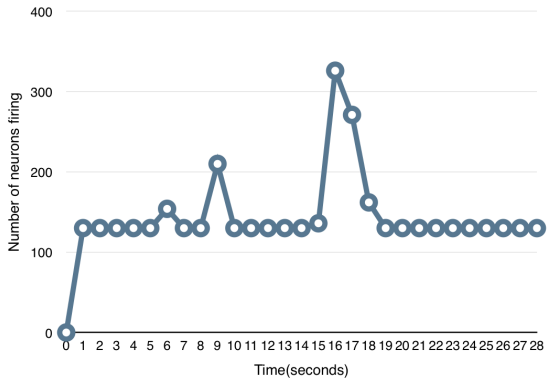
Trial #6:



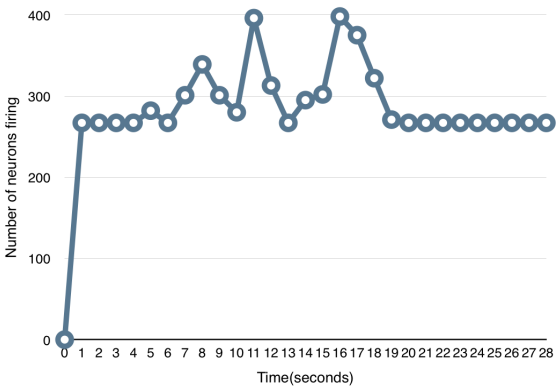


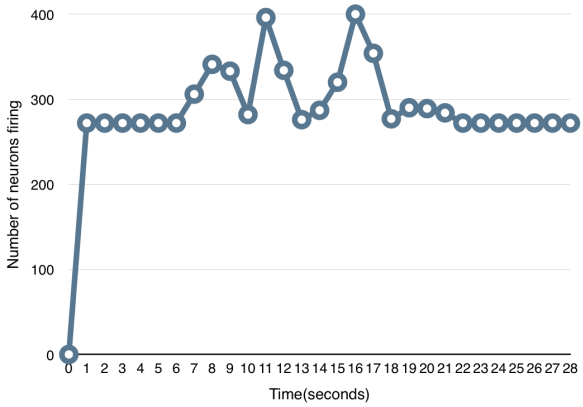
Trial #7:





Trail #8:





For every trial the graph always converge at some time in the future; In all of our trial, every one of cases will converge less than 30 seconds. When we start to initiate firing mechanism, assuming every other parameters are the same, if we have less threshold, the fires will cascade faster (Trial #3 vs #4). Difference in neighbor ratio will reduce overall neurons number which take part in cascaded fire (Trial #4

vs Trial #7). Comparing Trial #4 and Trial #8, we saw the same behaviour exhibited by neighbor ratio; higher input ratio magnitude, increases the number of neurons that take part in cascaded fire. Moreover, we found that the convergence neurons number is equal to the number of initial neurons which are triggered. After some time t, by receiving same repeated set of input (a set of initial neurons triggered), a set of neurons that have enough potential will start firing(expanding set of neurons who take part in firing sequence). Then after some time t, it will stabilize and return to initial number of neurons in initial set.

6.3. Compare output against hypothesis

As expected, by our hypothesis given a stimulus the neurons will start to fire to each other and expanding(more and more neurons cascading). After some time t, the set of neurons will stabilize and not changing anymore. Due time constraint, we are not able to implement our Reconnect mechanism, however we do implement our reweighting mechanism which allow each neurons to prioritize signal from one neuron from another(give more weight). This reweighting mechanism changes the state of related neuron neighbor each time when these neighbors take part in voting to decide not or to fire; this behaviour give some dynamic to our neural network. The hypothesis is true given we have full control of our parameters.

6.4. Abnormal case explanation (the most important task)

There are some limitation in our program when we tried to run several set of parameters. When running neurons number larger than 1000, the program will looks like it stand still. We suspect that because the processing power required grow exponentially, our machine is not able to handle it. Another reason for this problem might lies in our program architecture and library that we use to implement it. We need more time to understand how akka library handle its concurrent process, so that we might able to

find some limitation of the way akka actor interchange its messages to one another or maybe some convention which allow us to do the implementation more efficiently.

Parameters used in the example provided here are several sets that tested work in the program. There are many other sets which cause problems in the program. we need to be careful when we are choosing our parameter, for example: if the threshold is too high, the neurons firing sequence never cascade since we have a timer which reset neurons potential in case they are not firing for some duration of time. Another problem is when we set too high neighbor ratio or input ratio, these two cases cause lot of neurons to be triggered. Thus, requires lot of processing power which made our program stand still and waiting for responses from neurons.

6.5. Discussion

This is the most tentative try to simulate and understand the most complex mechanism,brain, in the world in my opinion. Obviously, the researches and experiments is far from complete, but we could still offer some lesson learned from this project. To appreciate what we do, let's go back to the question: why human want to simulate its own brain?

Two major motivations to do so:

1. **Functional concern:**

   Here exist some tasks that human could do at ease compared with current computers. The basic principle to find such task need you to recollect your introspective ability: try to find out those task we take for granted, which are those tasks computers can't do very well.

   What I mean here about easiness and hardness? We already saw a computer defeated chess world champion isn't it?

Cost and versatility is what really matters. A super computer could defeat a flesh would champion, but such computer can't learn new trick automatically. And the cost is very high. Energy cost: running human brain need comparatively very small portion of energy to a supercomputer. Sustain cost, maintenance cost and so on.

So basically, the process to simulate human brain is a process to learn how could we efficiently reproduce, tweak and implement what we already do efficiently on any manmade medium. Such medium could be on silicon chip, or anything human see fit.

Some applications imply huge possibility to improve human life and business profits.

2. **Out of curiosity:**

   We, human, sort of being hardcoded of the nature to be curious about the environment, even to human itself. So there is no surprise that we are curious about our own brain. During the past two decades, technology improvement really expanded the scope we could sense and measure. Microscope ,Scanning Electron microscope, Magnetic Resonance Image present the possibility for human to explore human brain.

So, in our simulation, the result resonates with our original hypothesis. Do we really decode the myth of how memory form? I don't think so. Here is the reason and what we could learn instead:

1) **Randomness**?

"As I have said so many times, God doesn't play dice with the world."

~Albert Einstein

We use so many random function in the code, why?

a) Because neuroscientist say so: at the beginning, how does our brain decide which neuron connect to which neuron even before any sensational stimulus? Random is the answer.
b) Because we don't know how to decide! We choose to use randomness in so many place because we don't know how it exactly work in reality.
    i) How do we decide which neuron should fire as the input?
    ii) How do we decide which neuron to reconnect, if there are so many disconnected neighbors?

Actually, randomness is easier said than done. In order to do randomness in appearance, we need to create a context for such randomness. For example, in order to create random neighbor map for each neuron, I need to aggregate whole map or each neurons so that I have the context to perform random selection. In this point of view, randomness is nothing more than other idea waiting to be implemented in software or even in hardware.

2) Simulating God's product vs Playing God?

In one abstraction layer, we know so well about how neurons interacting with each other. We've introduce so many ideas like, threshold, potential, re-weight and so on. We seem to know how things work and start to simulate. However, to implement into running code, we need to make hundreds of decisions. We digged deeper, and hit an abstraction layer so deep that we don't really know how things work. Then what happen? We have to decide. Choose parameters heuristically, try-and-error fashion or even arbitrarily. This is when we top simulating, but to playing God. Qualitatively, we know something about how things work, but quantitatively, we know so little. Here are some parameters we have to decide like God, but may not work out as God's real grand design.

a) **Threshold:** What number should it be? Every neuron has the same number across the neural network? Higher? Lower?
b) **Neuron timeout:** Potential should decay, what is the decay function?
c) **Neighbor ratio:** How many neurons could be a neighbor of one neuron?
d) **Input ratio:** How many neurons should fire simultaneously?

The problem not only lies in how and why we choose the quantity of those parameters, but how they interact and affect each other so that different result may emerge? As you can see, after finish the project, it brings us more questions than answers.

3) **Simultaneous and parallel?**

In reality, neurons are physically existing simultaneously. Even though in computation point of view, they don't compute very much, but the concurrence is an undeniable truth. Back to implementation, the most powerful machine we have at hand is a quad core laptop with 8 thread. So even we use akka actor to help use threading the process, they are not concurrent in fact. This could be a decisive difference and could pose huge impact on correctness and truthness of any neural network simulation.

Basically, time is just an idea about the order of change. And it is always relative. Exact point of time could be hard to distinguish, like when we say, these two events started at the same time, actually the only thing we can be sure is they happened closed enough for our current measuring scope, nothing more. If we could find a phenomenon changing periodically and faster than the temple of these two events, maybe we could tell the difference.

This hugely impacts the way we should imagine our simulation. Whenever we use the idea "same time", we need to know the limitation of our implementation, so the expectation could be set right because inevitable distortion is destined.

7. Conclusion and recommendations

7.1. Summary and conclusion

7.1.1. Summary:

Under the current architecture, our neural network receives predefined input, and the fire will cascade, maybe expand and contract in turns, then it shows the tendency to stabilize, which means our hypothesis could sustain under right configurations.

7.1.2. Conclusion:

The simulation here suggest that with good configuration, repeated firing input could result in stable final firing pattern. However, all questions aroused by this project have more values than the answer or result we try to present.

7.2. Recommendation for future studies

7.2.1. Build upon this project

-Neuron has two types: Excitatory and Inhibitory. In this project, we only implement excitatory neurons. However, inhibitory neuron is the key ingredient to construct neural network. Combining two type of neuron could produce different functions. We call this small scope network: Micro Network.

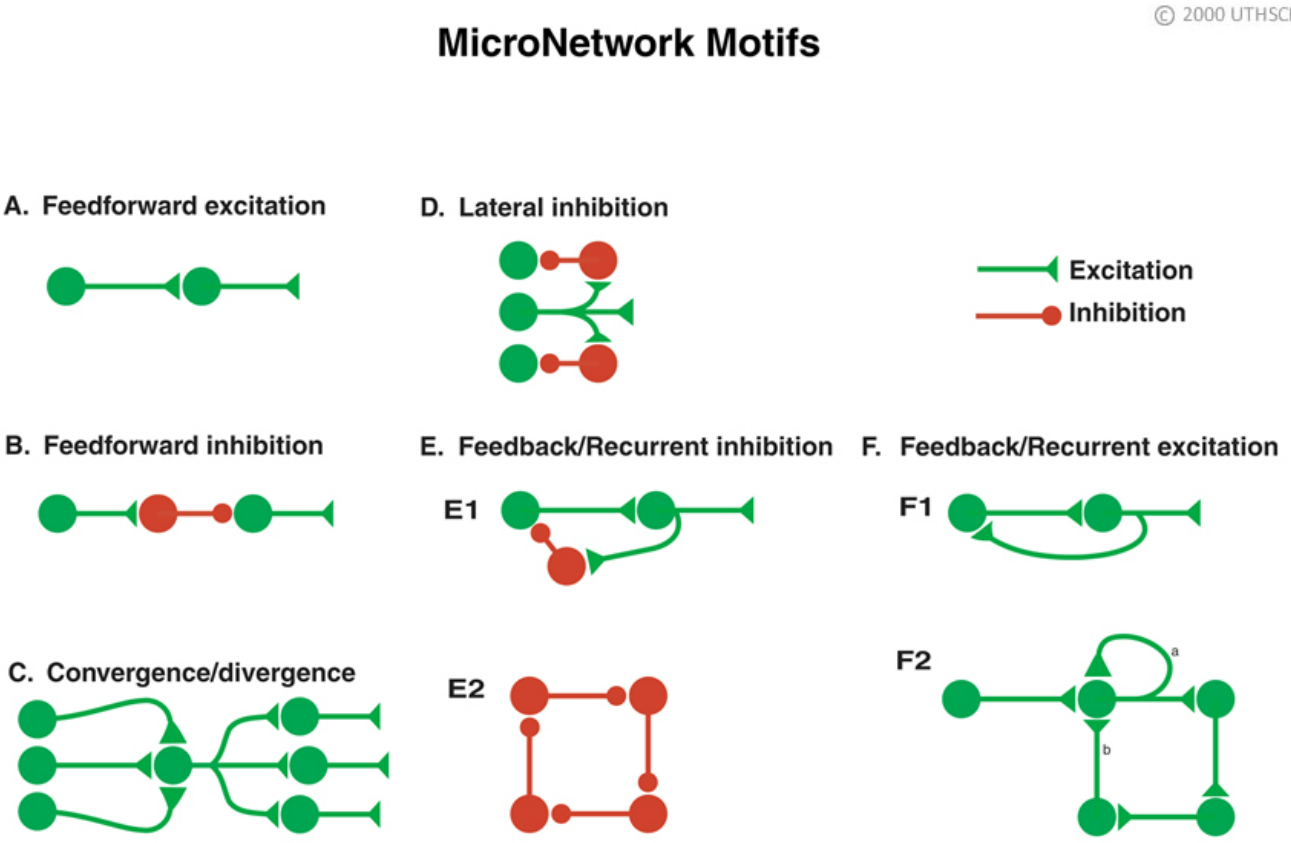


Figure 6

-Quantitative Formalization: formalize quantitative relationships between different parameters.

-Improve the performance of multi-threading

7.2.2. General future studies

In the business to combine Computer Science and Neuroscience, three ways to move forward.

1. **Improving computer science:**

   Parallel processing is just about to begin. If you think 8 cores is fantastic, just look our brain. It use 100 billion not so powerful cores and super intricate connection scheme to give us the ability to walk, eat, smile, feel and even to love.

2. **Improving neuroscience**

   a. **about tools: “Seeing is believing”**

      The real obstacle for neuroscience is we don’t have the right tool to get the detail visual in a cost-effective way. We are trying to understand human brain, but we don’t have so many chance to interact with it directly. Yes human can speak, but

that is also the result of complex brain behaviors. fMRI doesn't have enough resolution to analyze. SEM is freaking expensive and labor intensive to prepare the sample.

b. **about heuristics to tackle complexity: "where hierarchical model fails"**

From building space ship to particle accelerator at CERN, we use the one and the only heuristic: hierarchical structure. Now we are tackling brain, so we try to apply hierarchical structure to it as well. What is the result? The tension between phrenology and connectionism. Phrenologist believe human brain works in modules so that they try to map different functions to different zones. Connectionist believe we the whole brain should be considered as a whole. A neuron or a zone of neuron has its function, if any, is due the its connection and relationship to the entire neural network. We found pure phrenology has so limited explaining power to brain, but we couldn't find a way to develop connectionism in a systematic fashion. To me, phrenology is the last try for human to tackle this level of complexity by naively thinking simple divide and conquer could work as the old good day. They won't. We need new heuristic to tackle this unprecedented level of complexity.

Here I propose "developmental model." Let's take the seed for example. A seed may be easy to understand and analyze by current scientific approach. Our understanding for a seed is more than just a seed itself, we want to know how a seed could become a tree. The later is obviously performs more function and seems to be more complex than the former. The study of what is of a seed and how a seed interact with the environment could make it a different tree is precious. Because we could apply the gist of this developmental process to study how adult brain become what it is.

We don't have right tool to monitor human brain right now. We don't have many

chance to interact directly with real living brain now. And we don't have the right heuristic to tackle this level of complexity yet. Still we try to analyze adult brain directly by current approach? It time to face the truth of insufficiency of human intellectual power and move on.

3. **Improving the connection between two world**

   About neuron, this is a science, and about computer, this is an engineering subject. How do we get inspired from one side and apply what we learn to the other side? Engineering is used to sustain human life,survival issue. Now we are trying to engineering our brain, this is not about having a chance to survive. What we are trying to do is to exploring the chance to have a "good" life. Before we have the power to do so, we could start to rethink what is to be a good life and why would we really want from hi-tech?

8. Bibliography

1.http://www.research.ibm.com/cognitive-computing/neurosynaptic-chips.shtml?lnk=ushpls1#fbid=wRIw13XoOmo

2. http://www.artificialbrains.com

3.http://biomedicalengineering.yolasite.com/neurons.php

4.http://www.scala-lang.org/

5.http://akka.io/

6.http://www.research.ibm.com/cognitive-computing/neurosynaptic-chips.shtml?lnk=ushpls1#fbid=8W0z1SgqQv3

7.http://nengo.ca/build-a-brain/spaunvideos

8.http://bluebrain.epfl.ch/

9.Cognitive Computing Systems: Algorithms and Applications for Networks of Neurosynaptic Cores

Steve K. Esser, Alexander Andreopoulos, Rathinakumar Appuswamy, Pallab Datta, Davis Barch, Arnon Amir,John Arthur, Andrew Cassidy, Myron Flickner, Paul Merolla, Shyamal Chandrax, Nicola Basilicoy,Stefano Carpin,y, Tom Zimmerman, Frank Zeex, Rodrigo Alvarez-Icaza, Jeffrey A. Kusnitz, Theodore M. Wong,William P. Risk, Emmett McQuinn, Tapan K. Nayakz, Raghavendra Singhz, and Dharmendra S. Modha IBM Research - Almaden, San Jose, CA 95120 z IBM Research - India yUC Merced, Merced, CA 95343

10. A Review of Cell Assemblies by,Chris Huyck Peter Passmore October 14, 2011

11.A revised model of short-term memory and long-term learning of verbal sequences Neil Burgess, Graham J. Hitch. Institute of Cognitive Neuroscience and Department of Anatomy, University College London, 17 Queen Square, London WC1N 3AR, UK University of York, UK

Received 3 March 2006; revision received 4 August 2006

12.A synaptic model of memory: long-term potentiation in the hippocampus – T.V.P. Bliss & G.L. Colligridge

13.Cell Assemblies, Associative Memory and Temporal Structure in Brain Signals Thomas Wennekers and Gunther Palm Department of Neural Information Processing University of Ulm, D-89069 Ulm, Germany

14.Cell Assembly Dynamics in Detailed and Abstract Attractor Models of Cortical Associative Memory Anders Lansner, Erik Fransén, and Anders Sandberg

15.Long Short-Term Memory in Recurrent Neural Networks THESE ` N 2366 (2001) PRESENT ´ EE´ AU DEP´ ARTEMENT D'INFORMATIQUE

16.A Large-Scale Model of the Functioning Brain Chris Eliasmith,* Terrence C. Stewart, Xuan Choo, Trevor Bekolay, Travis DeWolf, Yichuan Tang, Daniel Rasmussen

17.Scala for the Impatient Cay S. Horstmann

18.Programming in Scala Martin Odersky, Lex Spoon, Bill Venners

19.Programming Scala Dean Wampler and Alex Payne

20.Scala By Example Martin Odersky

21.Algorithm and Software for Simulation of Spiking Neural Networks on the Multi-Chip SpiNNaker System Xin Jin, Francesco Galluppi, Cameron Patterson, Alexander Rast, Sergio Davies, Steve Temple, and Steve Furber

9. Appendices

9.1. Program source code with documentation

9.1.1. message.scala

```
package prototype

import akka.actor.ActorRef
import scala.collection.mutable.Map

case class Path(path:String) // to send path
case class SupervisorNumber(n:Int) // number of supervisor
case class SupervisorPing(neuronPerNode:Int) // supervisor send ping to server to inform the server of the number of neuron
case class WholeArray(array:Array[ActorRef]) // message containing ActorRef array
case class NeighborMap(map:Map[ActorRef,Int]) // message containing neighbormap


case object Ping // send ping
case object CreateNeuron // Ask create neuron
case object Test // for testing purposes
case object NeighborPing // neuron send ping to server
case object StartFirstFire // initiate fire

//Neuron Specific Messages

// Feedback for Reweighting,if value 1 increase, otherwise decrease
case object Ack
case object Nak
// Fire ignoring threshold (from stimulus)
case object DirectFire
// Fire taking threshold into consideration
case class GetFired(weight: Int)
// send to Log
case object Log
// Reconnect command
case object Reconnect
// Reset the potential
case object Reset


// sending neuron map for observer
case class NeuronMap(map:Map[ActorRef,Int]) //for observer
```

9.1.2. main.scala

```
package prototype

import akka.actor._
import com.typesafe.config.ConfigFactory

object Main{

	// our main app
		def main(args: Array[String]): Unit = {
		  //"Usage: ./sbt "run-main prototype.Main server/supervisor supNumber

	  // server path
		val serverPath = "akka.tcp://serverSystem@172.16.69.63:3333/user/server"

	  // server + supervisor start
		if (args(0) =="server") { //run server + supervisor
			val serverSystem = ActorSystem("serverSystem", ConfigFactory.load("server"))
			val server =serverSystem.actorOf(Props[Server],"server")
			val supervisorNumber = args(1).toInt

		// send the number of supervisor to server
			server ! SupervisorNumber(supervisorNumber)

			val supervisorSystem = ActorSystem("supervisorSystem", ConfigFactory.load("supervisor"))
			val supervisor = supervisorSystem.actorOf(Props[Supervisor])

			supervisor ! Path(serverPath)

		}
		else if (args(0) =="supervisor"){ //run supervisor only

			val supervisorSystem = ActorSystem("supervisorSystem", ConfigFactory.load("supervisor"))
			val supervisor = supervisorSystem.actorOf(Props[Supervisor])
	println("\tsupervisor started")

			supervisor ! Path(serverPath)

		}
	       }
}
```

9.1.3. server.scala

```
package prototype

import scala.collection.mutable.Set
import scala.collection.mutable.Map

import akka.actor._

class Server extends Actor{

		val supervisorSet = Set[ActorRef]() // path to all supervisor
		var supervisorNumber = 0 // supervisor number
		var supervisorCount = 0 // supervisor count, for verifying
		var neuronNumber = 0 // neuron number
		var neuronCount = 0  // neuron count, for verifying
		var neighborPingCount = 0 // ping count for verifying

		var neuronArray = new Array[ActorRef](1) // for first initialization
		var position = 0

		def receive ={
				case m:SupervisorNumber =>{ //predefined how many supervisor we are going to create
					supervisorNumber = m.n
				}
				case m:SupervisorPing =>{ //supervisor report to server, polling neuron number
					supervisorSet += sender //add sender

					neuronNumber += m.neuronPerNode //update whole neuron #

					supervisorCount += 1 //update supervisor #
					if (supervisorCount == supervisorNumber) {

						neuronArray = new Array[ActorRef](neuronNumber)

						supervisorSet.foreach( actorRef => actorRef ! CreateNeuron)
					}

				}
				case Ping =>{ //neurons report to server, collecting ActorRef
					neuronArray(position) = sender
					position += 1
```

```
                                        neuronCount += 1

                                        if (position == neuronArray.size){ //the array is filled already

                                                supervisorSet.foreach (actorRef => actorRef ! WholeArray(neuronArray)) //send out whole array to each supervisor

                                                val observer = context.actorOf(Props[Observer],"observer")
                                                val neuronMap = Map[ActorRef,Int]()
                                                for (i <- 0 until neuronArray.size){
                                                        neuronMap += ( neuronArray(i) -> i)
                                                }
                                                observer ! NeuronMap(neuronMap)
                                        }
                                }
                                case NeighborPing => // receiving ping
                                        neighborPingCount += 1

                                        if (neighborPingCount == neuronCount){
                                                context.children.head ! StartFirstFire
                                        }
                                case Test =>{ // for testing purposes
                                        println("\tTest message!! ( "+ self.path+" )")
                                }

        }
}
```

9.1.4. supervisor.scala

```
package prototype


import scala.util.Random
import akka.actor._
import scala.collection.mutable.Map

class Supervisor extends Actor{
        val neuronPerNode = 500 // number of neuron per supervisor
        val neighborRatio = 0.05 // neighbor ratio for each neuron, 1 is the highest

        var neuronArray = new Array[ActorRef](1) // initialize neuronArray var for storing neurons ActorRef

        var serverPath ="" // for storing the server path

        def receive = {

    // get server path
                case m:Path =>{
                        serverPath = m.path
                        context.actorSelection(serverPath) ! SupervisorPing(neuronPerNode)
                }

                case CreateNeuron =>{ //start signal from server to create neurons
                        for (i <- 1 to neuronPerNode){
                                var neuron = context.actorOf(Props[Neuron])
                                neuron ! Path(serverPath)
                        }
                }

    // receiving the whole address and put it inside array
                case m:WholeArray =>{
                        neuronArray = m.array
                        context.children.foreach( actorRef => actorRef ! NeighborMap(mkNeighbor(neuronArray)) )
                }

    // for testing purposes
                case Test =>{
                        println("\tTest message!! ( "+ self.path+" )")
                }
        }

  // for neighbors creation(for each neuron)
        def mkNeighbor(wholeMap:Array[ActorRef]): scala.collection.mutable.Map[ActorRef,Int] ={ //neighbor creation function
                val ranMachine = new Random()

                val scope = (wholeMap.size *neighborRatio).toInt
                val neighborMap = Map[ActorRef,Int]()

                val suffleArray = ranMachine.shuffle(0 to wholeMap.size-1).toArray

                for (i <- 0 to scope){
                        neighborMap += ( wholeMap(suffleArray(i)) -> 1)
                }

                return neighborMap
        }
}
```

9.1.5. neuron.scala

```
package prototype

import akka.actor._
import scala.util.Random
import scala.concurrent.duration._

//Neuron Actor implementation
class Neuron extends Actor {
  val threshold = 25 // predefined, neuron threshold
  var potential = 0 // current potential, if this potential >= threshold, it fires
  val timeout = 2 seconds // timeour, before neuron start to reset

  var neighborMap = scala.collection.mutable.Map[ActorRef, Int]() // storing neighbor ActorRef -> weight
  var firedBy = scala.collection.mutable.Set[ActorRef]() // storing neurons who fires to this

  // start the timer, start at timeout, with interval timeout
  var scheduler = context.system.scheduler.schedule(timeout,timeout,self, Reset)(context.system.dispatcher,self)
  scheduler.cancel // stop the timer

  var serverPath = "" // var for storing our server path
  var observerPath ="" // var for storing our observer path

  def receive = {

    // receiving server path and observer path
    case m:Path =>
      serverPath = m.path
      observerPath = m.path + "/observer"
      context.actorSelection(serverPath) ! Ping

    // receiving neighbormap
    case m:NeighborMap =>
      neighborMap = m.map
      context.actorSelection(serverPath) ! NeighborPing

    // initial fire, ignore potential
    case DirectFire =>
      potential = 0
      neighborMap.keys.foreach( i => i ! GetFired(neighborMap(i)))
      scheduler = context.system.scheduler.schedule(timeout,timeout,self, Reset)(context.system.dispatcher,self)
      context.actorSelection(observerPath) ! Log

    // fire, take potential into consideration
    case m:GetFired =>
      potential += m.weight
      firedBy += sender
      context.actorSelection(observerPath) ! Log

      // if potential greater equal than threshold fire, otherwise do nothing
      if (potential >= threshold) {
        neighborMap.keys.foreach( i => i ! GetFired(neighborMap(i)))
        firedBy.foreach( i => i ! Ack)
        potential = 0
        firedBy.clear

        scheduler.cancel
        scheduler = context.system.scheduler.schedule(timeout,timeout,self, Reset)(context.system.dispatcher,self)
      }

    // receiving ack, increment sender weight in neigborMap
    case Ack =>
      neighborMap(sender) += 1

    // receiving reset, send Nak to every neighbor, reset potential to 0, clear firedby
    case Reset =>
      firedBy.foreach(i => i ! Nak)
      potential = 0
      firedBy.clear

    // receiving Nak, decrement sender weight in neighborMap
    case Nak =>
      if (neighborMap(sender) == 1) neighborMap(sender) =0
      else neighborMap(sender) -= 1

    // test message, for testing purposes
    case Test =>
      println("\tTest message!! ( " + self + " )")

  }
}
```

9.1.6. observer.scala

```
package prototype

import akka.actor._
import akka.util._
```

```
import scala.concurrent.duration._
import java.util.Date
import scala.collection.mutable.Set
import scala.collection.mutable.Map

class Observer extends Actor {

  val timeout = 1 seconds // timeout for observer
  val inputRatio = 0.04 // input ratio 1 is the highest means 100%

  var oldSet = Set[Int]() // not implemented yet, to compare the neurons set
  var nowSet = Set[Int]() // not implemented yet, to compare the neurons set

  // the whole neurons map
  var wholeMap = Map[ActorRef,Int]()

  // for storing predefined input
  var input = scala.collection.immutable.Set[ActorRef]()
  var firstFireToggle = false

  //set receive timeout
  context.setReceiveTimeout( timeout )

  def receive = {

    // get the whole map
    case m:NeuronMap =>
      wholeMap = m.map
      input = wholeMap.slice(0, (wholeMap.size*inputRatio).toInt).keys.toSet

    // initiate the firing mechanism
    case StartFirstFire =>
      println("First fire\n")
      firstFireToggle = true

      input.foreach( i => i ! DirectFire)

    // receiving Log from neurons
    case Log =>
      nowSet += wholeMap(sender)

    // receive timeout do this
    case ReceiveTimeout =>
      if (oldSet.size == 0) {
        if (firstFireToggle) {
          println("observer: Number of neurons firing == " + nowSet.size)
          println("observer: who is firing: \n" + nowSet.mkString("-"))
          println()
        }
        oldSet = nowSet
        nowSet.clear
      }
      else {
        if (firstFireToggle) {
          println("observer: Number of neurons firing == " + nowSet.size)
          println("observer: who is firing:\n" + nowSet.mkString("-"))
          println()
        }
        oldSet = nowSet
        nowSet.clear
      }

      if (firstFireToggle) input.foreach( i => i ! DirectFire)


  }

}
```

9.1.7. server.conf

```
akka {

  actor {
    provider = "akka.remote.RemoteActorRefProvider"
  }

  remote {
    netty.tcp {
      hostname = "172.16.69.63"
      port = 3333
    }
  }

}
```

9.1.8. supervisor.conf

```
akka {

  actor {
    provider = "akka.remote.RemoteActorRefProvider"
  }

  remote {
    netty.tcp {
      hostname = "172.16.69.63"
      port = 0
    }
  }

}
```

9.2. Example output

```
[info] Set current project to prototype (in build file:/Users/kevinwidjaja/Desktop/final%20code/)
[info] Running prototype.Main server 1
[INFO] [03/18/2014 10:16:11.840] [run-main-0] [Remoting] Starting remoting
[INFO] [03/18/2014 10:16:12.016] [run-main-0] [Remoting] Remoting started; listening on addresses :[akka.tcp://serverSystem@172.16.202.139:3333]
[INFO] [03/18/2014 10:16:12.053] [run-main-0] [Remoting] Starting remoting
[INFO] [03/18/2014 10:16:12.066] [run-main-0] [Remoting] Remoting started; listening on addresses :[akka.tcp://supervisorSystem@172.16.202.139:54517]
First fire

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-177-156-148-127-106-98-77-454-48-27-404-375-304-296-123-94-73-421-23-44-342-292-271-242-221-213-163-92-469-440-419-267-259-238-209-188-159-138-28-486-109-465-457-59
-436-407-9-357-336-307-286-176-126-207-382-332-274-253-224-203-43-124-14-472-422-401-241-170-39-10-447-439-158-108-79-58-8-487-235-125-104-75-25-402-4-483-352-302-252-202-194-173-450-
429-400-269-350-219-190-169-140-111-467-69-90-61-11-367-115-107-86-57-36-434-7-334-284-255-234-184-155-482-105-453-55-432-301-280-251-230-222-201-180-172-151-122-449-399-370-349-299-
249-139-445-68-416-395-18-366-345-337-316-287-266-137-485-464-87-435-414-385-254-233-212-133-2-481-460-83-410-360-331-171-121-42-327-217-167-248-117-198-88-38-475-365-446-236-186-165
-34-413-392-282-363-232-313-182-132-74-53-32-24-3-351-330-149-128-78-70-49-20-397-347-297-268-218-197-147-495-314-293-264-214-193-143-135-114-462-441-85-35-412-312-283-458-60-81-408-3
1-489-358-308-258-250-229-200-179-150-19-477-427-406-398-377-246-225-196-15-494-96-452-444-394-373-344-323-134-84-390-361-290-261-240-211-80-161-30-488-459-438-328-409-278-359-228-19
9-47-476-455-405-376-355-326-145-116-37-393-474-322-272-162-141-91-62-33-12-470-420-289-260-239-210-189-181-160-131-110

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-177-156-148-127-106-98-454-77-48-27-404-375-304-296-123-94-73-421-23-44-342-292-271-242-221-213-163-92-469-440-419-267-259-238-209-188-159-138-28-109-486-465-457-59
-436-407-9-357-336-307-286-176-126-207-382-332-274-253-224-203-43-124-14-472-422-401-241-170-39-10-447-439-158-108-79-58-8-487-235-125-104-75-25-402-4-483-352-302-252-202-194-173-450-
429-400-269-350-219-190-169-140-111-467-69-90-61-11-367-115-107-86-57-36-434-7-334-284-255-234-184-155-105-482-453-432-55-301-280-251-230-222-201-180-172-151-122-449-399-370-349-299-
249-139-445-68-416-395-18-366-345-337-316-287-266-137-485-464-87-435-414-385-254-233-212-133-2-481-460-83-410-360-331-171-121-42-327-217-167-248-117-198-88-38-475-365-446-236-186-165
-34-413-392-282-363-232-313-182-132-74-53-32-24-3-351-330-149-128-78-70-49-20-397-347-297-268-218-197-147-495-314-293-264-214-193-143-135-114-462-441-85-412-35-312-283-458-60-81-408-3
1-489-358-308-258-250-229-200-179-150-19-477-427-406-398-377-246-225-196-15-494-96-452-444-394-373-344-323-134-84-390-361-290-261-240-211-80-161-30-488-459-438-328-409-278-359-228-19
9-47-476-455-405-376-355-326-145-116-37-393-474-322-272-162-141-91-62-33-12-470-420-289-260-239-210-189-181-160-131-110

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-177-156-148-127-106-98-77-454-48-27-404-375-304-296-123-94-73-421-23-44-342-292-271-242-221-213-163-92-469-440-419-267-259-238-209-188-159-138-28-109-486-465-457-59
-436-407-9-357-336-307-286-176-126-207-382-332-274-253-224-203-43-124-14-472-422-401-241-170-39-10-447-439-158-108-79-58-8-487-235-125-104-75-25-402-4-483-352-302-252-202-194-173-450-
429-400-269-350-219-190-169-140-111-90-69-467-61-11-367-115-107-86-57-36-434-7-334-284-255-234-184-155-105-482-453-432-55-301-280-251-230-222-201-180-172-151-122-449-399-370-349-299-
249-139-68-445-416-395-18-366-345-337-316-287-266-137-485-464-87-435-414-385-254-233-212-133-2-481-460-83-410-360-331-171-121-42-327-217-167-248-117-198-88-38-475-365-446-236-186-165
-34-413-392-282-363-232-313-182-132-74-53-32-24-3-351-330-149-128-78-70-49-20-397-347-297-268-218-197-147-495-314-293-264-214-193-143-135-114-462-85-441-35-412-312-283-458-60-81-31-40
8-489-358-308-258-250-229-200-179-150-19-477-427-406-398-377-246-225-196-15-494-96-452-444-394-373-344-323-134-84-390-361-290-261-240-211-80-161-30-488-459-438-328-409-278-359-228-19
9-47-476-455-405-376-355-326-145-116-37-393-474-322-272-162-141-91-62-33-12-470-420-289-260-239-210-189-181-160-131-110

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-177-156-148-127-106-98-454-77-48-27-404-375-304-296-123-94-73-421-23-44-342-292-271-242-221-213-163-92-469-440-419-267-259-238-209-188-159-138-28-109-486-465-457-59
-436-407-9-357-336-307-286-176-126-207-382-332-274-253-224-203-43-124-14-472-422-401-241-170-39-10-447-439-158-108-79-58-8-487-235-125-104-75-25-402-4-483-352-302-252-202-194-173-450-
429-400-269-350-219-190-169-140-111-467-69-90-61-11-367-115-107-86-57-36-434-7-334-284-255-234-184-155-105-482-453-432-55-301-280-251-230-222-201-180-172-151-122-449-399-370-349-299-
249-139-445-68-416-18-395-366-345-337-316-287-266-137-485-464-87-435-414-385-254-233-212-133-2-481-460-83-410-360-331-171-121-42-327-217-167-248-117-198-88-38-475-365-446-236-186-165
-34-413-392-282-363-232-313-182-132-74-53-32-24-3-351-330-149-128-78-70-49-20-397-347-297-268-218-197-147-495-314-293-264-214-193-143-135-114-462-441-85-35-412-312-283-458-60-81-408-3
1-489-358-308-258-250-229-200-179-150-19-477-427-406-398-377-246-225-196-15-494-96-452-444-394-373-344-323-134-84-390-361-290-261-240-211-80-161-30-488-459-438-328-409-278-359-228-19
9-47-476-455-405-376-355-326-145-116-37-393-474-322-272-162-141-91-62-33-12-470-420-289-260-239-210-189-181-160-131-110

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-177-156-148-127-106-98-77-454-48-27-404-375-304-296-123-94-73-421-23-44-342-292-271-242-221-213-163-92-469-440-419-267-259-238-209-188-159-138-28-109-486-465-457-59
-436-407-9-357-336-307-286-176-126-207-382-332-274-253-224-203-43-124-14-472-422-401-241-170-39-10-447-439-158-108-79-58-8-487-235-125-104-75-25-402-4-483-352-302-252-202-194-173-450-
429-400-269-350-219-190-169-140-111-467-69-90-61-11-367-115-107-86-57-36-434-7-334-284-255-234-184-155-105-482-453-432-55-301-280-251-230-222-201-180-172-151-122-449-399-370-349-299-
249-139-68-445-416-395-18-366-345-337-316-287-266-137-485-464-87-435-414-385-254-233-212-133-2-481-460-83-410-360-331-171-121-42-327-217-167-248-117-198-88-38-475-365-446-236-186-165
-34-413-392-282-363-232-313-182-132-74-53-32-24-3-351-330-149-128-78-70-49-20-397-347-297-268-218-197-147-495-314-293-264-214-193-143-135-114-462-441-85-35-412-312-283-458-60-81-408-3
1-489-358-308-258-250-229-200-179-150-19-477-427-406-398-377-246-225-196-15-494-96-452-444-394-373-344-323-134-84-390-361-290-261-240-211-80-161-30-488-459-438-328-409-278-359-228-19
9-47-476-455-405-376-355-326-145-116-37-393-474-322-272-162-141-91-62-33-12-470-420-289-260-239-210-189-181-160-131-110

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-177-156-148-127-106-98-77-454-48-27-404-375-304-296-123-94-73-421-23-44-342-292-271-242-221-213-163-92-469-440-419-267-259-238-209-188-159-138-28-109-486-465-457-59
-436-407-9-357-336-307-286-176-126-207-382-332-274-253-224-203-43-124-14-472-422-401-241-170-39-10-447-439-158-108-79-58-8-487-235-125-104-75-25-402-4-483-352-302-252-202-194-173-450-
429-400-269-350-219-190-169-140-111-467-69-90-61-11-367-115-107-86-57-36-434-7-334-284-255-234-184-155-105-482-453-432-55-301-280-251-230-222-201-180-172-151-122-449-399-370-349-299-
249-139-68-445-416-395-18-366-345-337-316-287-266-137-485-464-87-435-414-385-254-233-212-133-2-481-460-83-410-360-331-171-121-42-327-217-167-248-117-198-88-38-475-365-446-236-186-165
-34-413-392-282-363-232-313-182-132-74-53-32-24-3-351-330-149-128-78-70-49-20-397-347-297-268-218-197-147-495-314-293-264-214-193-143-135-114-462-441-85-35-412-312-283-458-60-81-408-3
1-489-358-308-258-250-229-200-179-150-19-477-427-406-398-377-246-225-196-15-494-96-452-444-394-373-344-323-134-84-390-361-290-261-240-211-80-161-30-488-459-438-328-409-278-359-228-19
9-47-476-455-405-376-355-326-145-116-37-393-474-322-272-162-141-91-62-33-12-470-420-289-260-239-210-189-181-160-131-110

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-177-156-148-127-106-98-77-454-48-27-404-375-304-296-123-94-73-421-23-44-342-292-271-242-221-213-163-92-469-440-419-267-259-238-209-188-159-138-28-109-486-465-457-59
-436-407-9-357-336-307-286-176-126-207-382-332-274-253-224-203-43-124-14-472-422-401-241-170-39-10-447-439-158-108-79-58-8-487-235-125-104-75-25-402-4-483-352-302-252-202-194-173-450-
429-400-269-350-219-190-169-140-111-467-69-90-61-11-367-115-107-86-57-36-434-7-334-284-255-234-184-155-105-482-453-432-55-301-280-251-230-222-201-180-172-151-122-449-399-370-349-299-
249-139-445-68-416-18-395-366-345-337-316-287-266-137-485-464-87-435-414-385-254-233-212-133-2-481-460-83-410-360-331-171-121-42-327-217-167-248-117-198-88-38-475-365-446-236-186-165
-34-413-392-282-363-232-313-182-132-74-53-32-24-3-351-330-149-128-78-70-49-20-397-347-297-268-218-197-147-495-314-293-264-214-193-143-135-114-462-441-85-35-412-312-283-458-60-81-408-3
1-489-358-308-258-250-229-200-179-150-19-477-427-406-398-377-246-225-196-15-494-96-452-444-394-373-344-323-134-84-390-361-290-261-240-211-80-161-30-488-459-438-328-409-278-359-228-19
9-47-476-455-405-376-355-326-145-116-37-393-474-322-272-162-141-91-62-33-12-470-420-289-260-239-210-189-181-160-131-110

observer: Number of neurons firing == 372
observer: who is firing:
```

0-356-466-437-306-387-206-177-156-148-127-106-98-77-454-48-27-404-375-325-304-296-123-94-73-421-23-44-342-292-271-242-221-213-163-142-92-469-440-419-267-259-238-209-188-159-138-28-10
9-486-465-457-59-436-407-386-9-357-336-307-286-176-126-207-403-382-332-324-274-253-224-203-43-124-14-472-451-422-401-372-241-170-39-120-10-447-439-158-108-79-58-29-8-487-285-154-235-
125-104-75-25-402-4-483-352-302-383-252-202-194-173-479-450-429-400-269-350-219-300-190-169-140-111-467-69-90-61-40-498-11-367-115-107-86-57-36-434-7-334-284-255-234-184-155-482-105-
453-432-55-301-280-251-230-222-201-180-172-151-122-499-478-101-72-428-449-399-378-370-349-299-270-249-139-445-68-416-18-395-366-345-337-316-287-266-166-137-485-464-443-87-435-414-385
-254-233-212-102-52-133-2-481-460-83-410-381-360-331-171-121-42-327-217-167-248-117-198-88-38-475-365-446-236-186-165-34-463-413-392-282-363-232-313-182-263-132-82-74-53-32-430-24-3-3
51-330-149-128-78-70-49-20-397-347-297-268-226-218-197-147-495-314-293-264-243-214-193-143-135-114-462-64-441-85-412-35-312-283-458-60-81-408-31-489-358-329-308-258-250-229-200-179-1
50-19-477-100-427-406-398-377-246-225-196-175-15-494-473-452-96-444-394-373-344-323-244-134-113-84-390-361-311-290-261-240-211-80-161-30-488-1-459-438-328-409-278-359-228-199-47-26-4
84-476-455-405-376-355-326-145-116-95-45-37-393-474-322-272-162-141-91-62-33-12-470-420-289-260-239-231-210-189-181-160-131-110

observer: Number of neurons firing == 450
observer: who is firing:
0-356-437-306-387-256-206-177-156-148-127-106-98-454-77-48-27-404-425-375-354-325-304-296-275-123-94-73-471-421-23-44-371-342-321-292-271-242-221-213-192-163-142-490-92-469-440-419-2
67-259-238-209-188-159-138-28-486-109-465-457-59-436-407-9-357-336-307-286-176-257-126-207-97-403-382-332-324-274-253-224-203-93-174-43-124-14-480-472-451-422-401-291-372-241-170-39-
497-10-468-447-439-418-389-368-318-158-108-79-58-29-8-487-335-204-285-154-235-125-104-75-54-25-402-4-483-352-433-302-383-273-252-223-202-194-173-144-450-429-319-400-269-350-219-300-1
90-169-140-119-111-90-69-467-61-40-498-11-367-388-338-317-288-136-115-107-86-57-36-434-7-384-334-305-284-255-234-205-184-155-105-482-453-432-55-301-280-251-230-222-201-180-172-151-12
2-499-101-449-72-428-399-378-370-349-320-299-270-249-139-68-445-416-395-18-366-345-337-316-295-287-266-237-166-56-137-493-485-464-87-443-435-414-385-364-254-233-212-102-183-133-2-481
-460-83-431-410-381-360-331-281-171-121-42-327-217-298-167-248-117-198-88-67-38-17-475-365-446-315-396-265-346-236-215-186-165-34-13-492-5-463-442-413-392-282-363-232-313-182-263-132
-103-82-74-53-430-32-24-380-3-461-351-330-178-149-128-78-70-49-397-20-347-297-268-226-218-197-168-147-495-118-314-293-264-214-193-185-143-135-114-491-462-85-64-441-412-391-35-362-333-
312-283-262-458-60-81-31-408-489-379-358-308-279-258-250-229-200-179-150-129-19-477-456-448-427-406-50-398-377-246-225-196-175-15-494-96-452-444-423-46-394-373-344-323-244-134-113-84
-63-411-390-369-361-290-261-240-130-211-80-161-30-488-1-459-438-328-409-278-359-228-199-47-26-484-476-455-426-405-376-355-326-145-116-66-37-393-474-343-322-191-272-162-141-112-91-62-
41-33-12-470-339-420-310-289-260-239-231-210-189-181-160-131-110

observer: Number of neurons firing == 381
observer: who is firing:
0-356-306-387-227-206-177-156-148-127-106-98-77-454-425-27-404-48-375-325-304-296-123-94-73-471-421-23-44-342-292-271-242-221-213-163-92-469-440-419-267-259-238-209-188-159-138-28-10
9-486-465-457-59-436-407-9-357-336-307-286-176-126-207-97-403-382-332-303-274-253-224-203-43-124-14-472-422-401-372-241-89-170-39-120-10-468-447-439-389-208-158-108-79-58-8-487-235-1
25-104-75-25-4-402-483-352-302-252-223-202-194-173-479-450-429-319-400-269-350-219-300-190-169-140-119-111-467-69-90-61-498-11-367-288-136-115-107-86-57-36-434-384-7-334-305-284-255-
234-205-184-155-482-105-453-55-432-301-280-251-230-222-201-180-172-151-122-499-72-428-449-399-370-349-299-270-249-139-220-68-445-416-395-18-366-345-337-316-287-266-237-166-56-137-6-4
85-464-443-87-435-414-385-254-233-212-183-133-2-481-460-83-410-360-331-171-121-42-327-217-167-248-117-198-88-38-17-475-365-446-236-186-165-34-492-5-463-413-392-282-363-232-313-182-15
3-132-103-82-74-53-32-24-3-351-330-149-128-78-70-49-20-397-347-297-268-247-218-197-147-495-424-314-293-264-243-214-193-164-143-135-114-491-462-441-85-412-391-35-312-283-458-60-81-408-
31-489-358-329-308-279-258-250-229-200-179-150-129-19-477-448-427-406-398-377-246-225-196-15-494-96-452-444-394-373-344-323-294-134-84-390-361-340-290-261-240-130-211-80-161-30-488-4
59-438-328-409-278-359-228-199-47-476-455-405-376-355-245-326-145-116-37-393-474-322-191-272-162-141-91-62-33-12-470-420-310-289-260-239-231-210-189-181-160-131-110

observer: Number of neurons firing == 443
observer: who is firing:
0-356-306-387-277-256-227-206-177-156-148-127-106-98-77-454-48-27-404-425-375-354-325-304-296-275-123-94-73-471-421-23-44-371-342-321-292-271-242-221-213-192-163-142-490-92-469-440-4
19-267-259-238-209-188-159-138-28-109-486-465-457-59-436-407-386-9-357-336-307-286-176-257-126-207-97-76-403-382-374-353-332-324-274-253-224-203-93-43-124-14-472-422-401-291-372-241-
89-170-39-497-10-468-447-439-418-368-208-158-108-79-58-8-487-335-285-235-125-104-75-25-402-4-483-352-433-302-383-252-223-202-194-173-152-479-450-429-319-400-269-350-219-300-190-169-1
40-111-467-69-90-61-417-11-367-388-317-288-157-136-115-107-86-57-36-434-7-384-334-305-284-255-234-184-155-105-482-453-432-55-301-280-251-230-222-201-180-172-151-122-478-101-499-449-7
2-399-22-370-349-320-299-270-249-139-220-445-68-416-395-18-366-345-337-316-295-287-266-237-216-187-166-56-137-493-6-485-464-87-435-414-385-254-233-212-102-133-2-481-460-83-431-410-36
0-331-171-121-71-42-21-348-327-217-167-248-117-198-88-67-38-17-496-475-365-446-396-265-346-236-215-186-165-34-492-5-463-413-392-282-363-232-313-182-263-132-82-74-53-32-24-3-380-351-33
0-178-149-128-99-78-70-49-20-397-347-297-268-226-218-197-147-495-118-424-314-293-264-243-214-193-185-143-135-114-462-441-85-412-391-35-362-341-333-312-283-262-458-81-60-408-31-489-37
9-358-308-258-250-229-200-179-150-129-19-477-456-448-427-406-50-398-377-246-225-196-175-15-494-473-452-96-444-46-394-373-344-323-244-134-113-84-63-390-369-361-311-290-261-240-211-80-
161-30-488-1-459-438-328-409-278-359-228-199-47-26-484-476-455-426-405-376-355-326-195-276-145-116-95-66-45-37-393-16-474-343-322-191-272-162-141-91-62-41-33-12-470-339-420-310-289-2
60-239-210-189-181-160-131-110

observer: Number of neurons firing == 500
observer: who is firing:
0-356-437-306-387-256-206-156-148-106-98-454-48-404-354-304-296-123-471-73-421-23-371-321-271-221-213-163-469-419-238-188-138-486-436-386-336-286-97-403-353-303-253-203-14-451-401-89
-170-39-120-468-418-368-318-79-29-335-204-285-154-235-104-54-402-4-483-352-433-302-383-252-202-194-152-144-450-319-400-269-350-219-300-169-119-111-467-69-61-417-498-11-367-317-136-86
-434-36-384-334-284-234-184-482-432-301-251-201-151-499-101-449-399-349-299-249-68-416-18-366-316-266-216-166-464-414-364-233-102-183-52-133-2-481-83-431-381-331-281-42-348-217-298-1
67-248-117-198-67-17-496-365-446-315-396-265-346-215-165-34-463-413-282-363-232-313-182-263-132-82-74-430-32-24-380-461-330-149-99-49-397-347-297-247-197-147-495-314-264-214-164-114-
462-64-412-362-312-262-81-31-379-329-279-229-179-129-477-427-377-246-196-65-146-15-494-96-452-444-46-394-344-294-244-113-63-411-369-361-311-261-130-211-80-161-30-459-328-409-278-359-
228-309-47-484-476-426-376-245-326-195-276-145-95-45-37-393-474-343-162-112-62-12-310-260-210-160-110-466-277-227-177-127-77-425-27-375-325-275-94-44-342-292-242-192-142-490-92-440-2
67-259-209-159-28-109-465-457-59-415-407-9-357-307-176-257-126-207-76-382-374-332-324-274-224-93-174-43-124-480-472-422-291-372-241-497-10-447-439-389-208-158-108-58-8-487-125-75-25-
273-223-173-479-429-190-140-90-40-388-338-288-157-115-107-57-7-305-255-205-155-105-453-55-280-230-222-180-172-122-478-72-428-22-378-370-320-270-139-220-445-395-345-337-295-287-237-18
7-56-137-493-6-485-87-443-435-385-254-212-460-410-360-171-121-71-21-327-88-38-475-236-186-13-492-5-442-392-153-103-53-3-351-178-128-78-70-20-268-226-218-168-118-424-293-243-193-185-14
3-135-491-85-441-35-391-341-333-283-458-60-408-489-358-308-258-250-200-150-19-100-456-448-50-406-398-225-175-473-423-373-323-134-84-390-340-290-240-51-488-1-438-199-26-455-405-355-11
6-66-16-322-191-272-141-91-41-33-470-339-420-289-239-231-189-181-131

observer: Number of neurons firing == 384
observer: who is firing:
0-356-437-306-387-256-206-156-148-106-98-454-48-404-304-296-123-471-73-421-23-371-321-271-221-213-163-469-419-238-188-138-486-436-336-286-353-253-203-14-401-89-170-39-468-318-79-29-3
35-235-104-402-4-483-352-433-302-252-202-194-144-450-400-269-350-219-169-119-111-467-69-61-498-11-367-136-86-434-36-384-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68
-416-18-366-316-266-216-464-414-233-133-2-481-83-431-381-331-42-217-298-167-248-117-198-496-365-446-165-34-413-282-363-232-313-182-132-82-74-430-32-24-461-330-149-99-49-397-347-297-1
97-147-495-314-264-214-114-462-412-312-262-81-31-229-179-129-477-427-377-246-196-146-15-494-96-452-444-394-344-113-361-261-211-80-161-30-459-328-409-278-359-228-309-47-476-376-326-27
6-145-37-393-474-162-62-12-260-210-160-110-466-177-127-77-27-375-94-44-342-292-242-490-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-382-374-332-274-224-93-174-4
3-124-480-472-422-241-497-10-447-439-208-158-108-58-8-487-125-75-25-273-223-173-429-190-140-90-388-288-157-115-107-57-7-255-205-155-105-453-55-280-230-222-180-172-122-428-370-320-139
-445-395-345-337-287-56-137-485-87-435-385-254-212-460-410-360-171-121-21-327-88-38-475-236-186-13-492-5-392-53-3-351-128-78-70-20-268-218-118-293-193-143-135-85-441-35-283-458-60-408
-489-358-308-258-250-200-150-19-448-50-406-398-225-175-373-323-134-84-390-290-240-488-1-438-199-455-405-355-116-66-322-191-272-141-91-41-33-470-339-420-289-239-189-181-131

observer: Number of neurons firing == 457
observer: who is firing:
0-356-437-306-387-256-206-156-148-106-98-454-48-404-354-304-296-123-471-73-421-23-371-321-271-221-213-163-469-419-238-188-138-486-436-386-336-286-97-403-353-303-253-203-14-451-401-89
-170-39-120-468-418-368-318-79-29-335-285-154-235-104-54-402-4-483-352-433-302-383-252-202-194-152-144-450-319-400-269-350-219-300-169-119-111-467-69-61-417-498-11-367-86-434-36-384-
334-284-234-184-482-432-301-251-201-151-499-101-449-399-349-299-249-68-416-18-366-316-266-464-414-364-233-102-183-52-133-2-481-83-331-281-42-217-298-167-248-117-198-67-17-496-365-446
-315-215-165-34-463-413-282-363-232-313-182-263-132-74-430-32-24-461-330-149-99-49-397-347-297-247-197-147-495-314-264-214-164-114-462-64-412-362-312-262-81-31-379-329-279-229-179-12
9-477-427-377-246-196-65-146-15-494-96-452-444-46-394-344-294-361-311-261-130-211-80-161-30-459-328-409-278-359-228-309-47-484-476-376-326-195-145-95-45-37-393-474-343-162-62-12-260-
210-160-110-466-277-227-177-127-77-425-27-375-325-275-94-44-342-292-242-192-142-490-92-440-267-259-209-159-28-109-465-457-59-415-407-9-357-307-176-257-126-207-76-382-374-332-274-224-
93-174-43-124-480-472-422-291-372-241-10-447-439-389-208-158-108-58-8-487-125-75-25-273-223-173-479-429-190-140-90-40-388-288-157-115-107-57-7-305-255-155-105-453-55-280-230-222-180-
172-122-478-428-22-378-370-320-270-139-445-395-345-337-295-287-237-187-56-137-6-485-87-435-385-254-212-460-410-360-171-121-21-327-88-38-475-236-186-13-392-53-3-351-178-128-78-70-20-2
68-226-218-168-118-424-293-243-193-143-135-491-85-441-35-391-341-283-458-60-408-489-358-308-258-250-200-150-19-100-456-448-50-406-398-225-175-473-423-373-323-134-84-390-290-240-488-1
-438-199-26-455-405-355-116-322-191-272-141-91-41-33-470-339-420-289-239-231-189-181-131

observer: Number of neurons firing == 500
observer: who is firing:
0-356-437-306-387-256-206-156-148-106-98-454-48-404-354-304-296-123-471-73-421-23-371-321-271-221-213-163-469-419-238-188-138-486-436-386-336-286-97-403-353-303-253-203-14-451-401-89
-170-39-120-468-418-368-318-79-29-335-204-285-154-235-104-54-402-4-483-352-433-302-383-252-202-194-152-144-450-319-400-269-350-219-300-169-119-111-467-69-61-417-498-11-367-317-136-86
-434-36-384-334-284-234-184-482-432-301-251-201-151-499-101-449-399-349-299-249-68-416-18-366-316-266-216-166-464-414-364-233-102-183-52-133-2-481-83-431-381-331-281-42-348-217-298-1
67-248-117-198-67-17-496-365-446-315-396-265-346-215-165-34-463-413-282-363-232-313-182-263-132-82-74-430-32-24-380-461-330-149-99-49-397-347-297-247-197-147-495-314-264-214-164-114-
462-64-412-362-312-262-81-31-379-329-279-229-179-129-477-427-377-246-196-65-146-15-494-96-452-444-46-394-344-294-244-113-63-411-369-361-311-261-130-211-80-161-30-459-328-409-278-359-
228-309-47-484-476-426-376-245-326-195-276-145-95-45-37-393-474-343-162-112-62-12-310-260-210-160-110-466-277-227-177-127-77-425-27-375-325-275-94-44-342-292-242-192-142-490-92-440-2
67-259-209-159-28-109-465-457-59-415-407-9-357-307-176-257-126-207-76-382-374-332-324-274-224-93-174-43-124-480-472-422-291-372-241-497-10-447-439-389-208-158-108-58-8-487-125-75-25-
273-223-173-479-429-190-140-90-40-388-338-288-157-115-107-57-7-305-255-205-155-105-453-55-280-230-222-180-172-122-478-72-428-22-378-370-320-270-139-220-445-395-345-337-295-287-237-18
7-56-137-493-6-485-87-443-435-385-254-212-460-410-360-171-121-71-21-327-88-38-475-236-186-13-492-5-442-392-153-103-53-3-351-178-128-78-70-20-268-226-218-168-118-424-293-243-193-185-14

3-135-491-85-441-35-391-341-333-283-458-60-408-489-358-308-258-250-200-150-19-100-456-448-50-406-398-225-175-473-423-373-323-134-84-390-340-290-240-51-488-1-438-199-26-455-405-355-116-66-16-322-191-272-141-91-41-33-470-339-420-289-239-231-189-181-131

observer: Number of neurons firing == 411
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-471-73-421-23-321-271-221-213-163-469-419-238-188-138-486-436-336-286-97-403-353-253-203-14-401-89-170-39-318-79-29-285-154-235-104-54-402-4-483-352-302-383-252-202-194-144-450-400-269-350-219-300-169-119-111-467-69-61-11-367-317-136-86-434-36-384-334-284-234-184-482-432-301-251-201-151-101-449-399-349-299-249-68-416-18-366-316-266-166-464-414-233-102-52-133-2-481-83-431-331-42-217-298-167-248-117-198-17-496-365-446-315-346-215-165-34-463-413-282-363-232-313-182-263-132-74-430-32-24-461-330-149-99-49-397-347-297-197-147-495-314-264-214-164-114-462-412-312-262-81-31-279-229-179-129-477-427-377-246-196-65-15-494-96-452-444-394-344-294-244-411-361-311-261-211-80-161-30-459-328-409-278-359-228-47-484-476-426-376-326-276-145-37-393-474-343-162-62-12-310-260-210-160-110-277-177-127-77-27-375-325-94-44-342-292-242-142-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-76-382-332-324-274-224-93-174-43-124-480-472-422-291-241-497-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-388-338-115-107-57-7-305-255-155-105-453-55-280-230-222-180-172-122-72-428-22-370-270-139-220-445-395-345-337-287-237-137-485-87-443-435-385-254-212-460-410-360-171-121-71-21-327-88-38-475-236-186-13-492-5-392-153-103-53-3-351-178-128-78-70-20-268-218-168-118-293-243-193-143-135-85-441-35-391-283-458-60-408-489-358-308-258-250-200-150-19-456-50-406-398-225-423-373-323-134-84-390-340-290-240-51-488-1-438-199-26-455-405-355-116-322-191-272-141-91-41-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 377
observer: who is firing:
0-356-437-306-387-206-156-148-106-98-454-48-404-354-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-97-353-253-203-14-401-170-39-120-318-79-335-204-285-154-235-104-402-4-483-352-302-383-252-202-194-144-450-319-400-269-350-219-169-119-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-216-464-414-233-102-183-133-2-481-83-331-42-217-298-167-248-117-198-67-365-446-165-34-463-413-282-363-232-313-182-132-74-32-24-380-461-330-149-49-397-347-297-247-197-147-495-314-264-214-114-462-412-312-262-81-31-229-179-477-427-377-246-196-65-146-15-494-96-452-444-394-344-113-361-261-211-80-161-30-459-328-409-278-359-228-47-476-426-376-326-145-37-393-474-343-162-62-12-310-260-210-160-110-277-177-127-77-27-375-275-94-44-342-292-242-490-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-208-158-108-58-8-487-125-75-25-173-429-190-140-90-338-157-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-478-378-370-320-270-139-445-395-345-337-287-56-137-485-87-435-385-254-212-460-410-360-171-121-71-327-88-38-475-236-186-492-392-153-103-53-3-351-128-78-70-20-268-218-168-424-293-193-185-143-135-85-441-35-391-333-283-458-60-408-489-358-308-258-250-200-150-19-456-406-398-225-175-373-323-134-84-390-340-290-240-488-438-199-455-405-355-116-16-322-272-141-91-33-470-339-420-289-239-189-181-131

observer: Number of neurons firing == 342
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-386-336-286-353-253-203-14-401-170-39-79-335-235-104-402-4-483-352-302-252-202-194-450-400-269-350-219-169-119-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-166-464-414-233-133-2-481-83-331-42-348-217-167-248-117-198-365-446-396-346-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-361-261-211-80-161-30-459-328-409-278-359-228-309-47-476-376-326-145-37-393-474-343-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-76-382-332-274-224-43-124-472-422-241-10-447-439-389-158-108-58-8-487-125-75-25-173-429-190-140-90-388-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-378-370-139-445-395-345-337-287-56-137-485-87-443-435-385-254-212-460-410-360-171-121-21-327-88-38-475-236-186-442-392-53-3-351-128-78-70-20-268-218-168-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-200-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-231-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-450-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-117-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-361-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-200-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-450-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-117-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-361-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-200-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-450-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-117-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-361-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-200-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-450-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-117-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-361-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-200-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-450-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-117-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-361-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-200-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-450-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-117-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-361-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-200-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-4
50-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-11
7-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-3
61-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-
9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-
395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-20
0-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-4
50-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-11
7-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-3
61-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-
9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-
395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-20
0-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-4
50-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-11
7-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-3
61-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-
9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-
395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-20
0-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-4
50-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-11
7-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-3
61-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-
9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-
395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-20
0-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131

observer: Number of neurons firing == 322
observer: who is firing:
0-356-306-387-206-156-148-106-98-454-48-404-304-296-123-73-421-23-271-221-213-163-469-419-238-188-138-486-436-336-286-253-203-14-401-170-39-79-235-104-402-4-483-352-302-252-202-194-4
50-400-269-350-219-169-111-467-69-61-11-367-86-434-36-334-284-234-184-482-432-301-251-201-151-449-399-349-299-249-68-416-18-366-316-266-464-414-233-133-2-481-83-331-42-217-167-248-11
7-198-365-446-165-34-413-282-363-232-313-182-132-74-32-24-330-149-49-397-347-297-197-147-495-314-264-214-114-462-412-312-81-31-229-179-477-427-377-246-196-15-494-96-452-444-394-344-3
61-261-211-80-161-30-459-328-409-278-359-228-47-476-376-326-145-37-393-474-162-62-12-260-210-160-110-177-127-77-27-375-94-44-342-292-242-92-440-267-259-209-159-28-109-465-457-59-407-
9-357-307-176-126-207-382-332-274-224-43-124-472-422-241-10-447-439-158-108-58-8-487-125-75-25-173-429-190-140-90-115-107-57-7-255-155-105-453-55-280-230-222-180-172-122-370-139-445-
395-345-337-287-137-485-87-435-385-254-212-460-410-360-171-121-327-88-38-475-236-186-392-53-3-351-128-78-70-20-268-218-293-193-143-135-85-441-35-283-458-60-408-489-358-308-258-250-20
0-150-19-406-398-225-373-323-134-84-390-290-240-488-438-199-455-405-355-116-322-272-141-91-33-470-420-289-239-189-181-131