Computational Neuroscience
Final Project
Simulation of Neuron addition using Nengo

Yu-Wei Su b06607057[*]

Jan 15, 2021
July 31, 2023 (translated)

[*]Department of Economics, National Taiwan University

Contents

List of Figures

List of Tables

1 INTRODUCTION

In the class, it was mentioned that using SPA (Semantic Pointer Architecture) for vector addition and multiplication as a high-level cognitive operation can have some good properties, such as the ability to preserve information to a large extent and to transform low-level information into symbols for logical operations. In this implementation, I attempted to investigate how and to what extent that addition in the brain may be simulated with SPA.

2 METHODS AND MODELS

2.1 Methods

In this implementation, I used nengo-spa to simulate how SPA vectors can be utilized for adding numbers. The experimental model does not reference any actual physiological operating models; it solely investigates how neural networks can achieve addition within the SPA framework. Four different models were employed, primarily focusing on the effects when adding larger numbers.

Model 1 represents the simplest "1+1" model. Models 2 to 5 were constructed to test larger addition in different settings. The architecture and settings for these models are as follows:

2.2 Model Architecture

The model architecture is drawn as the following diagram. The input represents the neural vectors decoded from visual neurons. They are then sent to the higher-order cortical region called "VISION" which is responsible for visual processing.

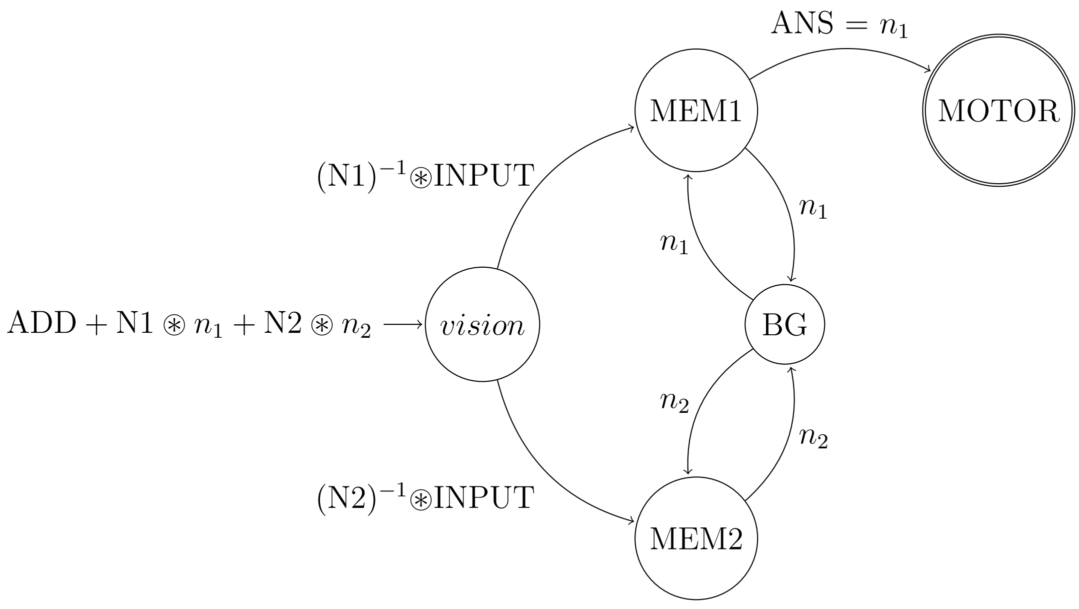
ANS $= n_1$
MEM1
MOTOR
$(\mathrm{N1})^{-1}$⊛INPUT
n_1
n_1
ADD + N1 ⊛ n_1 + N2 ⊛ n_2 ⟶
vision
BG
n_2
n_2
$(\mathrm{N2})^{-1}$⊛INPUT
MEM2

Figure 2.1 : Diagram of the model architecture

The main input structure is as follows:

$$\text{INPUT} = \text{ADD} + \text{N1} \circledast n_1 + \text{N2} \circledast n_2$$

ADD represents the vector for addition. N1 and N2 represent the semantic vectors for the operands (the numbers being added). n_1 and n_2 correspond to the specific numbers associated with the operands. This simulation aims to model the semantic vectors that might be generated when processing a statement like "1+1." The VISION cortex will then decode this information and send them to the Memory cortex MEM1 and MEM2.

$$(\text{N1})^{-1} \circledast \text{INPUT} \rightarrow \text{MEM1}$$
$$(\text{N2})^{-1} \circledast \text{INPUT} \rightarrow \text{MEM2}$$

The memory cortex will process the input and transmits its state to the Basal Ganglia and Thalamus regions (BG). The BG holds the mapping of addition rules, which means it contains the information necessary to perform addition operations. It will determine how the addition should be carried out.

```
if (state == $(n_1, n_2)$ and $n_2 > 0$)
  then (state = $(n_1 + 1, n_2 - 1)$)
else (send(MEM1, MOTOR), state = $(n_1, n_2)$)
```

N2 can be considered as the counting value. If N2 ¿ 0, the rules within the BG will change N1 into its successor N1+1. If N2 is zero, it indicates that the answer has been found, and it will be outputted. Consequently, the model needs to have a mapping of dimension $dim(N1) * dim(N2)$ to understand the next action for any given addend (N1) and its corresponding counting value (N2).

The heteroassociative memory model used in model 3 allows some of the state associations to be handled by Nengo's model. However, in practice, the number of rules needed would probably be the same. Table 2.1 shows the parameter in different models:

Model	model 1	model 2	model 3	model 4	model 5
Semantic Vector Dimension	64	64	64	64	128
Numbers	2+2	4+4	4+4	4+4	4+4
Associative Memory	None	None	HETER	WTA	WTA

Table 2.1: model parameters

3 RESULTS

Below are the descriptions of the diagrams for the regions output and its action selection output:

Output Graph: The Output Graph represents the temporal output of different model regions (vision, mem1, mem2, motor) over time. It illustrates how the information evolves and is processed through the various stages of the system. Each region's output is depicted along the timeline, allowing observation of the changes and interactions between the regions as the system progresses through different time steps. This graph provides insights into how the information is transformed and propagated across the different model regions throughout the entire process.

Action Selection Graph: The action selection model diagram represents the decision-making process involving the BG (Basal Ganglia) and Thalamus. It illustrates how the utility (or reward) for each potential action is calculated and compared. The action with the highest utility, reaching a certain threshold, will be selected as the chosen action. The diagram depicts the connections and interactions between the BG and Thalamus, showcasing how they process the utility information to make a decision.

Model 1

In Model 1, the input '0+2' initiates the computation process. Starting with an initial value of 0, the 'mem1' module undergoes '0' and '1', eventually reaching a value of 2. Meanwhile, the 'mem2' module begins with a value of 2 and gradually decreases to zero. Once 'mem2' reaches zero, the 'motor' module outputs the value stored in 'mem1,' resulting in Vector 2 indicating that the addition operation for '0+2' has been calculated. Model 2 further attempts to increase the numbers to 4.

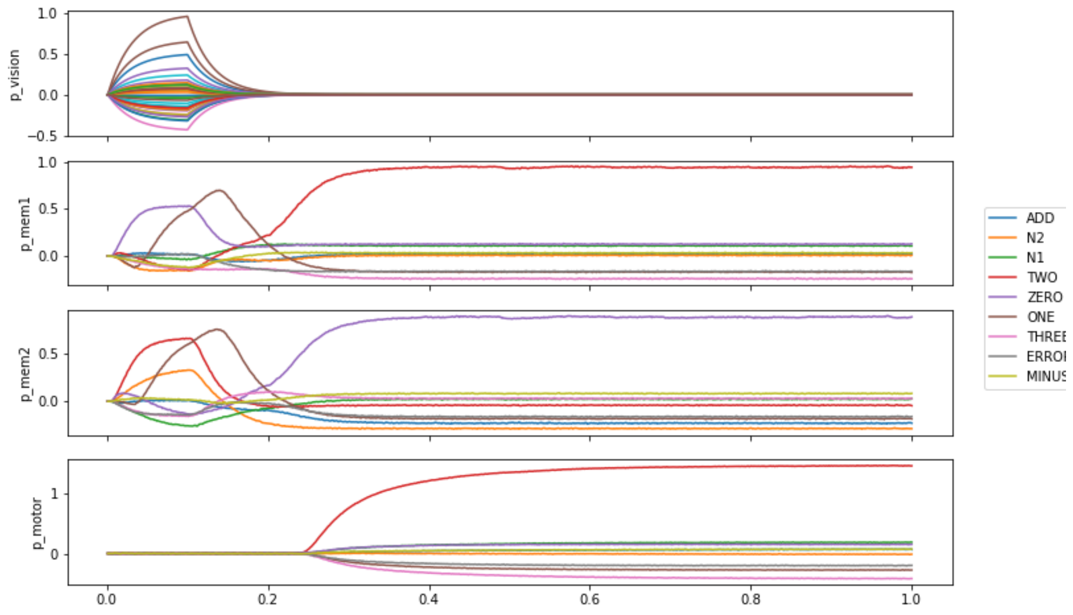

Figure 3.2 : model 1 output graph

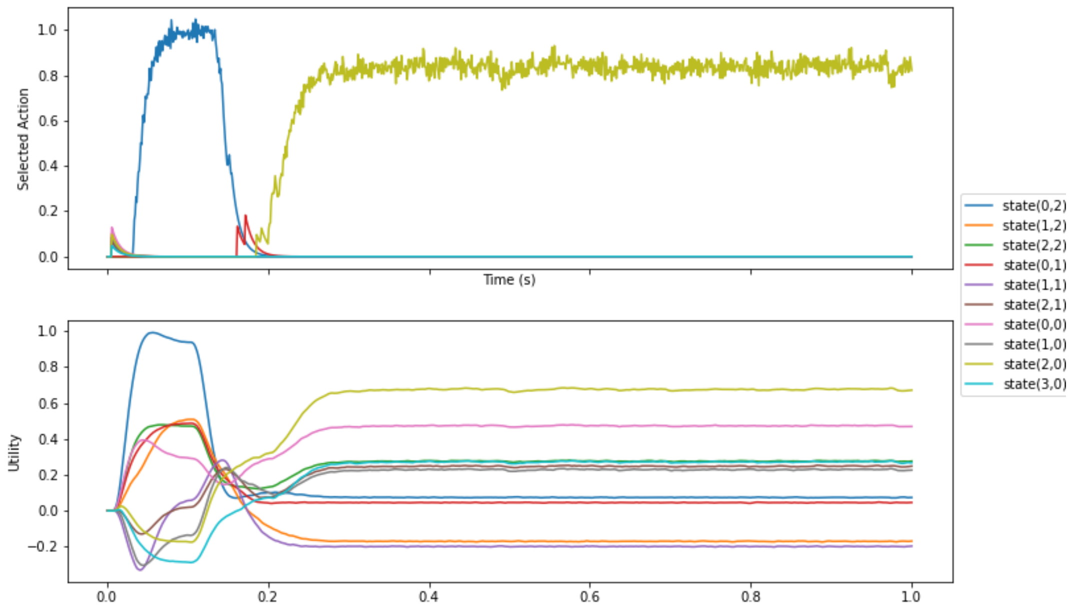

Figure 3.3 : model 1 action selection graph

Model 2

In Model 2, the input is '0+4', but the Output Graph for the 'motor' module shows no output, indicating that there is a problem recognizing input signals, and the model fails to produce the desired result.

Upon further observation of the BG and utility graphs, it is noticed that the utility values among different states are very close and show a decay pattern. This suggests that there might be interference or confusion within the BG's decision-making process. One possible explanation for this behavior is the excessive number of model states recorded in 'mem' leads to overlapping or conflicting representations.

To address this issue, an attempt was made to utilize SPA's heterogeneous memory model. Incorporating the heterogeneous memory model may help manage the states and reduce interference within the BG, providing a potential solution to the problem encountered in Model 2's computation.

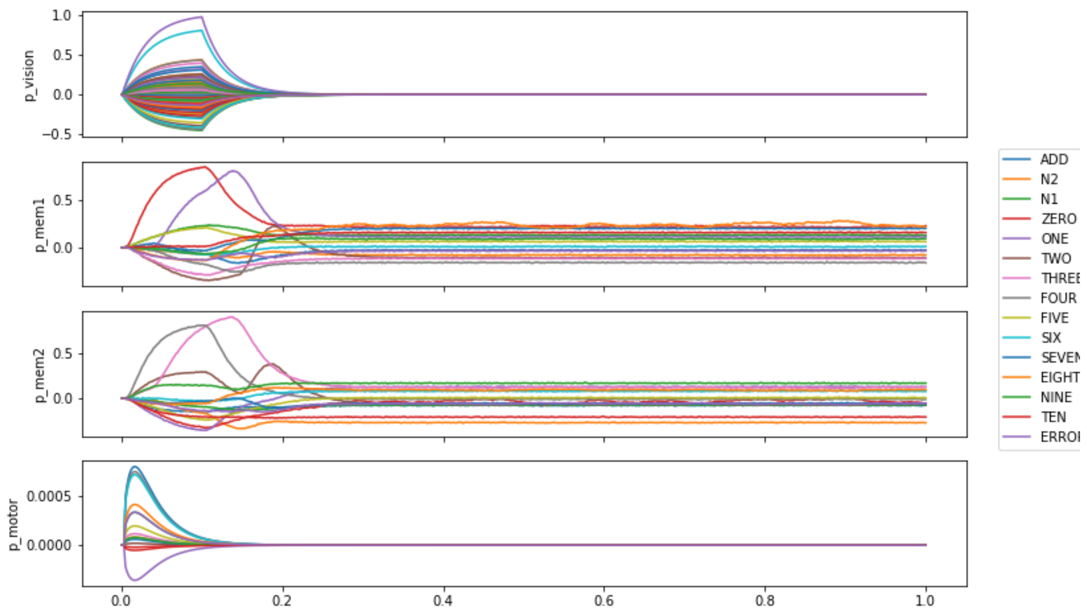

Figure 3.4 : model 2 output graph

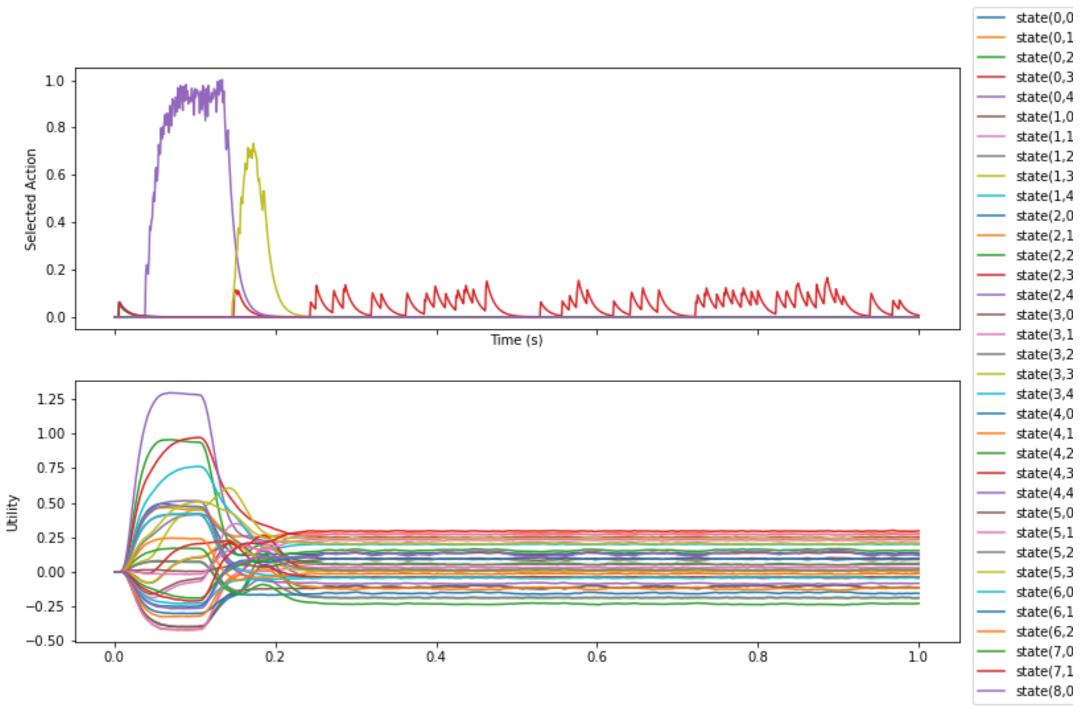

Figure 3.5 : model 2 action selection graph

Model 3

In Model 3, the input is '3+4'. Due to the use of heterogeneous connections, the involvement of 'mem2' is not necessary. However, in practice, the number of rules to be remembered remains the same. The BG's decision-making process still exhibits decay, causing the computation to stop when it reaches '3+2'. As a result, I attempted to address this decay issue in Model 4 by adopting the Winner-Takes-All (WTA) memory model to see if it could resolve the decay problem. By employing the WTA memory model in Model 4, it is hoped that the interference and decay observed in Model 3 could be mitigated, potentially leading to a more stable and accurate decision-making process in the BG.

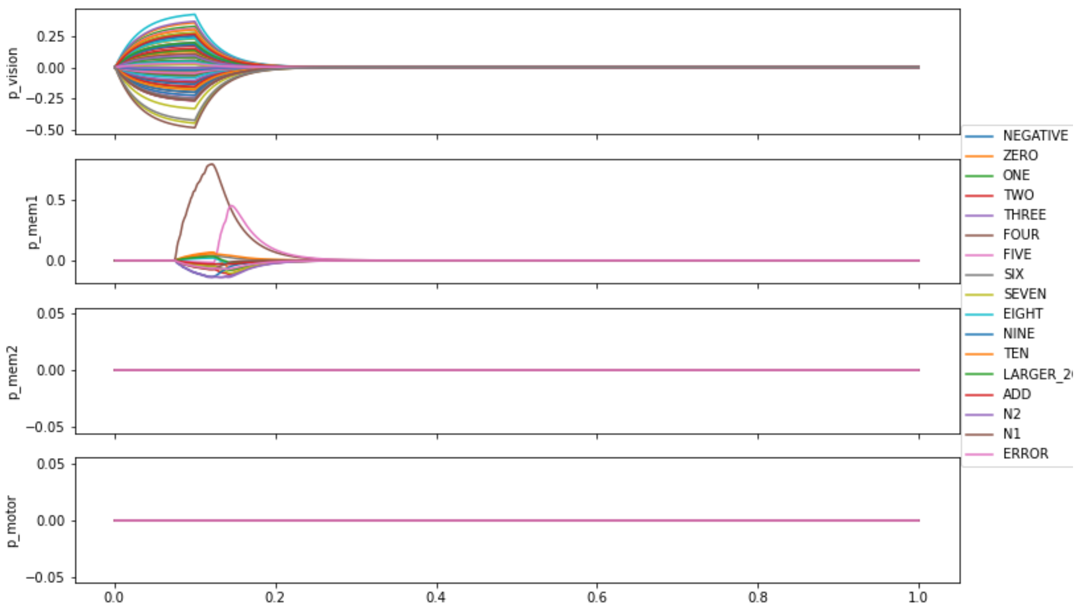

Figure 3.6 : model 3 output graph

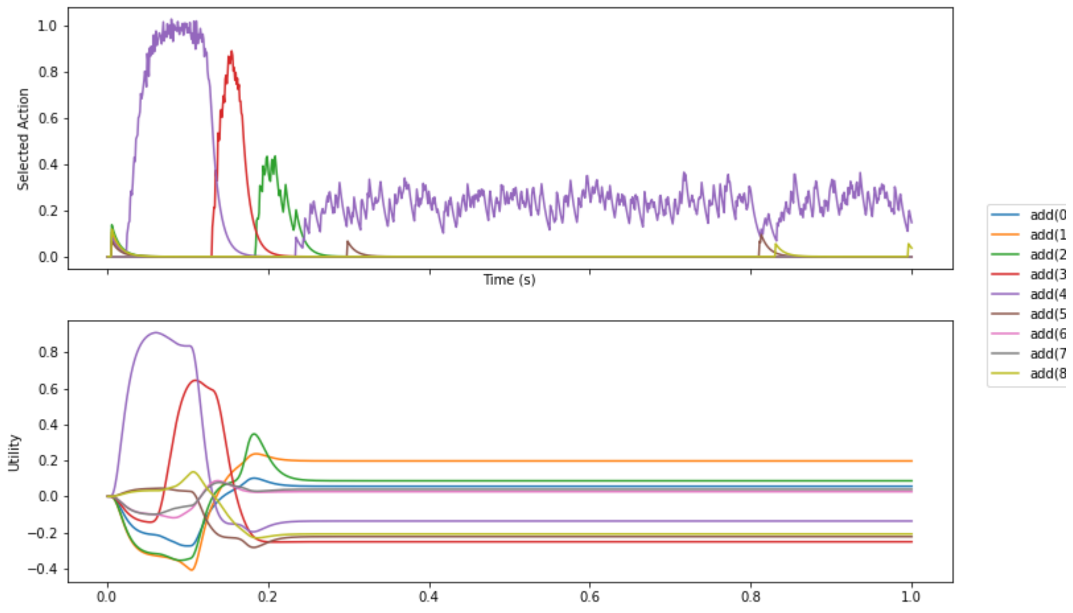

Figure 3.7 : model 3 action selection graph

Model 4

The change to the WTA memory model seems to have brought about some improvements in Model 4. However, certain challenges still persist. One observation is that although the output of 'mem2' remains constant, there are continuous oscillations in the process, especially when reaching the final result. Additionally, 'mem1' still exhibits decay behavior, which could be impacting the overall stability of the computation.

BG's utility graph reveals that many utility values are very close to each other, likely due to the excessive number of states, which is causing difficulty in distinguishing between them. The presence of two states being activated simultaneously at one point during the third step further confuses the decision-making process within the BG.

Regarding the 'motor' module, although it appears to have some output, the values are relatively small, leading to only minor oscillation.

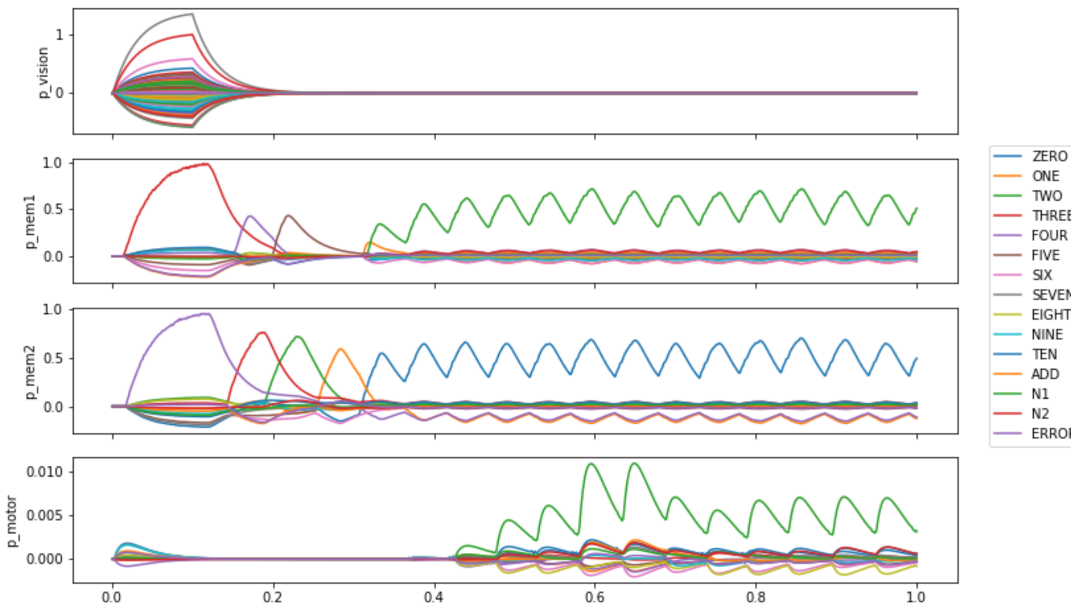

Figure 3.8 : model 4 output graph

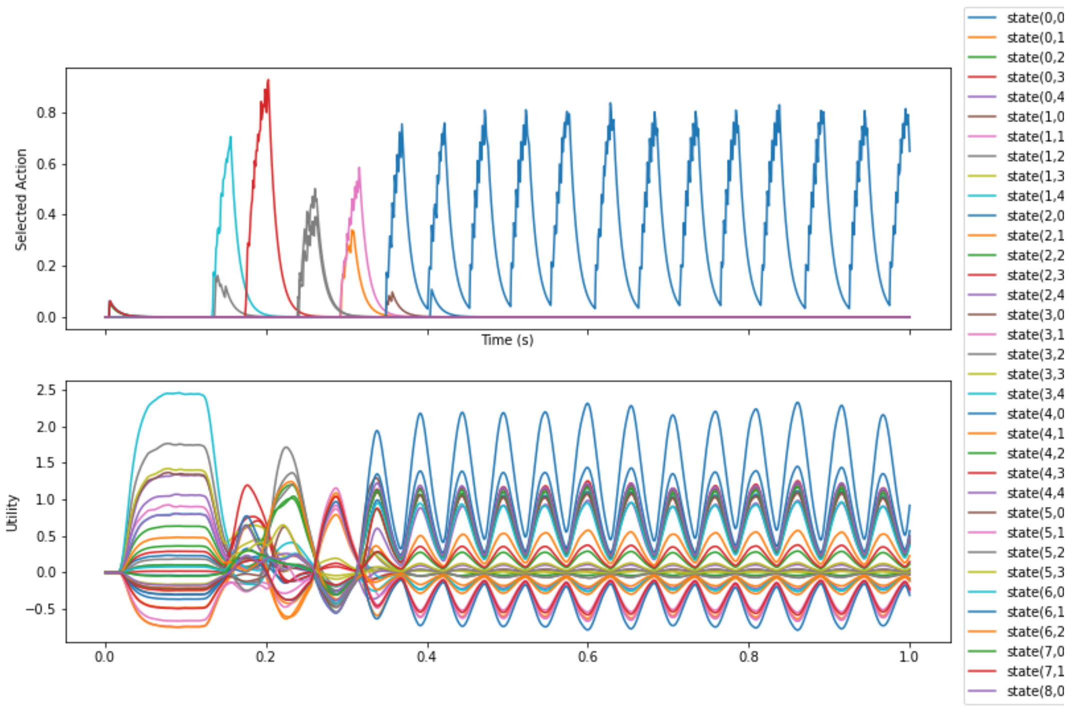

Figure 3.9 : model 4 action selection graph

Model 5

In Model 5, the input remains '3+4'. To address the issues in Model 4, I increased the vector dimension to 128. Model 5 seems to obtain the desired output. The 'mem1' module undergoes four state changes, starting from three and reaching seven, while the 'mem2' module experiences four state changes from four to zero. The 'motor' module also outputs '7' correctly, demonstrating the effective resolution of the addition operation for '3+4' in Model 5.

By increasing the vector dimension to 128, the model seems to have been better equipped to handle the complexity of the computation, leading to improved performance and successful results. In Model 5, it is evident that the counting method requires a significant increase in computational complexity as the numbers to be added increase. As the digits grow, the computational effort required seems to far surpass the growth of the numbers being added. This observation suggests that the counting approach may not be scalable and may encounter challenges in handling larger numbers efficiently.

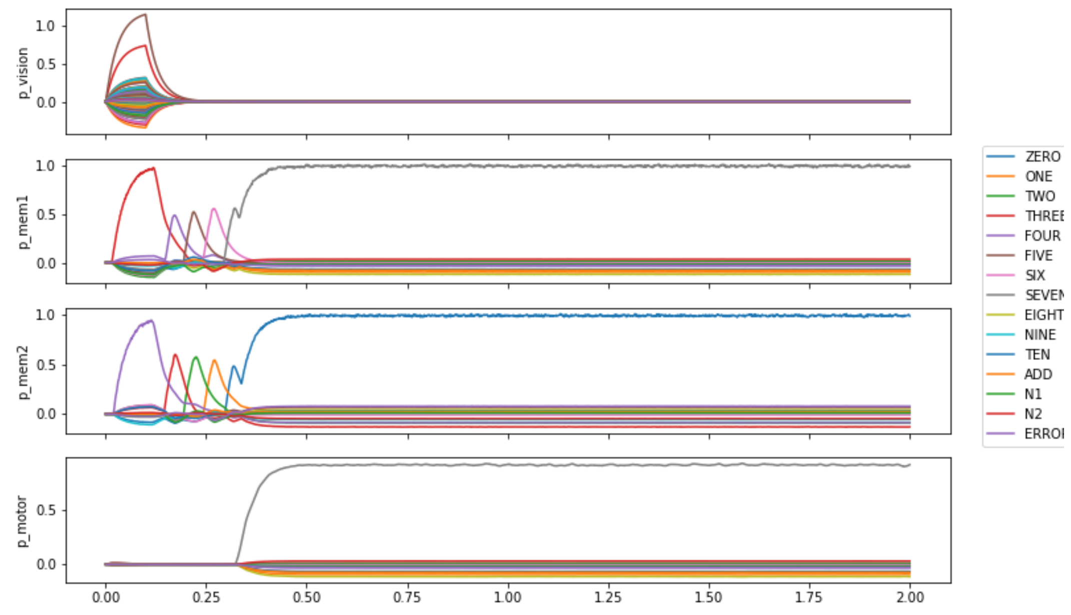

Figure 3.10 : model 5 output graph

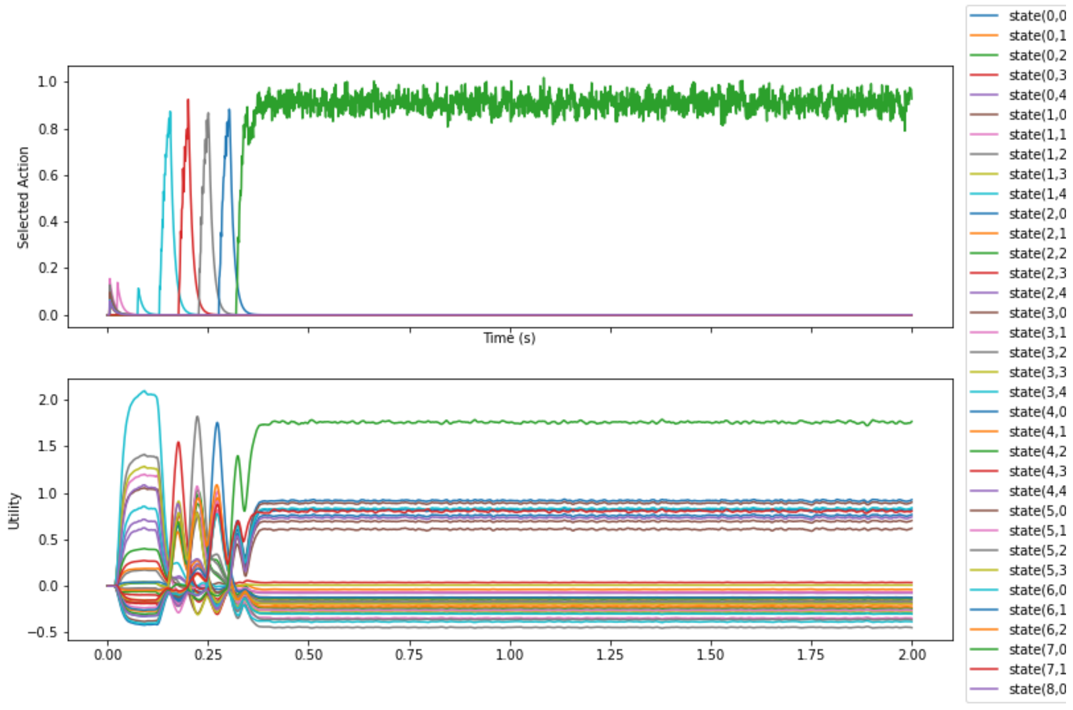

Figure 3.11 : model 5 action selection graph

4 Conclusion

This implementation experience with building the '1+1' model highlights an essential aspect of using SPA for computation. While it was relatively straightforward for simple calculations, it encountered significant challenges when attempting to increase the numbers. As discussed in class, SPA exhibits some advantageous properties, such as information preservation. However, it can lead to exponential growth in representing vectors like this as the structure becomes more complex. If the brain were to perform computations in this manner, the vector used for simplifying neural impulses would still become overly intricate.

This observation suggests that the brain may possess alternative mechanisms to handle computations or encoding in a more efficient and effective manner. Understanding these potential mechanisms can offer valuable insights into how the brain manages complex computations while preserving simplicity and avoiding exponential growth. Exploring and uncovering such mechanisms may contribute to developing more powerful and scalable computational models inspired by the brain's extraordinary capabilities.

5 IMPROVEMENTS AND DIRECTIONS

5.1 Lacking Neurobiological Basis

This implementation lacks a solid physiological basis, as the rules and the models were self-defined, and they may not fully align with the actual neural circuits in the brain. Moreover, the fixed computation rules restrict the model's flexibility, which differs from the brain's ability to adapt and learn from experience.

In reality, the brain has the remarkable capacity to learn and memorize rules through experience and training. It can dynamically adjust its connections and adapt its computations to optimize performance and discover better arithmetic patterns, such as the direct mapping of '4+3=7'.

To address these limitations and create more biologically plausible models, future improvements could focus on incorporating learning mechanisms into the models. This would enable the models to learn and adapt their computation rules based on experience, just like the brain. By allowing the models to learn from data and experience, they can achieve more flexible and adaptive arithmetic computations, moving closer to the brain's capabilities.

Additionally, further research into the neurobiological basis of arithmetic processing in the brain could inform the development of more realistic models.

Studying the neural circuits involved in arithmetic tasks and understanding the principles of computation in the brain could provide valuable insights for designing improved and biologically plausible models.

5.2 Binding with Real Visual Input

The implementation focused solely on high-level cognition, and the actual input messages should be capable of binding with visual input. By incorporating the binding of input messages with visual input, the model can simulate more complex cognitive tasks that closely resemble the SPAUN model's capabilities.

5.3 Improved Mapping

As previously mentioned, the current mapping requires an exponential number of rules to record all states. There may be better algorithms or approaches to achieve computation without such an immense rule set.

Additionally, investigating how the brain's neural circuits handle complex computations could inspire the development of more biologically plausible algorithms. By gaining insights into the brain's computation principles, we can design models that are not only more efficient but also closer to the brain's natural cognitive processes.

References

[1] Stewart, Terrence. Choo, Xuan. Eliasmith, Chris. (2010)"Dynamic Behaviour of a Spiking Model of Action Selection in the Basal Ganglia," *10th International Conference on Cognitive Modeling*

[2] Bear, M. F., Connors, B. W., & Paradiso, M. A. (2016) "Neuroscience: exploring the brain. Fourth edition." *Philadelphia: Wolters Kluwer.*

[3] compneuro.uwaterloo(2020), "SYDE 556: Simulating Neurobiological Systems," URL: `http://compneuro.uwaterloo.ca/courses/syde-750.html`