Neurocomputers

Joanna Matuszak, Joanna Wojciechowicz

1 Introduction

It is believed that new computer that will perform more and more complex tasks should employ principles of the human brain. This kind of computer, called neurocomputer, consists of many interconnected units - so called neurons - performing simple nonlinear transformations in parallel. Its major aim is not general-purpose computation, but pattern recognition using associative memory. There are various types of neurocomputers, including feedforward neural network, Hopfield's network and many more. The most promising are these mimicing rhythmic behaviour of the brain - oscillatory neural networks.

In this project, we introduce the Neurocomputers topic by studing first Hopfields model and later Oscillatory Neural Network model for pattern recognition. In both models we explain all the theory behind the network, perform simulations on different examples and discuss its performance.

2 The Hopfield model [1]

2.1 Introduction to the Hopfield model

The Hopfield network finds inspiration in the intricate connections among biological neurons in the human brain, mimicking nature's efficiency in information processing. Similar to the synaptic communication in biological systems, the Hopfield network utilizes connections to store and retrieve patterns. Hopfield network also provides a model for understanding human memory.

2.2 The Hopfield model dynamics

The Hopfield model consists of N neurons - each labeled by a lower index i, where $1 \leq i \leq N$. Neurons in this model can have only two states. A neuron i is ON if its state variable equals $S_i = +1$ and neuron is OFF if it equals $S_i = -1$. The dynamics evolves in the discrete time with time steps Δt.

Neurons interact with each other with weights w_{ij}. We define the input potential of neuron i, influenced by the activity of other neurons as

$$h_i(t) = \sum_j w_{ij} S_j(t). \tag{1}$$

This input potential of the neuron at time t influences the probability of update of the state variable S_i into specific state in the next time step:

$$P(S_i(t+\Delta t) = +1 | h_i(t)) = g(h_i(t)) = g\left(\sum_j w_{ij} S_j(t)\right), \tag{2}$$

where g - a monotonically increasing gain function with values between 0 and 1. In our simulations we chose commonly used g function:

$$g(h) = \frac{1}{2}[1 + \tanh(\beta h)] \tag{3}$$

with a parameter β. For chosen $\beta \to \infty$, we have

$$g(h) = \begin{cases} 1 \ for \ h \geq 0 \\ 0 \ for \ h \leq 0. \end{cases} \tag{4}$$

Therefore, we can see that in our case in (2) we do not have any randomness, the model is simply deterministic.

The task for Hopfield model is to store and recall M different patterns. Patterns are labeled by the index μ with $1 \leq \mu \leq M$. Each pattern μ is defined as a desired configuration $\{p_i^\mu = \pm 1; \ 1 \leq i \leq N\}$. The network of N neurons correctly represents pattern μ, if the state of all neurons $1 \leq i \leq N$ is $S_i(t) = S_i(t + \Delta t) = p_i^\mu$. In other words, patterns must be fixed points of the dynamics.

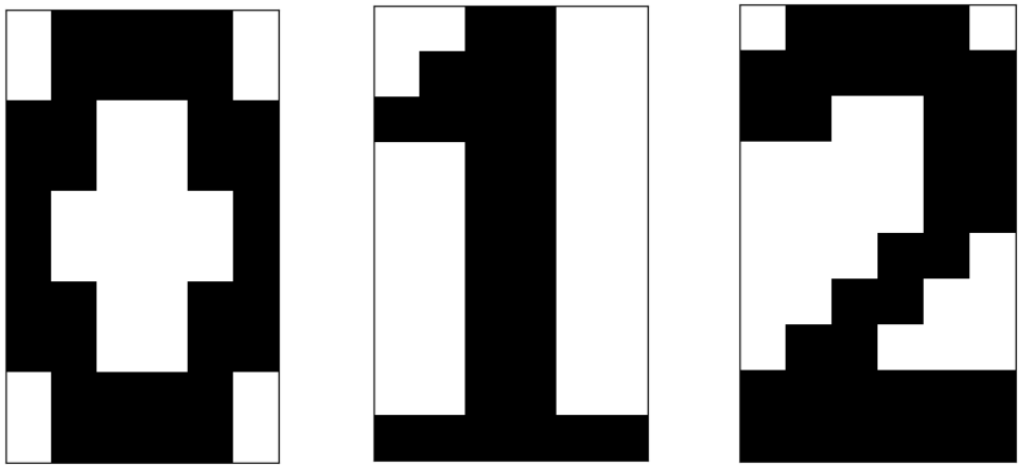

Figure 1: Memorized patterns

At the very beginning of working with Hopfield model for patterns, a random number generator generates, for each pattern μ string of N independent binary numbers $p_i^\mu = \pm 1;\ 1 \leq i \leq N$ with expected value $< p_i^\mu >= 0$. Strings of different patterns are independent. The weights are chosen as

$$w_{ij} = c \sum_{\mu=1}^{M} p_i^\mu p_j^\mu \tag{5}$$

with positive constant $c \geq 0$. The network has full connectivity. The standard choice of the constant c is $c = \frac{1}{N}$.

In order to mimic memory retrieval in the Hopfield model, an input is given by initializing the network state $S(t_0) = S_i(t_0);\ 1 \leq i \leq N$. After initialization, the network evolves freely under dynamics. Ideally the dynamics should converge to a fixed point corresponding to the pattern μ which is most similar to the initial state.

In order to measure the similarity between the current state $S(t) = S_i(t);\ 1 \leq i \leq N$ and a pattern μ, we introduce the overlap

$$m^\mu(t) = \frac{1}{N} \sum_i p_i^\mu S_i(t). \tag{6}$$

The overlap takes a maximum value of 1, if $S_i(t) = p_i^\mu$, i.e., if the pattern is retrieved. It is close to zero if the current state has no correlation with pattern μ. The minimum value $m^\mu(t) = -1$ is achieved if each neuron takes the opposite value to that desired in pattern μ.

We can summarize the Hopfield model dynamics for pattern recognition with a couple steps.

1. Generate M patterns.
2. Initialize the model by giving the random state on the input.
3. Compute the input potential h of each neuron.
4. Compute the probability of each neuron to be ON or OFF - $g(h)$ function (with our chosen h deterministic).
5. Change the state of each neuron.
6. Repeat 3-5 until convergence.

In our Hopfield network all the updates of neurons are done in parallel, but an update scheme where only one neuron is updated per time step is also possible.

2.3 Numerical simulations

In the numerical simulations we consider the case of a network consisting of 60 neurons and three memorized patterns defined on a rectangular 10×6 grid (Figure 1)

We will consider two types of initial states of the network: random initial state and initial state close to one of memorized patterns.

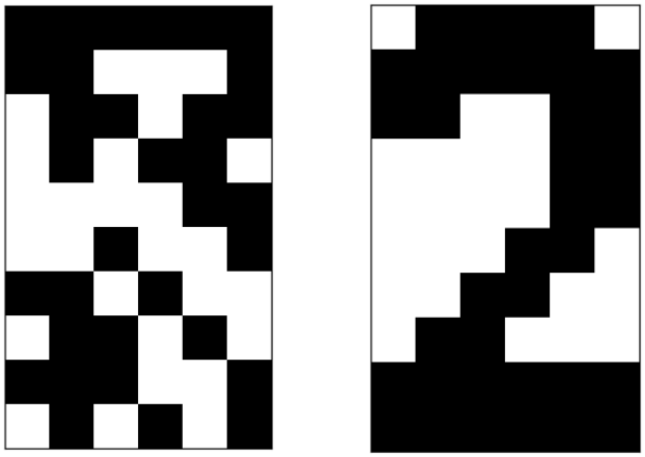

Figure 2: Hopfield model recognition for random initial pattern. Convergence in 1 step to one of the memorized patterns.

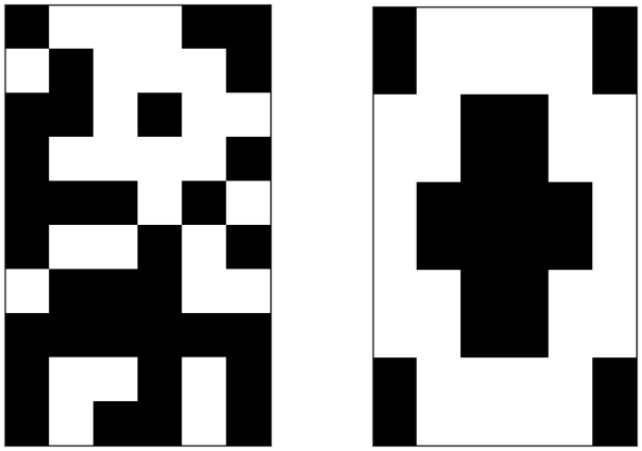

Figure 3: Hopfield model recognition for random initial pattern. Convergence in 1 step to the inverse of one of the memorized patterns.

2.3.1 Random initial state

In the first example of the simulation we see that the model converged to one of the memorized patters, in this case to the image representing the number '2' (Figure 2). However, we observe that convergence to one of the memorized patterns is not guaranteed. For example, in the figure 3 we see that the model converged to the inverse of the memorized pattern, and in the figure 4 we see that the model converged to the pattern that doesn't resemble any of the memorized patterns. In order to study this phenomena we perform 10000 Monte Carlo simulations. The conclusions are following:

- the model always converges, but sometimes to patterns that were not memorized.
- in the case of random initial state in 37.4 % of cases we converged exactly to one of the memorized patterns, 36.1 % to inverted memorized pattern and 26.5 % to different pattern.

2.3.2 Initial state close to one of the memorized patterns

The next example we studied was the case when the input is a pattern close to one of the memorized ones. In order to simulate this situation we choose one of the memorized patterns at random and reverse the state of each neuron with given probability $p = 0.2$. The results are shown in the figure 5. We see that the initial state is close to the pattern '1' and we converged to this pattern in one step. From our observations we conclude that in the case of the initial state close to one of the memorized patterns the model also sometimes returns the pattern that is none of the memorized ones. However, we experience those kind of situations significantly less often. After performing 10000 independent simulations the statistics are following:

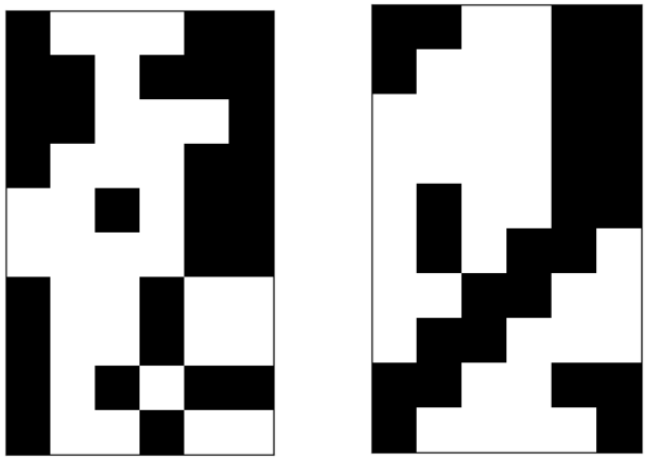

Figure 4: Hopfield model recognition for random initial pattern. Convergence in 1 step to a pattern that was not memorized.

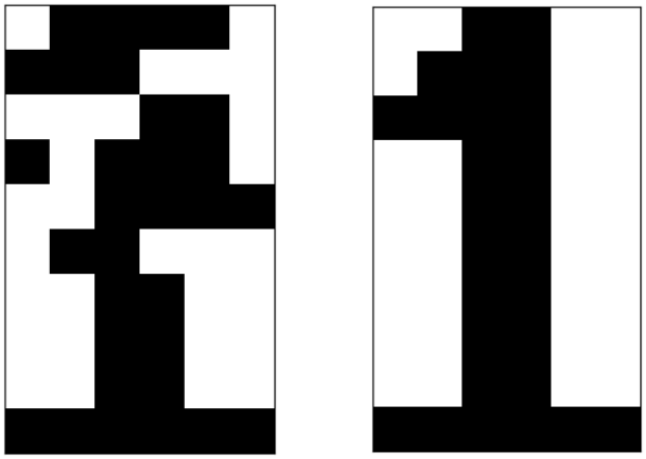

Figure 5: Hopfield model recognition for initial pattern close to one of the memorized patterns. Convergence in 1 step to one of the memorized patterns.

- the model always converges, but sometimes to patterns that were not memorized.
- in the case of the initial state close to one of the memorized patterns in 99.2 % of cases we converged exactly to one of the memorized patterns, 0.1 % to inverted memorized pattern and 0.7 % to different pattern.

2.4 The Hopfield model convergence

In the context of the Hopfield model, convergence to one of the stored patterns is not guaranteed. When the network receives an input, it tries to move towards a stable state, which ideally corresponds to one of the stored patterns. However, it may also converge to a state that is a combination of multiple stored patterns or a state that is not exactly one of the stored patterns. This is known as spurious states or spurious attractors.

Spurious states are patterns $x_s \notin M$, where M is the set of patterns to be memorized. In other words, they correspond to local minima in the energy function. They can be composed of various combinations of the original patterns or simply the negation of any pattern in the original pattern set.

Whether the network converges to one of the stored patterns or to a spurious state depends on various factors including the network architecture, the number and nature of the stored patterns, the initialization of neuron states, and the dynamics of updating.

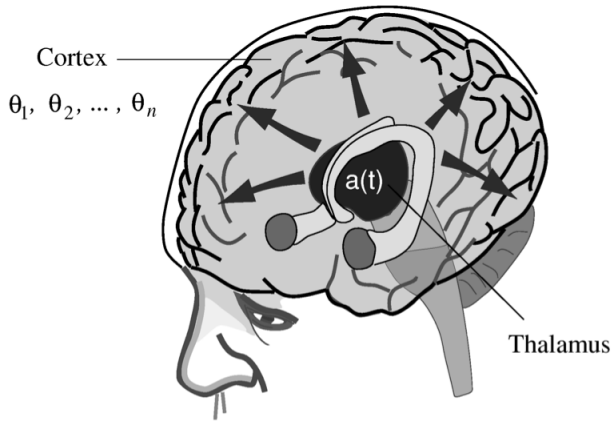

Figure 6: Source: [2].

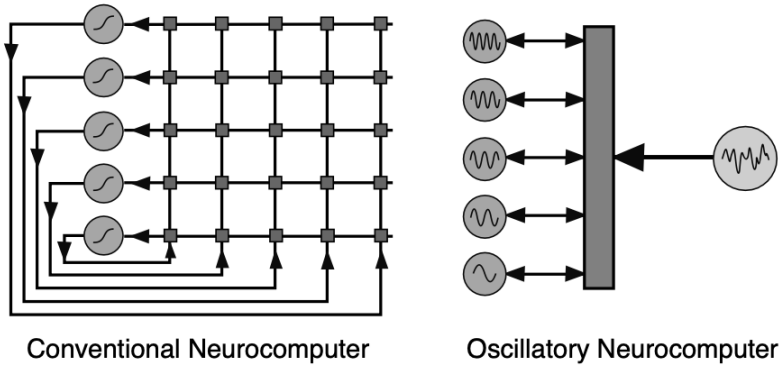

Figure 7: Source: [2].

3 Oscillatory Neural Network [2]

3.1 Introduction to the Oscillatory Neural Network

As mentioned before there are many neural network models that can be used as a theoretical basis for a neurocomputer. The most promising are oscillatory neural networks because they take into account rhythmic behavior of the brain.

3.2 The Oscillatory Neural Network dynamics

Looking at the brain (6), we can see that we treat the cortex as being a network of weakly connected autonomous oscillators forced by the thalamic input. The fact whether these oscillators communicate with each other or not depends on their frequencies.

- If two oscillators have approximately equal frequencies, then they do communicate in the sense that the phase (timing) of one of them is sensitive to the phase of the other.
- When they have different frequencies, their phases uncouple.

It means that the oscillator can interact with other oscillator only if it has appropriate frequency. This suggest specific architecture of the oscillatory neural network model. It consists of oscillators having different frequencies and connected homogeneously and weakly to a common medium. It is worth mentioning that this architecture results in the significant reduction in the number of connections of the network. Looking at the 7 figure, we can see that a conventional neurocomputer having n neurons (circles) would have n^2 connections (squares). An oscillatory neurocomputer with dynamic connectivity imposed by the external input (large circle) needs only n connections: between each neuron (circle) and a common medium (rectangle).

We will ilustrate the model using Kuramoto's phase model:

$$\dot{\vartheta}_i = \omega_i + \epsilon a\,(t) \sum_{j=1}^{n} \sin\,(\vartheta_j - \vartheta_i), \tag{7}$$

where ϑ_i is the phase of the i-th oscillator, $a\,(t)$ is the external input and $\epsilon \ll 1$ is the strength of connection. Let

$$\vartheta_i\,(t) = \Omega_i t + \varphi_i, \tag{8}$$

where Ω_i - natural frequency of i-th oscillator, φ_i - phase deviation of i-th oscillator. Then after inputting equation (8) into Kuramoto model (7) we get

$$\dot{\varphi}_i = \epsilon a\,(t) \sum_{j=1}^{n} \sin\,(\{\Omega_j - \Omega_i\}t + \varphi_j - \varphi_i). \tag{9}$$

Suppose we are using a quasiperiodic external input. We are given a matrix of connections $C = (c_{ij})$. Let

$$a\,(t) = a_0 + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cos\,(\{\Omega_j - \Omega_i\}t) \tag{10}$$

be a time dependent external input, which is a quasiperiodic function of t. If we denote $s_{ij} = \frac{c_{ij}+c_{ji}}{2}$, use the slow time $\tau = \epsilon t$ then we can rewrite system (9) as

$$\varphi_i' = \sum_{j=1}^{n} s_{ij} \sin\,(\varphi_j - \varphi_i). \tag{11}$$

We can see that taken external input can dynamically connect any two oscillators provided that the corresponding c_{ij} is not zero.

As a learning algorithm we introduce the Hebbian learning rule. Suppose we are given a set of m key vectors to be memorized:

$$\xi^k = \left(\xi_1^k, \xi_2^k, \ldots, \xi_n^k\right),\ \xi_i^k = \pm 1,\ k = 1, \ldots, m, \tag{12}$$

where $\xi_i^k = \xi_j^k$ means that the i-th and j-th oscillators are in-phase ($\varphi_i = \varphi_j$), and $\xi_i^k = -\xi_j^k$ means they are antiphase ($\varphi_i = \varphi_j + \pi$). A Hebbian learning rule of the form

$$s_{ij} = \frac{1}{n} \sum_{k=1}^{m} \xi_i^k \xi_j^k \tag{13}$$

is the simplest one among many possible learning algorithms. It suffices to apply the quasiperiodic external input with $c_{ij} = s_{ij}$ for all i and j.

Suppose we are given a vector $\xi^0 \in \mathbb{R}$ to be recognized. Let us apply the external input $a\,(t)$ with $c_{ij} = \xi_i^0 \xi_j^0$ for certain period of time. This results in the phase deviation system of the form

$$\varphi_i' = \sum_{j=1}^{n} \xi_i^0 \xi_j^0 \sin\,(\varphi_j - \varphi_i). \tag{14}$$

We can see that if $\xi_i^0 \xi_j^0 = 1$, then $\varphi_i\,(t) - \varphi_j\,(t) \to 0$, and if $\xi_i^0 \xi_j^0 = -1$, then $\varphi_i\,(t) - \varphi_j\,(t) \to \pi$ for all i and j. When we restore the original input $a\,(t)$, which induces the desired dynamic connectivity, the recognition starts from the input image ξ^0 with added noise, from where we should converge to the desired pattern.

We can summarize the Oscillatory Neural Network pattern recognition with two main steps.

1. Initialize the network. Let ξ^0 be the pattern that has to be recognized. Then, we define a matrix of connections $C = (c_{ij}) = \left(\xi_i^0 \xi_j^0\right)$, and we run the model until convergence.

2. Pattern recognition. Now we use the seed given by the previous step as a initial condition of the network, with some noise. Then, we simulate the system with the matrix s_{ij} defined beofre, until convergence.

3.3 Numerical simulations

Similarly as before, let's consider the case of a network consisting of 60 neurons and three memorized patterns defined as in the figure 1. As an input pattern we choose a pattern that is close to the one of the memorized patterns (Figure 8).

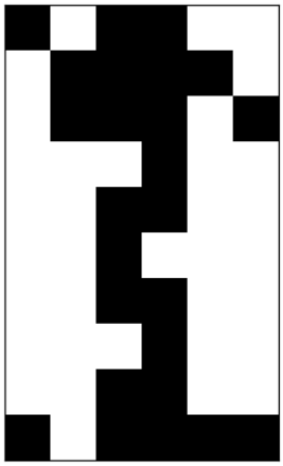

Figure 8: Input pattern for the oscillatory neural network.

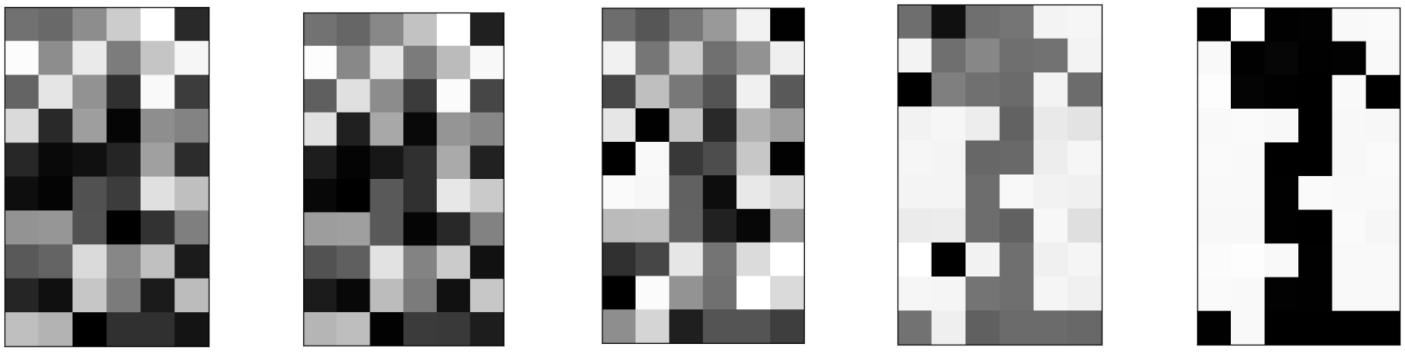

Figure 9: Initialization stage of the simulation. States are shown for t=0, t=2.5, t=5, t=7.5, t=10.

3.3.1 Initialization

As described before, the first task is to tranlate the input image into the initial condition, vector of $\varphi_i(0)$, since our implementation operates on the phase deviations $\varphi_i(t)$. We do that by creating a model that has only one pattern in the memory - the pattern we would like to use as an input pattern. We set a random initial condition and let the model evolve freely under dynamics (14). We see that all initial phase deviations $\varphi_i(0)$ evolved and converged to two values with difference π (Figures 10, 11) and this state corresponds exactly to our input pattern (Figure 9).

3.3.2 Pattern recognition

Having initialized our network we can begin the recognition stage. We take the output of the initialization phase, add gaussian noise to it and introduce such noised pattern to our network. In this stage, our network has three patterns in memory (Figure 1) and evolves under dynamics (11) with learning rule defined as in (13). The evolution can be observed in the Figure 12. We can see that we were able to restore the memorized pattern '1' quite well. We can also see the

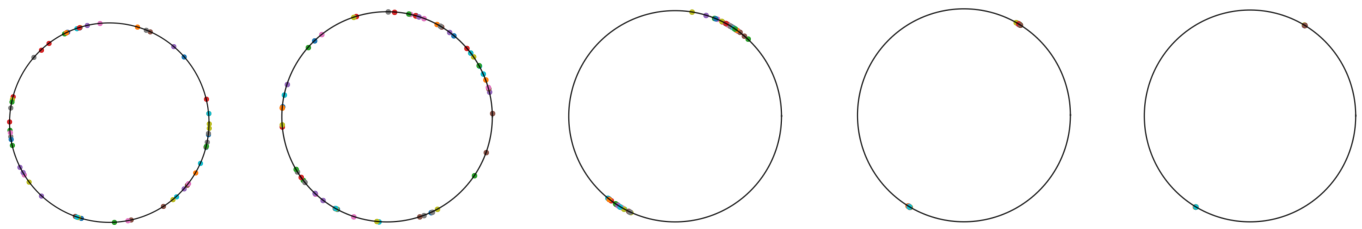

Figure 10: Evolution of $\varphi_i(t)$ during initialization stage of the simulation. States are shown for t=0, t=2.5, t=5, t=7.5, t=10.

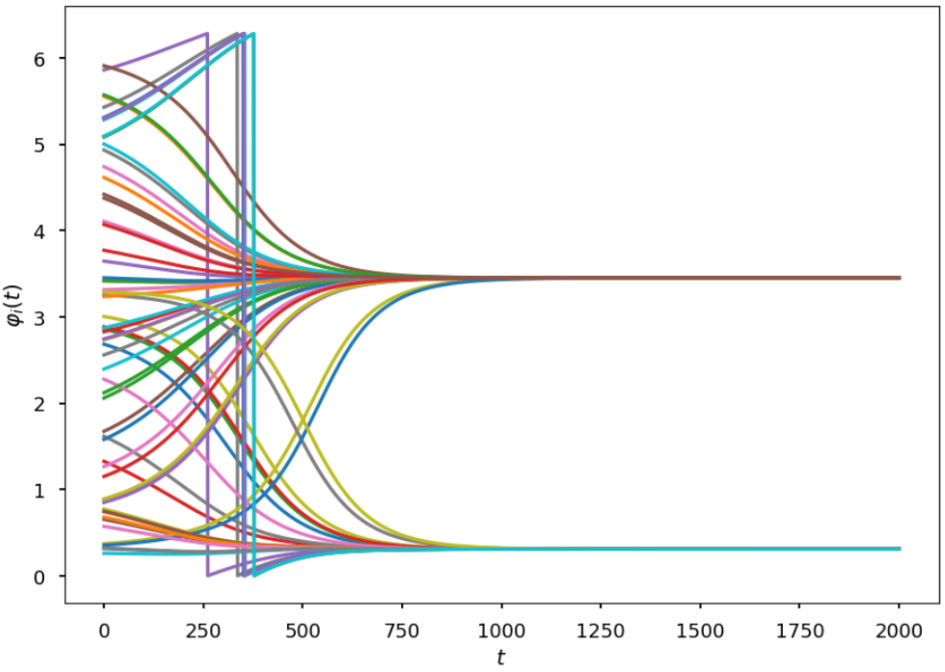

Figure 11: Evolution of $\varphi_i(t)$ during the initial phase.

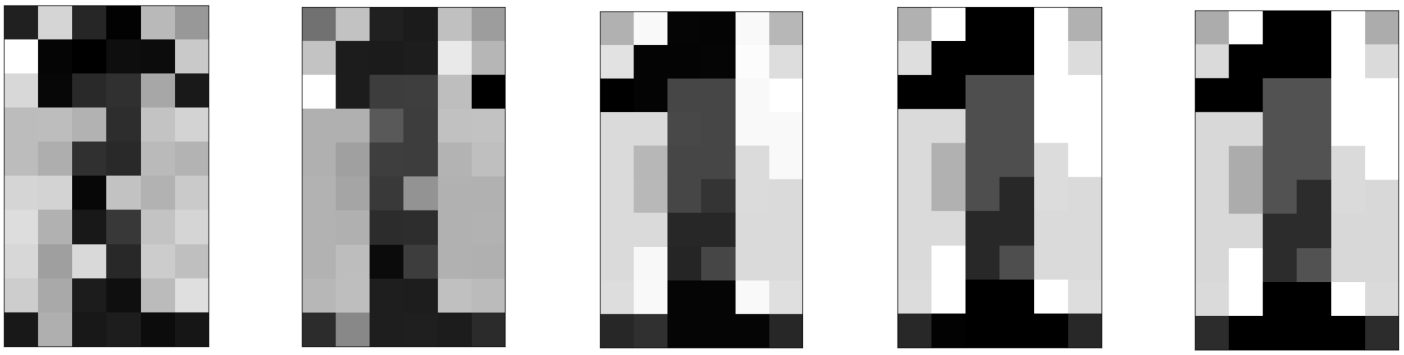

Figure 12: Recognition stage of the simulation. States are shown for t=0, t=2.5, t=5, t=7.5, t=10.

evolution of the $\varphi_i(t)$ in the figures 13 and 14. We observe that we converged into 6 different values of phase deviations. Worth noticing is the fact that we got three pairs of φ, and in each pair the difference between the values is equal to π.

4 Summary

In this project we introduced the topic of Neurocomputers by first introducing the Hopfield model and then focusing on the concept of Oscillatory Neural Network.

In the case of Hopfield model we explained the theory behind the model and then performed simulations in Python. From two examples of simulations - starting from a totally random state and starting from the one of the memorized patterns with noise - we can see that in general if we input in to the network the pattern that is close enough to one of memorized patterns (memorized pattern + noise) it will convergence to the proper pattern.

Afterwards, we moved the focus from Hopfield model to Oscillatory Neural Network. We introduced the theory behind the concept starting from Kuramoto model, with chosen Hebbian learning rule as the learning algorithm. For this model we also performed simlations with Python on the two-stage pattern recognition algorithm.

In both cases we used as patterns real digits. However, it is possible to test the models on easier patterns like 1D sequences of -1 and 1.

The concept of Neurocomputer mimicing the behaviour of the brain and its mechanism of pattern recognition is a very complex topic. Both of presented models introduces different approach to the topic - without or with oscillations - and is interesting science concept especially in today's interest in neural networks.

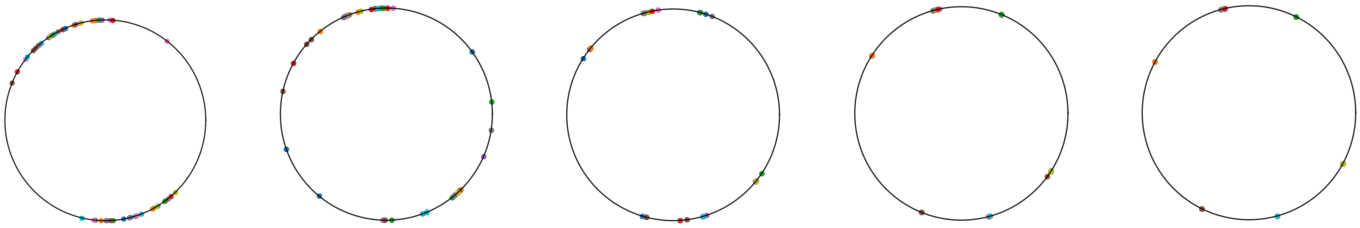

Figure 13: Evolution of $\varphi_i(t)$ during recognition stage of the simulation. States are shown for t=0, t=2.5, t=5, t=7.5, t=10.

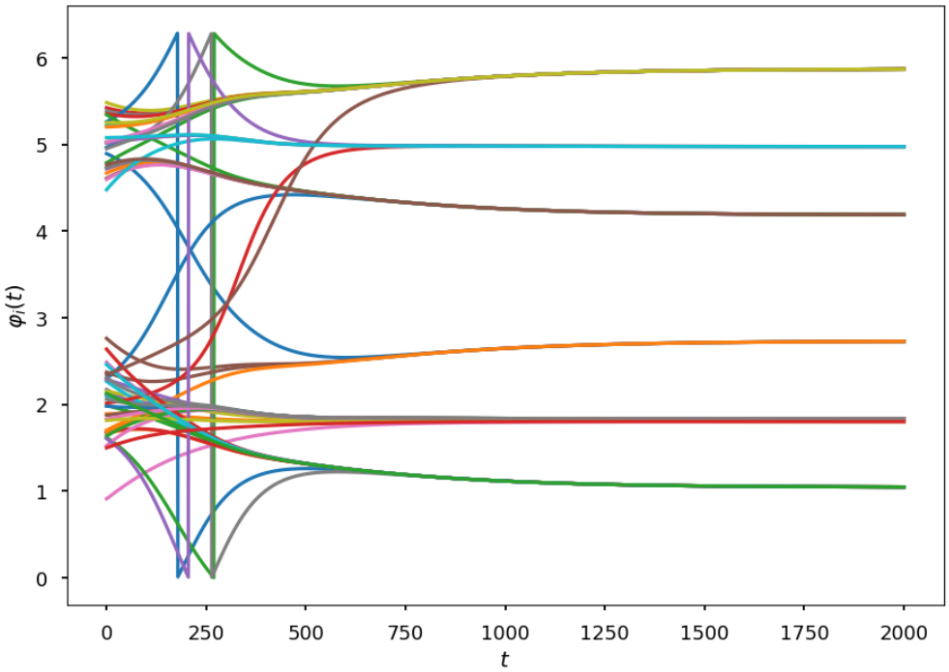

Figure 14: Evolution of $\varphi_i(t)$ during the recognition phase.

References

[1] Gerstner, W., Kistler, W. M., Naud, R., Paninski, L. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[2] Hoppensteadt, F. C., Izhikevich, E. M. Oscillatory neurocomputers with dynamic connectivity. *Physical Review Letters 82*, 14 (1999), 2983.