Humboldt-Universität zu Berlin

Does intrinsic plasticity prefer regularity? A Markov chain Monte Carlo perspective on a self-organizing recurrent neural network

A thesis submitted for the degree of:
Master of Science (M.Sc) in Statistics

submitted by

Carlo Michaelis

born on 29th of September 1989 in Dieburg

First supervisor: Prof. Dr. Vladimir Spokoiny
Second supervisor: Dr. Christian Tetzlaff

The thesis is published under the following license:
Attribution 4.0 International (CC BY 4.0)

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Berlin, 29.01.2018
Carlo Michaelis

Abstract

The self-organizing recurrent neural network (SORN) is a promising approach in modeling brain processes. It extends randomly initialized reservoir networks with binary threshold neurons by three biologically motivated plasticity mechanisms: spike-timing-dependent plasticity, synaptic normalization and intrinsic plasticity. It was already shown that SORN outperforms static reservoir networks and is able to reproduce numerous experimental findings. In particular, recent findings show that spontaneous activity has a special role in this deterministic network. As former approaches used specific input patterns, in this thesis, input patterns were generalized and sampled from systematically chosen Markov chains. Results show that the performance of the network depends on the structure of the Markov chain. If chains contain some less probable and irregular states, their performance is lower and closer to results from static reservoir networks. Finally, it was hypothesized that intrinsic plasticity is the cause for this behavior, due to arising spontaneous activity in less active neurons. It was concluded that SORN does not outperform static reservoir networks in every case and that the performance is not necessarily independent from the chosen input pattern. It remains to be shown if this effect does also occur in natural networks.

Contents

List of Figures 7

List of Tables 9

Abbrevations 10

Nomenclature 11

1 Introduction 12

1.1 Learning and memorization of arbitrary patterns 13

1.2 Structure of the thesis 14

2 Neurons and synapses 16

2.1 Structure of a typical neuron 16

2.2 Action potential of a typical neuron 17

2.3 Structure and behavior of a typical synapse 17

2.4 Inhibitory and excitatory neurons 19

2.5 Classifications of neurons 20

3 Plasticity 22

3.1 Hebb's rule 23

3.2 Spike-timing-dependent plasticity 23

3.3 Homeostatic plasticity 25

3.4 Synaptic normalization 27

3.5 Intrinsic plasticity 27

4 Self-organizing recurrent neural network (SORN) 29

4.1 Artificial neural networks 29

4.2 Classification of ANN 31

4.3 Reservoir Networks 31

4.4 Self-organizing recurrent neural network (SORN) 32

4.5 Properties of SORN 36

5 Markov chain Monte Carlo (MCMC) 39

5.1 Markov chains 39

5.2 Properties of Markov chains 40

5.3 Stationary distribution 42

5.4 Measures 45

5.5 Markov Chain Monte Carlo (MCMC) 48

6 Methods 49

6.1 Network structure 49

6.2 Modification of the intrinsic plasticity 50

6.3 Network training and testing 50

6.4 Classification of states 52

6.5 Measuring performance 57

7 Results 61

7.1 Input dependent performance 61

7.2 IP-hypothesis 74

8 Discussion 85

8.1 Perspectives on SORN 85

8.2 Reverse learning effect 86
8.3 Biological plausibility 87
8.4 Markov chains 89
8.5 Classification 90
8.6 IP mechanism 91
8.7 Conclusion 92

Appendix **94**

A Proof of null recurrent chain 94
B Testing phase with STDP 97
C No-plastic training 99
D Hamming classification 100
E Models close to equal probabilities 103
F IP learning rate 105
G Network connectivity 106

References **108**

List of Figures

1 Structure of a typical neuron 16

2 Structure of a typical synapse 18

3 Morphological classification of neurons 21

4 Spike-timing-dependent plasticity 23

5 Learning windows 24

6 Instability of networks 26

7 Intrinsic plasticity and spontaneous activity 28

8 Spike-timing-dependent plasticity rule used in SORN. 33

9 Sketch of the self-organizing recurrent neural network 34

10 Markov chain properties 41

11 Lorenz curve 47

12 Selection mechanism for activity pattern representations 54

13 First Markov models 61

14 Performance of first Markov models 63

15 Influence of training to first Markov models 65

16 Size of the network and size of the input cluster 67

17 Available steps for comparison 69

18 Markov model series II 70

19 Performance and information of model series II 72

20 Concentration of the stationary distribution 74

21 Input clusters example 75

22 Four Markov model series to test the IP-hypothesis 77

23 Performance of four Markov model series 78

24 Influence of the IP target rate 80

25 Influence of the IP target rate range 82

26 Reverse learning effect and fraction of excitatory-to-excitatory connections . . . 87

27 Complex rule used in oddball paradigm 88

28 Beta distribution 91

29 Null recurrent Markov chain 94

30 Performance with and without STDP in testing phase 97

31 Influence of the length of the no-plastic training phase 99

32 Histograms of minimum hamming distances for 5 models 101

33 Performance results associated with hamming distances 102

34 Model series close to equal probabilities 103

35 Performance of model series close to equal probabilities 104

36 Influence of the IP learning rate 105

37 Variation of connectivity 106

List of Tables

1 State classification using Hamming distance 56

2 p-values of performance differences between Markov models 62

3 Example for a learned transition matrix 64

4 Stationary distributions of model series I 68

5 Stationary distributions of model series II 71

6 p-values of performance differences between $\bar{H}^{\mathrm{IP}}$ values 81

List of abbreviations

ANN	Artificial neural network
ERP	Event-related potential
ESN	Echo State Networks
GLM	Generalized linear model
i.i.d.	Independent and identically distributed
IP	Intrinsic plasticity
iSTDP	Inhibitory spike-timing-dependent plasticity
LSM	Liquid State Machines
LTD	Long-term depression
LTP	Long-term potentiation
MCMC	Markov chain Monte Carlo
MMN	Mismatch negativity
MSE	Mean squared error
RM-SORN	Reward-modulated self-organizing recurrent neural network
RNN	Recurrent neural networks
SN	Synaptic normalization
SORN	Self-organizing recurrent neural network
STD	Short-term depression
STDP	Spike-timing-dependent plasticity
STP	Short-term potentiation

Nomenclature

$\bar{H}^{\mathrm{IP}}$	Average target rate of IP mechanism
$\boldsymbol{v}(t)$	Spiking vector of input neurons
$\boldsymbol{x}(t)$	Spiking vector of excitatory neurons
$\boldsymbol{y}(t)$	Spiking vector of inhibitory neurons
$\boldsymbol{\pi}$	Stationary distribution of a Markov chain
η_{IP}	Learning rate for intrinsic plasticity
η_{STDP}	Learning rate for spike-timing-dependent plasticity
$\hat{M}$	Estimated transition matrix of a Markov chain
$\hat{\boldsymbol{\pi}}$	Estimated stationary distribution of a Markov chain
$\sigma(a_i)$	Activation function
σ^{IP}	Target rate range of IP mechanism
ε_{π}	Estimated mean squared error of stationary distribution
ε_{M}	Estimated mean squared error of transition matrix
M	Transition matrix of a Markov chain
N^E	Number of excitatory neurons
N^I	Number of inhibitory neurons
$T_i(t)$	Threshold of neuron i at time t
T_{x_i}	Number of steps until a Markov state is reached again for the first time
W	Weight matrix
w_{ij}	Weight between neuron i and neuron j

1 Introduction

In order to come closer to an understanding of the the human mind and behavior, many methods emerged in the last decades. In the late 19th century Wilhelm Wundt founded the first psychological institute in Leipzig, based on empirical research, which was the starting point of todays psychology. Some few years later, Waldeyer (1891) published a review about the *neuron doctrine*. He supported Ramón y Cajal and August Forel suggesting that nerve-cells are the basic units of the nervous system and introduced the term *neuron* (Finger, 2001). Like Max Planck was introducing the idea that energy could be quantized in 1900 in "*an act of despair ... I was ready to sacrifice any of my previous convictions about physics.*" (Baddeley, 2008), also the theory of the neuron doctrine required a non-continuous assumption. Similar to Planck, Forel, who introduced the neuron doctrine, was struggling with the idea that the nervous system could be discrete. In his biography he stated:

> It was as though the scales had fallen from my eyes. I asked myself the question: But why do we always look for anastomoses? Could not the mere intimate contact of the protoplasmic processes of the nerve-cells effect the functional connection of nervous conduction just as well as absolute continuity? [...] All the data supported the theory of simple contact. (Finger, 2001)

As well as the quantum theory was the beginning of modern physics, the neuron doctrine built the framework for the neuroscience as we know it today. A significant stage of development was the invention of computers in the 1940s and the fast acceleration in computational power, described by Moore's law (Moore, 1998). Since then, it was possible to use computers for simulations. Theoretical models about the functions of synapses, neurons or networks could be tested and compared to experimental findings. When in the 1980s some few disciplines in this area grew, in 1985 Eric Schwartz was requested to organize a conference bringing researchers together. On top of the conference a book with the title *Computational Neuroscience* was published, collecting contributions from the participants (Schwartz, 1993).

Even if computational neuroscience is an efficient way to verify theories about the nervous system, it still depends on experimental findings. Only if computational models behave like living organisms the results are confirmed. But once a model has proven its reliability, one is able to explore even more properties very cheaply. It is even possible to test specific behavior which is very hard or even impossible to obtain in classical experiments, like resetting the brain activity or including some artificial components.

Most people would probably agree that insights into the function of our mind and behavior, especially learning and memory, is important for our individual life and our society. For example to fight mental disorders and to improve education. One could argue that fighting mental disorders and education is only necessary *because* we started to understand ourself and developed technologies, ending in alienation from our social and natural roots. However, it probably depends on the way the knowledge is used. Tenzin Gyatso (the 14th Dalai Lama) put it in a nutshell, when he said:

> By coming to a better understanding of the workings of the mind, we can learn to tackle our disturbing emotions and mental afflictions, something we can't do with either weapons or money.

1.1 Learning and memorization of arbitrary patterns

Learning and memory are the main subjects of neuroscience, since both of them are highly dynamic and flexible, especially in mammals like the human being. Both are connected, since learning includes storing of information and changing memory is due to learning. On a neuronal level, synapses play a major role in encoding information through long-term plasticity and short-term plasticity. In general, plasticity is the driving force of changes on neuronal level.

In psychology, many different memory and learning types were classified. The main important classification is temporal, and separates between long-term memory, working memory, short-term memory and sensory memory. Learning, on the other hand, can be classified in pre-attentive (unconscious) and attentive (unconscious and conscious), where the former is mostly connected to sensory memory (Schröger, 1997). It is also possible to separate between non-associative learning, where just one stimulus leads to habituation and sensitization, and associative learning, where the connection between two or more stimuli is learned.

In this thesis a neural model, called self-organizing recurrent neural network (SORN), with about 200 neurons is used to learn specific input patterns. Even though it is possible to store information for a longer time in the network, due to implemented synaptic plasticity, in most applications the network is build to store and test information on a sensory or short-term level.

The learned patterns are sampled from a distribution, where the current step depends on the former step. This corresponds to associative learning, since transition probabilities need to be learned between different states representing stimuli. It is also very likely that the behavior of SORN is similar to results from pre-attentive learning.

SORN was developed in 2009 by Lazar, Pipa, and Triesch (2009) with the goal to have a simplified network, which simulates basic synaptic and neuronal mechanisms. At that time there were already a lot of networks in use in machine learning, but they are focused on specific technological applications. Biologically, they are not plausible in most cases. There were already so called *reservoir networks*, but the synaptic connections, the weights, are randomly initialized and do not change while training. The idea came up to combine a reservoir network with plasticity, meaning that the weights (the strength of the synapses) are learned during a training phase. The task was to find a combination of plasticity rules which allow a stable network behavior, running at the edge of chaos (Bertschinger & Natschläger, 2004). In her doctoral thesis, Lazar (2009) discusses the approach in relation to SORN in detail.

In the last years, a lot of studies used SORN and showed that it can reproduce a lot of experimental findings (Lazar et al., 2009; Zheng, Dimitrakakis, & Triesch, 2013; Aswolinskiy & Pipa, 2015; Hartmann, Lazar, Nessler, & Triesch, 2015). After the network has proven its ability in replicating specific biological behavior using certain learning patterns, the question arises if the learning patterns can be generalized. For that purpose Markov chains are used to generate the input patterns.

There are several goals of this research:

- Can SORN learn arbitrary input patterns, modeled as Markov chains?
- Are there differences in the performance between different patterns?
- Whatever behavior will be observed, is it possible to find experimental findings supporting the observations? Or is the observation limited to SORN?
- Is it possible to find an explanation for the behavior of the network?

1.2 Structure of the thesis

In the following sections, first, the basics of neurophysiology, especially plasticity, will be explained. Afterwards, artificial neural networks in general and the specific self-organizing recurrent neural network (SORN) are introduced. The theoretical part closes with an introduction to Markov chain Monte Carlo (MCMC).

In the methods section, more details about SORN are explained, including the properties of the used network, how it is trained and statistics to evaluate outcomes of the network. The methods will directly lead to the results, where in the beginning the main observation is presented, showing that the performance depends on the chosen Markov chain. In the following subsections a so called IP-hypothesis is stated and tested.

Afterwards, results are discussed regarding their biological plausibility and implication. Furthermore, on basis of the findings, further research ideas are presented. In the appendix, extending analysis were done to support some statements from the methods and results sections.

2 Neurons and synapses

A *neuron* is a cell type which occurs in all eumetazoa and therefore in most multicellular organisms. It can process and transmit electrical and chemical signals. So called *synapses* are connections between neurons. Together they form neural networks. The organisms use these networks to process information.

2.1 Structure of a typical neuron

A typical neuron consists of:

- Soma (cell body): The main part of the cell, containing the organelles
- Dendrites: Branched projection of the soma
- Axon: Long slender projection of the soma, which has some branches in the end

The dendrites and the axon are called *neurites*, which are projections from the cell body. The branches of the axon of one neuron are connected to another neurons dendrites. The connection points between the neurons are called synapses. In figure 1 a schematic representation of three neurons is shown where soma, dendrites and axon are indicated. The synapses are highlighted by circles.

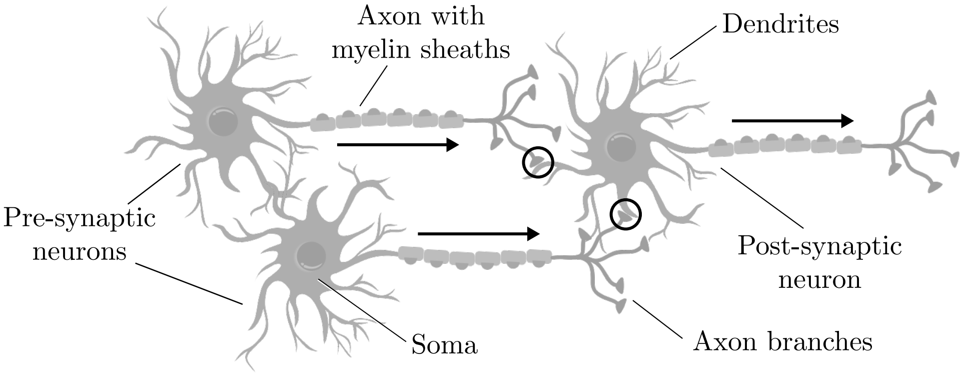


Figure 1: Three neurons showing the typical structure with soma, axon and dendrites. The synapses between presynaptic neurons and the postsynaptic neuron are highlighted with circles. The direction of the conduction is shown by arrows.
Adapted from: *Set Of Neuron Vector.* (2017, December 16). Retrieved from https://www.vecteezy.com/vector-art/115973-set-of-neuron-vector

2.2 Action potential of a typical neuron

According to Deetjen, Speckmann, and Hescheler (2005), a typical cell has a membrane potential of about $-70\,mV$ at its axon hillock, it is called the *resting potential.* If the cell gets some electrical stimulation, which brings the membrane potential up to a *threshold potential* of at least $-55\,mV$, the *depolarisation* of the membrane potential starts, due to changes in the concentration of Na^+ and K^+ ions inside and outside of the cell. This is triggered by changes in the permeability of voltage-gated ion channels in the membrane of the neuron. The potential grows to about $40\,mV$, the *action potential* or *spike.* The action potential spreads over the neuron, mainly along the axon. If the axon is covered with myelin sheaths the propagation of the signal is called *saltatory conduction*, which is faster than conduction without myelin sheaths.

The signals are transmitted nearly binary, where the threshold potential decides if the current activation is enough to transmit a signal or not. That principle is called *all-or-none law.* The height of the threshold can vary, depending on two factors. First, directly after an activation potential was fired from the neuron, there is a short period, where the cell is not excitable at all (*absolute refractory period*), which equals a threshold of infinity. After that, there is a period where the neuron is excitable again, but where the threshold is higher than normal (*relative refractory period*). Both is due to ongoing recovery processes of ions in the cell. Second, the threshold is influenced by the extracellular concentration of ions, where also Ca^{2+} plays an important role (Deetjen et al., 2005).

2.3 Structure and behavior of a typical synapse

The end of the axon or a branch of the axon of one cell (*pre-synaptic cell*) is connected to the dentride of another cell (*post-synaptic cell*). The cell membranes are called *pre-synaptic membrane* and *post-synaptic membrane* respectively. Both are shown in figure 2.

Synapses can occur as a *chemical synapse* or as an *electrical synapse.* Chemical synapses transmit signals via chemical substances (*neurotransmitter*), since the gap between the pre-synaptic and post-synaptic cell is $12\,nm$ to $20\,nm$ (Savtchenko & Rusakov, 2007) and cannot be bridged by electrical activation. The neurotransmitters need some time to reach the post-synaptic cell. Simulations show that it takes about $0.5\,ms$ until 50% of the distributed transmitters receive the post-synaptic neuron, assuming a synaptic gap of $20\,nm$ (Clements, 1996). Electrical synapse

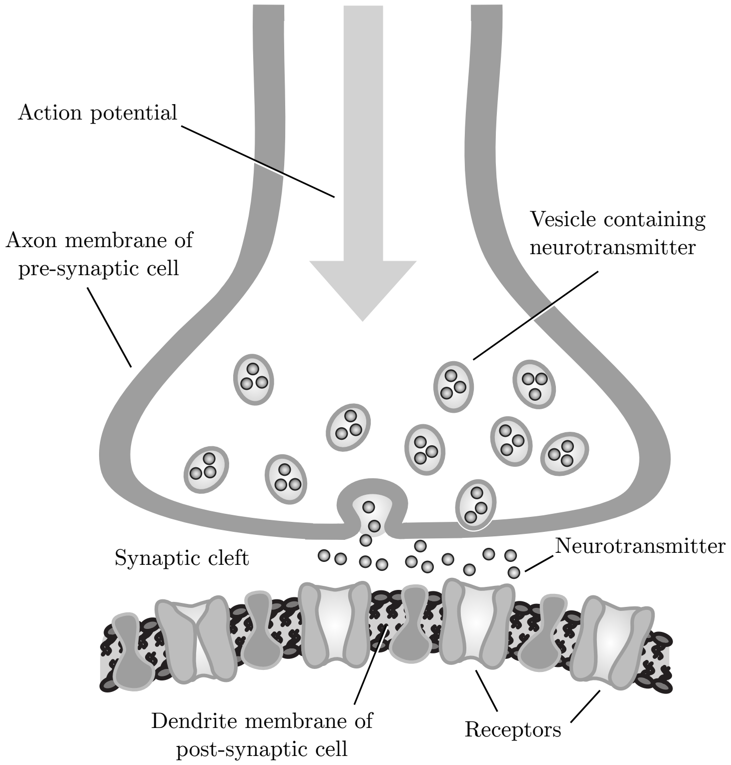

Action potential
Axon membrane of pre-synaptic cell
Vesicle containing neurotransmitter
Synaptic cleft
Neurotransmitter
Dendrite membrane of post-synaptic cell
Receptors

Figure 2: Structure of a typical chemical synapse. Because of an incoming potential, vesicles move to the synaptic cleft and release the containing neurotransmitter. The neurotransmitter will connect to the receptors in the membrane of the post-synaptic neuron. This causes a flow of ions in the post-synaptic membrane, changing its potential.
Adapted from: Schandry (2007, p. 73)

are nearly directly connected with a gap of just $2\,nm - 3\,nm$ (Peracchia, 1977). They are connected by gap junction proteins, called *connexins*, and can transmit electrical signals directly. In those synapses is nearly no delay in transmission compared to chemical synapses.

In chemical synapses the pre-synaptic membrane contains the neurotransmitter, collected in *vesicles*. If a potential arrives at the synapse, the vesicles move to the pre-synaptic membrane. The vesicles merge with the membrane and the neurotransmitters are released into the *synaptic cleft*, the space between the pre-synaptic membrane and the post-synaptic membrane. At the post-synaptic membrane are receptors and ion channels, often they are joined to a *receptor-channel-complex*. If they are combined to such a complex, the receptor is called *ionotropic receptor*. The neurotransmitter binds to the ion channel and changes its structure which allows an exchange of ions. Since the neurotransmitter operates like a key which opens a lock, where the lock is the ion channel, this mechanism is called *key-lock principle*. Since both parts (receptor and ion channel) are in one place, the reaction is fast (Deetjen et al., 2005, p. 40). In the other case, receptor and ion channels are separated and the receptor is called *metabotropic* *receptor*. The neurotransmitter connects to a receptor, but this time the receptor releases a messenger substance called *second messenger*. This second messenger finally opens an ion channel. The effect is slower, since it takes some time until the reaction inside the post-synaptic cell is ready and the ion channels are opened. A simplified scheme of a chemical synapse is shown in figure 2.

2.4 Inhibitory and excitatory neurons

The signals, reaching the neuron via the synapses, can be *inhibitory* or *excitatory*. Inhibitory synapses reduce the potential and decrease the chance to reach the threshold potential that leads to an action potential. They mainly operate with GABA (γ-Aminobutyric acid), Glycine, Serotonin or Dopamin as neurotransmitter. Excitatory synapses rise the potential and therefore increase the chance to start an action potential. They mainly use Acetylcholine or Noradrenalin. Both types of synapses have different chemical mechanisms to change the potential. According to *Dale's principle*, a neuron has either only inhibitory or excitatory axonal synapses (Dale, 1935; Eccles, 1976). There are exceptions from this rule, but in most cases the principle holds (Sossin, Sweet-Cordero, & Scheller, 1990). If a neuron has inhibitory outgoing synapses, it is called an *inhibitory neuron*. In case of excitatory synapses it is called *excitatory neuron*.

The incoming signals from all connected synapses — which can be several thousands (Schandry, 2007, p. 89) — sum up to an overall potential at the soma of the post-synaptic neuron. The summation can be spatial or temporal. *Spatial summation* means that the potential is rising if many different synapses are active at the same time. *Temporal summation* describes a process where at a synapse many signals are coming in a row and sum up to a higher overall potential. The time gap between the signals can be very different, depending on the type of synapse. In some synapse it is necessary, that the signals must be closer than $20\,ms$ to each other to have a summing effect, in some synapses a gap of $1\,s$ can still influence the potential (Schandry, 2007, p. 92).

In many artificial networks the neuron i is modeled with a specific activation a_i at a time, which is linearly dependent from incoming signals. In that case spatial summation can be modeled with

$$a_i = \sum_{j=1}^{n_i} w_{ij} z_j - t_i, \tag{1}$$

where n_i is the number of potential input signals (number of synapses), z_j the activation of the single sources and w_{ij} the weights from the pre-synaptic neuron to the post-synaptic neuron. t_i is the threshold of the post-synaptic neuron.

The transmission of a signal from one neuron to another is modeled with an *activation function* $\sigma(a_i)$ that modulates the intensity and can decide if the neuron is sending a signal or not. To model the binary output of a biological network an indicator function can be used, like the *Heaviside step function*

$$\sigma(a_i) = \Theta(a_i) = \begin{cases} 0, & a_i < 0 \\ 1, & a_i \geq 0 \end{cases}. \tag{2}$$

If the activation $\sum_{j=1}^{n_i} w_{ij} z_j$, coming from the pre-synaptic neurons, is higher than the threshold t_i, the Heaviside function is equal to one and an action potential of the post-synaptic neuron is simulated.

In many cases it is necessary to calculate derivations of the activation function (e.g. for the backpropagation algorithm, which is widely used in machine learning). In that case smooth approximations like the sigmoid function can be used. Statistically the activation function σ is the inverse of the link function in a *generalized linear model* (GLM) and therefore the part that brings non-linearity to the model, since a_i is a linear function. In case of the sigmoid function, the activation function at every neuron is equal to a logistic regression. Further details regarding the mathematical model are given in section 4.1.

2.5 Classifications of neurons

In general, neurons can be classified by their function (Schandry, 2007, p. 39).

- *Sensory neurons*: Receives information from a sensory cell
- *Motor neurons*: The axon is connected to a muscle

- *Interneurons*: The most common type, which receives from and sends information to other neurons

In artificial neural networks the *input* can be seen as sensory neurons in living beings, where the whole network structure is equivalent to the interneurons. The *readout* of a network is analog to the current state of the interneurons. The *output* is similar to the motor neurons which could, for instance in robotics, control some electric motors.

Neurons can also be classified morphologically.

1. *Unipolar neuron*: Has just one neurite, mostly one axon
2. *Bipolar neuron*: Has two neurites, one axon and one dendrite
3. *Multipolar neuron*: Has many neurites, one axon and many dendrites
4. *Anaxonic neuron*: Axon and dendrites cannot be differentiated

All four types are are shown in figure 3. There are many specialized cells like for example purkinje cells, which are large multiplolar interneurons in cerebellum, or pyramidal cells, which are multipolar interneurons in many areas of the brain. Pyramidal cells, for example, are the main source for measurements using electroencephalography (Kirschstein & Köhling, 2009).

In many artificial networks, like the one which is used in this thesis, the input is designed with a neuron which has just an axon with some projections, the output has just dendrites. Inbetween are multipolar or rarely bipolar neurons, which form the characterizing part of the artificial neural network. As well as in the brains of living beings, the most common neuron in artificial neural networks is modeled with the multipolar neuron.

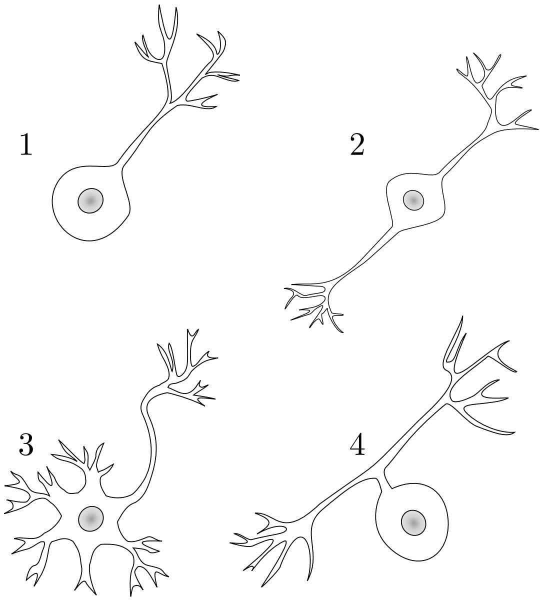


Figure 3: Morphological classification of neurons: (1) Unipolar neuron, (2) Bipolar neuron, (3) Multipolar neuron and (4) Anaxonic neuron.
Adapted from: *Neuron.* (2017, December 16). Retrieved from https://en.wikipedia.org/wiki/Neuron

3 Plasticity

Plasticity or more specific *neuroplasticity*, when it comes to brains, is the ability of the brain to change its neuronal structure depending on experience. It can be separated in *functional plasticity* and *structural plasticity*.

- Functional plasticity: Changes in synaptic transmission characteristics due to changes in amount of neurotransmitter or receptors
- Structural plasticity: Changes in structure of the neuron, like growing or shrinking of axons, dendrites, synapses or even whole neurons

Furthermore, *synaptic plasticity* describes all processes related to synapses, like changes in transmission characteristics and growing or shrinking of synapses. Regarding synaptic plasticity one can distinguish between *pre-synaptic plasticity* and *post-synaptic plasticity*. Pre-synaptic plasticity means, that the amount of released neurotransmitters or the time for readmission of neurotransmitters changes. Post-synaptic plasticity changes the sensibility for a given amount of neurotransmitters. For example the amount of receptors (ion channels) can be adapted. Furthermore, plasticity can vary in duration. *Short-term plasticity* changes the transmission characteristic of the synapses in a scale of milliseconds to minutes, while *long-term plasticity* has an effect for hours, or even years. Both can be separated in so called depression — short-term depression (STD) and long-term depression (LTD) — which means a decrease in connectivity, and potentiation — short-term potentiation (STP) and long-term potentiation (LTP) — which means an increase in connectivity between the synapses.

Second messengers (see above in section 2.3) play an important role in synaptic plasticity (Skrebitskii & Chepkova, 1999). Metabotropic receptors can not only start mechanisms to open ion channels, they can also start processes to change the behavior of receptors, channels or neurotransmitters.

In artificial neural networks the synaptic transmission characteristic can be modeled as a weight w_{ij} between neuron i and neuron j, as already used in equation (1). If there is no connection between two neurons, it holds that $w_{ij} = 0$. Functional plasticity corresponds to a change in $w_{ij} \in (0, 1]$, indicated as Δw_{ij}, whereby structural plasticity sets the weight to zero or non-zero.

3.1 Hebb's rule

A rule that simplifies a functional plasticity mechanism is the *Hebb rule.* Biologically, synapses strengthen their connection if the firing of neuron j and the firing of neuron i is close together. In that case the weight w_{ij} will increase.

For artificial neural networks holds therefore

$$\Delta w_{ij} = \eta \cdot z_j \cdot a_i, \tag{3}$$

where η is a learning rate. A high learning rate results in fast but inaccurate convergence, while a low learning rate results in slow but accurate convergence.

3.2 Spike-timing-dependent plasticity

The *spike-timing-dependent plasticity* (STDP) is a causal type of Hebb's rule, depending on temporal correlations between the spikes of pre-synaptic and post-synaptic neurons (Sjöström & Gerstner, 2010). Spikes arriving repeatedly at the pre-synapses, some time before or after an action potential at the post-synaptic neuron, lead to long-term plasticity. In case that spikes arrive shortly before an action potential of the post-synaptic neuron, it leads to long-term

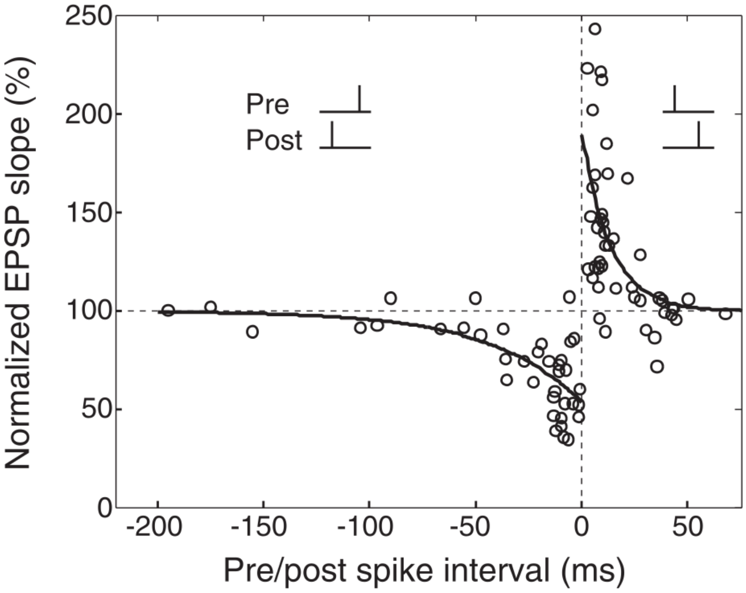


Figure 4: Spike-timing-dependent plasticity (Dan & Poo, 2006, figure 1). The closer the post-synaptic neuron spikes before the pre-synaptic neuron spikes, the higher will be the *decrease* in weight and vice versa.

potentiation. In case that spikes arrive shortly after an action potential of the post-synaptic neuron, it leads to long-term depression. The time scale where plasticity occurs, varies between different types of synapses. In most cases it is of the order of milliseconds (Dan & Poo, 2006). An illustration of the STDP is shown in figure 4.

A mathematically formulation of the STDP was done by Sjöström and Gerstner (2010) and more specific by Kempter, Gerstner, and Van Hemmen (1999). Consider a post-synaptic neuron with activity $x_i(t)$ and a pre-synaptic neuron with activity $x_j(t)$. Assume many spikes at the post-synaptic neuron, counted by $l \in \{1, ..., n_l\}$, where n_l is the total number of spikes. t_i^l indicates the time for every spike at neuron i with $x_i(t_i^l) = 1$. Furthermore, the number of spikes of the pre-synaptic neurons is counted by $k \in \{1, ..., n_k\}$, where n_k is the total number of spikes. t_j^k indicates the points in time at which the pre-synaptic neuron is firing with $x_j(t_j^k) = 1$.

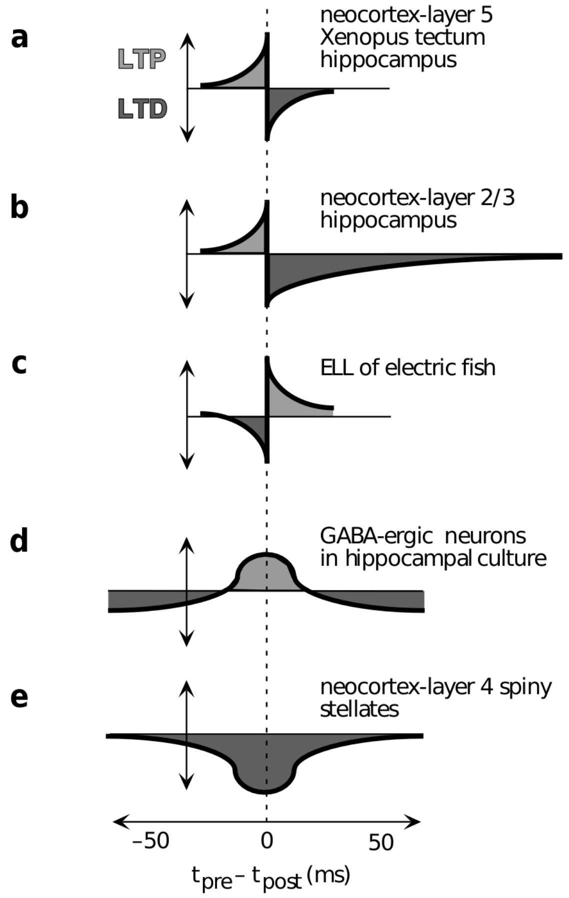


Figure 5: STDP learning windows (Abbott & Nelson, 2000, figure 2).

The *learning window* is a function $f(\Delta t)$, where $\Delta t = t_j^k - t_i^l$. It can be defined in various ways, some are shown in figure 5. Using the learning window, the change of the weight can be defined as

$$\Delta w_{ij} = \sum_{k=1}^{n_k} \sum_{l=1}^{n_l} f(\Delta t) = \sum_{k=1}^{n_k} \sum_{l=1}^{n_l} f(t_j^k - t_i^l). \tag{4}$$

$f(\Delta t)$ is often chosen as an exponential function (Sjöström & Gerstner, 2010). It could be shown, that an exponential learning window is able to model experimental findings (Zhang, Tao, Holt, Harris, & Poo, 1998). It can be defined as

$$f(\Delta t) = \begin{cases} A_+ \exp(-\Delta t/\tau_+), & x > 0 \\ -A_- \exp(\Delta t/\tau_-), & x < 0 \end{cases}, \tag{5}$$

where A_+ and A_- are parameters to choose. τ_+ and τ_- are time constants and often chosen as $\tau_+ = \tau_- = 10\,ms$.

If, for the exponential learning window, t_j^k and t_i^l are close to each other most of the time, which means that pre- and post-synaptic neurons spiking in succession, there will be a change in the weight between those neurons, depending on the sign. The more often those neurons fire close to each other, the higher will be the sum and the change of the weight. The principle of the exponential learning window is shown in figure 5a.

For simulations of neural networks which are focused on more high level network based research, STDP is often modeled more simple. One simplified version, where the weight only depends on the spiking between two discrete time steps, will be introduced in section 4.4.

3.3 Homeostatic plasticity

A network just processing Hebb's rule would soon end in a destabilized state. If LTP becomes dominant, the synapses in the network will increase unbounded. If LTD becomes dominant, the synapses will decrease to zero at some point. Hence, the synaptic activity in the network is diverging and therefore needs constraints (Miller & MacKay, 1994; Abbott & Nelson, 2000). In figure 6 the stability problem is shown for a simplified example. A small feed forward network with 5 layers is only able to propagate an input signal if the gain is close to 1. If it is higher than 1, the activity 'explodes' and the sinusoidal input signal cannot be represented in the last layer. Otherwise, if the gain is lower than 1, the signal 'dies out' and is not able to reach the last layer. The same holds for recurrent networks, where the problem of propagation happens in time, compared to propagation in space regarding the layers in a feed forward structure (G. G. Turrigiano & Nelson, 2004). In machine learning this problem is known as the *exploding and vanishing gradient problem* (Pascanu, Mikolov, & Bengio, 2013).

STDP can allow some more stability than Hebb's rule alone (Abbott & Nelson, 2000), depending

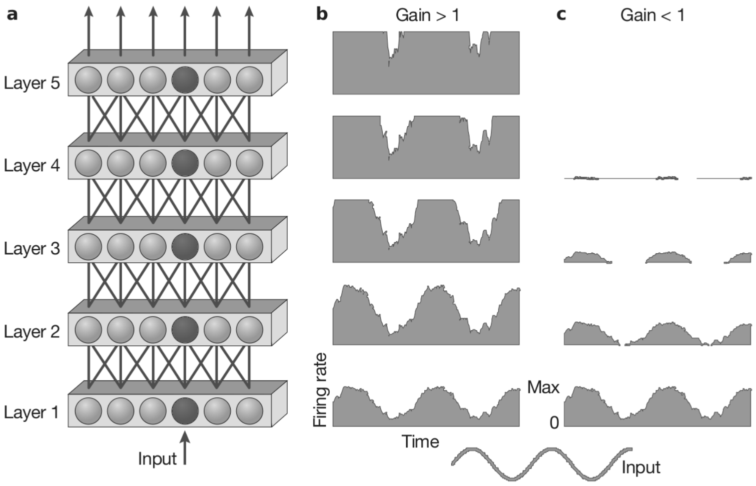


Figure 6: Instability of networks for a simplified case (G. G. Turrigiano & Nelson, 2004, figure 1). **a)** A small feed forward network with 5 layers, the activity of the dark neuron is shown in b and c. **b)** If the gain is higher than 1, the activity 'explodes' and the sinusoidal input signal cannot be represented in the last layer. **c)** If the gain is lower than 1, the signal 'dies out' and is not able to reach the last layer.

on the learning window, which leads to more precise LTP and LTD. The ability, how much STDP is able to stabilize the network, depends on the parameters of $f(\Delta t)$.

However, biologically, one can observe a whole set of different constraints which appear in natural neural networks in order to maintain a specific target firing rate. They are reviewed for example by G. G. Turrigiano and Nelson (2004), G. G. Turrigiano (2008), Pozo and Goda (2010) and G. Turrigiano (2012) and summarized at Scholarpedia by Williams, O'Leary, and Marder (2013). The main goal of those processes is to keep a specific activity in the network. It acts as a set point or baseline for the overall activity. If some areas in the network are going up in activity for some time, the networks reacts with a re-normalization of the whole activity, where the relative connections between the neurons stay the same. It is important to note that those mechanisms operate in a decentralized way, where every neuron adjusts for itself. There cannot be any centralized power, controlling the overall activity. This kind of plasticity is called *homeostatic plasticity.* There are several different complex mechanisms of homeostatic plasticity and some of them are still not fully understood. However, those mechanisms can be classified by:

- Type: Intrinsic (affecting the neuron) or synaptic (pre-synaptic and post-synaptic)
- Spatial: More global (all synapses connected to a neuron) and more local or quasi-local (single synapse or small groups of synapses connected to a neuron)
- Temporal: Fast (seconds to hours) and slow (hours to days)

3.4 Synaptic normalization

One important class of synaptic homeostatic plasticity is *synaptic scaling*, where incoming synaptic connections to a neuron are normalized to a target weight. While adjusting the weights, their relative strengths are maintained. G. G. Turrigiano (2008) summarizes that neurons use calcium-dependent sensors to detect changes in their own firing rates. Depending on those changes the number of receptors at the synapses will be regulated. Note that the more local the mechanisms are and the fewer synapses are involved, the more the synaptic scaling mechanism will interact with LTD and LTP mechanisms, like STDP.

While the normalizing process is highly complex in natural networks, in artificial systems it is relatively simple to perform. For example, in a simplified more global rule, every weight connected to a neuron i is just normalized with all incoming weights from neurons j, using

$$w_{ij}(t) \leftarrow \frac{w_{ij}(t)}{\sum_{j=1}^{N} w_{ij}(t)}, \tag{6}$$

where $w_{ij}(t)$ is the weight between neuron i and j at time t. Due to the mathematical process of normalization, it is also called *synaptic normalization*.

3.5 Intrinsic plasticity

Intrinsic plasticity (IP) is a process, which is local and intrinsic, since it only affects the conduction inside of a single neuron. In IP no synapse and therefore, when modeling artificial networks, no weight is affected. There are several different types of intrinsic plasticity and many of them are not fully understood (Cudmore & Desai, 2008). In principal, the neuron regulates biophysical properties of ion channels in the membrane. Awatramani, Price, and Trussell (2005) showed that a small depolarization of about $10\,ms$ (resulting in a potential between $-60\,mV$ and $-65\,mV$), induced by a change of the extracellular concentration of calcium ions, enhances the probability of transmitter release up to 2-fold.

A simple implementation in artificial neural networks affects the threshold for conducting the action potential inside of the neuron. If the neuron was active, the threshold will increase for the next step. If the neuron was not active, the threshold will decrease, to make the neuron more sensible. Note that if the threshold of a neuron becomes negative, the neuron fires spontaneously. The effect is shown in figure 7. The rule can be formalized with

$$T_i(t+1) = T_i(t) + \eta_{\text{IP}} \left(x_i(t) - H_{\text{IP}} \right), \qquad (7)$$

where η_{IP} is the learning rate and $x_i(t) \in \{0, 1\}$ the activity of neuron i at time t. H_{IP} is the *target rate*. All neurons will in average be active with a rate of H_{IP}. The choice of H_{IP} is not trivial, since not all choices of the target rate will necessarily lead to a network with good performance (Lazar et al., 2009).

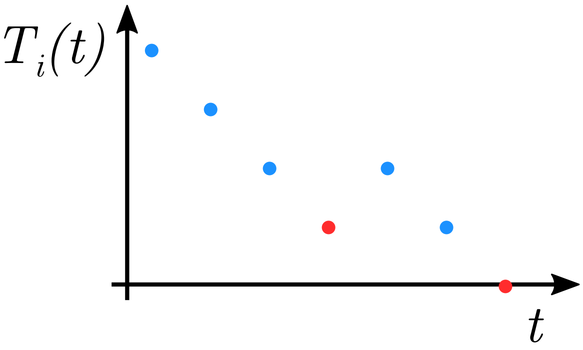


Figure 7: Simple implementation of intrinsic plasticity. If the neuron is not spiking (blue dot), the threshold is decreased in the next step of time. If the neuron is spiking (red dot), the threshold is increased in the next step of time. If the threshold moves below zero, the neuron fires spontaneously.

4 Self-organizing recurrent neural network (SORN)

4.1 Artificial neural networks

Two different scientific approaches motivate an *artificial neural network* (ANN). Originally those networks were invented to simulate processes in the brain to get a better understanding of how the brain stores and processes information and how recognition and behavior is influenced by neural activity. Especially Rosenblatt (1958) formulated a first idea to simulate a simple neural system, the so called *perceptron*. On the other side, the idea came up to construct classification and prediction algorithms using ANN. Even though the approach was promising, for a long time artificial networks were not useful, because networks are computational costly and need big datasets to perform well. But when Hinton, Osindero, and Teh (2006) published a fast way to train ANN and computers became more powerful at the same time, ANN became a widely used approach in machine learning. In machine learning *supervised* ANN, where the algorithms learns with a given ‘truth’, are still inspired by biology, but in most cases very different from real neural networks. On the other hand, computational neuroscience uses mostly *unsupervised* networks, since there is no evidence that humans learn from an absolute truth. Those networks mostly try to be as close as possible to real neural processes or try to understand specific properties of biological networks. Nevertheless, in computational neuroscience the used networks highly depend on the level of research. For research on cell level, many details of neural conduction and synapses behaviors are modeled. Otherwise, if models on higher levels with hundreds or thousands of neurons are build, e.g. modeling memory and learning, many biological details on cell level are omitted in order to reduce complexity.

Independent of the approach, the basic principle for modeling neurons is the perceptron. Statistically the perceptron can be understood as a generalized linear model (GLM), classifying the incoming signals either directly as 0 or 1 or giving a probability for those classes.

Beginning with a regression model, some inputs $\boldsymbol{x} \in \mathbb{R}^d$ are chosen to describe some output $y \in \mathbb{R}$, using a *regression function* $f(\boldsymbol{x})$. The regression is of the form

$$y = f(\boldsymbol{x}) + \epsilon. \tag{8}$$

Instead of the direct data $\boldsymbol{x}$ it is also possible to use a transformation $\phi(\boldsymbol{x})$, which is called a *feature*. For modeling a neural network, data $\boldsymbol{x}$ is used directly. $\epsilon \in \mathbb{R}$ is a stochastic error term, a random variable, and can be distributed in several ways. In most applications it is assumed to be normally i.i.d. distributed with $\epsilon \sim N(0, 1)$. The regression function $f(\boldsymbol{x})$ can be arbitrary. In a linear case it has the form

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\beta}, \tag{9}$$

where $\boldsymbol{\beta} \in \mathbb{R}^d$ are the parameters of dimension d. While the estimator of a linear model is relatively easy to obtain, an estimator for a non-linear $f(\boldsymbol{x})$ is not easy to find. In order to have the opportunity to use non-linear models, in GLMs a so called link function $g^{-1}(f(\boldsymbol{x})) = g^{-1}(\boldsymbol{x}^T \boldsymbol{\beta})$ is introduced. It brings non-linearity to a linear model. If the expected value of y, modeled with the link function, is taken, it results in

$$\mu = \mathbb{E}[y] = \mathbb{E}[g^{-1}(\boldsymbol{x}^T \boldsymbol{\beta}) + \epsilon] = g^{-1}(\boldsymbol{x}^T \boldsymbol{\beta}), \tag{10}$$

since $g^{-1}(\boldsymbol{x}^T \boldsymbol{\beta})$ is deterministic and $\mathbb{E}[\epsilon] = 0$. At that point, the derived statistical model can be mapped to an ANN use case. $\boldsymbol{x}$ can be identified with the input signals from all pre-synaptic neurons $\boldsymbol{z}$ from equation (1), where the first input is a dummy and is set to one. Furthermore, it can be seen that $\boldsymbol{\beta}$ is equal to the weights $\boldsymbol{w}_i$, where the first weight is $-b_i$. It means that $\sum_{j=1}^{n_i} w_{ij} z_j - b_i = \boldsymbol{x}_j^T \boldsymbol{\beta}_j$. Finally, the inverse of the link function $g^{-1} = \sigma$ is the *activation function*, introduced in section 2.4, and $o = \mu$ denotes the output at the post-synaptic neuron. The output is given by

$$\mu = g^{-1}(\boldsymbol{x}^T \boldsymbol{\beta}) = g^{-1}\left(\sum_{j=1}^{n_i} w_{ij} z_j - b_i\right) \overset{(1)}{=} \sigma(a_i) = o. \tag{11}$$

As already discussed in section 2.4, the choice of the activation function depends on the requirement. In machine learning, classically, a continuous function like a sigmoid or, more recent, piecewise rectified linear units (ReLUs) are used, since derivations are often necessary. In that case, there is always some activation, which rises strongly, if the input neurons increase in activity. Statistically, the perceptron with a sigmoid function is a logistic regression.

In case of an indicator function, an all-or-none law is realized, which is a closer approximation to real neurons. It results in zero activity, if the input neurons sum to a value below the threshold and an activity of one, if the sum of the activity hits the threshold. Such an indicator function was introduced in equation (2).

So far, the perceptron model is able to simulate some pre-synaptic neurons and one post-synaptic neuron, which depends on the activity of the pre-synaptic neurons and the activation function. If multiple perceptrons are modeled, the output of one neuron can be an input for another neuron and so forth. Mathematically, a neural network with multiple neurons is a recursive generalized linear model. It can be modeled in one direction with a *feed forward* structure or in arbitrary directions with a *recurrent* structure.

4.2 Classification of ANN

Over time, a large number of network types were developed. Besides many different architectural types, two important types will be mentioned here:

- *Feed-forward neural networks* are basically multilayer perceptrons, where synapses are just built in one direction. The flow of information is going from input neurons to output neurons with hidden layers inbetween.
- *Recurrent neural networks* (RNN) have neurons which have synapses not only reaching into the next layer, but also to a neuron in next time step of processing. Therefore, time-coded information can be included.

Another possible classification refers to the number of connections. A *fully connected* network has all connections established, and therefore weights of non-zero between all neurons. A *sparsely connected* network has just some few connections, whereas most connections are fixed to zero. In some networks, like Echo State Networks (see below), the connectivity is in an order of about 1%.

4.3 Reservoir Networks

A specific case of RNN is *Reservoir Computing* (Lukoševičius & Jaeger, 2009). Those networks can be called reservoir networks. Input signals are fed into a sparsely and randomly connected network, called *reservoir*. The reservoir behaves like a complex nonlinear dynamic filter that transforms the incoming signals using a high-dimensional temporal mapping (Schrauwen, Verstraeten, & Van Campenhout, 2007). Schrauwen et al. described similarities between reservoir computing and kernel methods. Kernel methods are often used in machine learning to transform the input into a multi-dimensional or even infinite-dimensional feature space, where linear separation or linear fitting performs better than in the original input space. Similar to kernels, also a reservoir can be seen as a transformation of the input into higher dimensions, where the readout is kind of a feature space. If the reservoir contains N neurons, the input is projected into a N-dimensional space. A state of the reservoir is mathematically just a point in this high dimensional space. Temporal signals entering the reservoir can be described as a trajectory in that space. Lazar (2009) hypothesized in her dissertation that, theoretically, the brain computes with trajectories in a similar way, which would justify the use of reservoir networks in neural computation approaches.

Two major types of Reservoir Networks are Echo State Networks (ESN) (Jaeger & Haas, 2004) and Liquid State Machines (LSM) (Maass, Natschläger, & Markram, 2002). Both networks were independently developed (Goodfellow, Bengio, & Courville, 2017) and are quite similar. LSM uses neurons with binary outputs. They are designed to explain processes in the brain, whereas ESN uses continuous hidden units and focuses on machine learning tasks. Under the condition of the *separation property* (different inputs result in separable outputs) and the *fading memory property* (information about recent input can be maintained) the LSM fulfills the Stone-Weierstrass theorem, namely that the LSM can approximate any continuous function (Maass & Markram, 2004). Therefore LSM is an interesting starting point for reservoir based neural simulation.

4.4 Self-organizing recurrent neural network (SORN)

LSM are developed for supervised learning and parameters are trained in a way to minimize the error. To simulate brain dynamics it is necessary to find a more unsupervised way of learning. Furthermore, those networks have normally a static reservoir, which means that the weights are randomly initialized and fixed in further processing. To tackle both problems, plasticity rules are a promising approach. They could be used as described in section 3. There were already some additions to Liquid State Machines, like adding hebbian learning (Norton & Ventura, 2006), but a first sophisticated reservoir network with plasticity was proposed by Lazar et al. (2009), called a *self-organizing recurrent neural network* (SORN). It uses three plasticity rules in a reservoir network with binary threshold neurons, namely:

- Spike-timing-dependent plasticity (STDP)
- Synaptic normalization (SN)
- Intrinsic plasticity (IP)

Plasticity rules The synaptic normalization and intrinsic plasticity are realized according to equation (6) and equation (7) respectively. Regarding the IP, recall that it is the source for spontaneous activity. SORN is build such that noise is optional and in most applications the network is using spontaneous activity from IP only. In consequence the system can be described as deterministic chaos, if the initial weights are given. Only if noise is included, it would truly behave stochastically. The only remaining stochastic part is the random initialization of the weights.

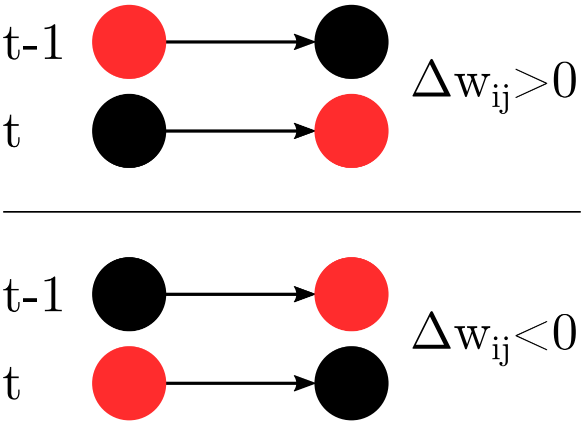


Figure 8: Spike-timing-dependent plasticity rule used in SORN. The black circle corresponds to a silent neuron, the red circle represents a spiking neuron. The arrow indicates the direction of the weight. If the pre-synaptic neuron is firing, before the post-synaptic neuron fires, the weight between them is increased. Otherwise, the weight is decreased.

The spike-timing-dependent plasticity can be derived from equation (4). In SORN only the last step in time and thus only one spike is considered. If the pre-synaptic neuron fires directly before the post-synaptic neuron, the weight is increased. Otherwise, the weight is decreased, as shown in figure 8. Hence, there is no summing necessary and the rule is simplified to

$$\Delta w_{ij}(t) = f(\Delta t) = f(t_j - t_i), \tag{12}$$

where t_j is the time when the pre-synaptic neuron j spikes and t_i when the post-synaptic neuron

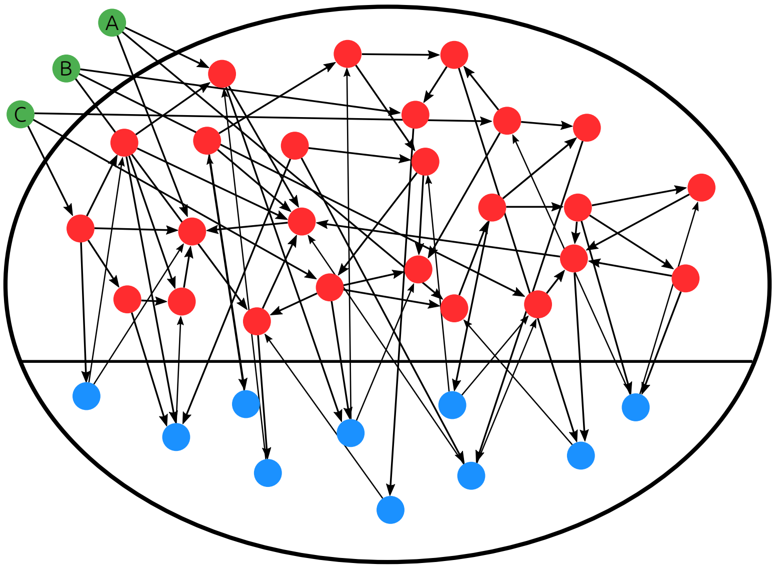


Figure 9: Sketch of the self-organizing recurrent neural network. Red circles are excitatory neurons, blue circles represent inhibitory neurons. Connections between excitatory neurons are sparse, inhibitory and excitatory neurons are fully connected in both directions. Input neurons are represented by green circles. They are connected to excitatory neurons.

i spikes. The learning window is defined as

$$f(\Delta t) = \eta_{\mathrm{STDP}} \cdot \begin{cases} x_j(t_i - 1)x_i(t_i) & t_j \leq t_i \\ -x_j(t_j)x_i(t_j - 1) & t_j > t_i \end{cases}, \tag{13}$$

where the learning rate η_{STDP} was introduced. Note that $f(\Delta t)$ is zero, if both neurons are not active in a row. If t denotes the absolute time of the network, the weight change can be simplified to

$$\Delta w_{ij}(t) = \eta_{\mathrm{STDP}} \cdot (x_j(t-1)x_i(t) - x_j(t)x_i(t-1)), \tag{14}$$

since $x_i(t), x_j(t) \in \{0, 1\}$. Equation (14) corresponds to Lazar et al. (2009, equation 5).

What kind of plasticity rule is used, depends on the phase of training and testing. All the rules, which are included in a phase are applied after every discrete calculation step.

Network setup N^U is the number of input neurons and N^E the number of excitatory neurons. Excitatory and inhibitory neurons were separated. The number of inhibitory neurons is set as a fraction of excitatory neurons, for example $N^I = 0.2 \cdot N^E$. All neurons have a weight of $w_{ij} \in [0,1]$. Weights are established between excitatory and excitatory neurons, modeled in a matrix W^{EE}, between inhibitory and excitatory neurons W^{EI}, excitatory and inhibitory neurons W^{IE} and input neurons and excitatory neurons W^{EU}. The matrices W^{EI} and W^{IE} are fully connected, W^{EE} is sparsely connected.

There are no connections between inhibitory and inhibitory neurons in the model. While those connections exist in real biological networks, they occur sparsely and their function is highly specialized (Chamberland & Topolnik, 2012). Important are connections between excitatory and inhibitory neurons and vice versa. Those connections provide feedback loops, which help to stabilize the system (G. G. Turrigiano & Nelson, 2004). Finally, the target rate for the IP was chosen as $H_{\mathrm{IP}} = 2 \cdot N^U / N^E$.

The structure of the network is sketched in figure 9.

Updating the network The general model, defined in equation (11), can be specified for the case of a SORN network. $\boldsymbol{x}(t) \in \{0,1\}^{N^E}$ denotes the spiking state of the excitatory neurons and $\boldsymbol{v}(t) \in \{0,1\}^{N^U}$ represents the input neurons. In the next step of time, the state of the excitatory neurons is given by

$$x_i(t+1) = \Theta\left(\sum_{j=1}^{N^E} W_{ij}^{EE}(t) x_j(t) - \sum_{k=1}^{N^I} W_{ik}^{EI} y_k(t) + v_i(t) - T_i^E(t)\right). \tag{15}$$

Note that $v_i(t) = \sum_{l=1}^{N^U} W_{il}^{EU} v_l(t)$. But since every excitatory neuron has not more than one connection to an input neuron, the notation is simplified by $v_i(t) \in \{0,1\}$. The threshold $T_i^E(t)$ depends on intrinsic plasticity, which was defined in equation (7).

$\boldsymbol{y}(t) \in \{0,1\}^{N^I}$ denotes the spiking state of the inhibitory neurons. They are updated in a more simple way, using

$$y_i(t+1) = \Theta\left(\sum_{j=1}^{N^E} W_{ij}^{IE} x_j(t) - T_i^I\right). \tag{16}$$

The threshold T_i^I for the inhibitory neurons is not updated using IP. Also note that STDP is applied to $W_{ij}^{EE}(t)$, but not to the inhibitory-to-excitatory weights, neither to the excitatory-to-inhibitory weights. W_{ik}^{EI} and W_{ik}^{IE} are not time-dependent. Consequently, also synaptic normalization is only applied to $W_{ij}^{EE}(t)$.

If input is given to the network, the network is trained. Without input, meaning that $v_i(t) = 0$ for all $i \in \{1, ..., N^E\}$, the network is left alone with spontaneous activity only. The presented equations can also be found in Lazar et al. (2009), where most of the notation is very similar. Further details regarding the readout and the training are also given in section 6.

4.5 Properties of SORN

Since SORN was invented in 2009, many studies were done with this network. Results from four papers will be introduced, which show the performance of the network on one hand and the implications regarding the general understanding of dynamic biological networks on the other hand.

In the original paper from Lazar et al. (2009), the SORN network was able to outperform static reservoir networks in several tasks. The network was tested in an ‘counting’ and an ‘occluder’ task. They were able to show that in both cases the performance of the SORN was better than in networks where the weights were just initially set. Lazar et al. (2009) argues that especially STDP has an important role. There is a lot evidence that STDP-like mechanisms are essential for stimuli encoding in the brain. In several experiments it was possible to show that repetitive stimulation with temporal patterned inputs causes a rapid reorganization, which bases on STDP mechanisms (Yao & Dan, 2001; Fu et al., 2002; Yao, Shi, Han, Gao, & Dan, 2007).

Zheng et al. (2013) extended the SORN by two more plasticity rules. First, *structural plasticity* was added in order to allow new synaptic connections between excitatory neurons. With a probability of $p_c = 0.1$ a new connection was established between a random pair of neurons with a weight of $w_{ij} = 0.001$. Since STDP tends to silence some specific connections to a weight close to zero, the structural plasticity enabled the network to balance this effect. Second, *inhibitory spike-timing-dependent plasticity* (iSTDP) added STDP from inhibitory to excitatory neurons. With this adapted SORN network, it was possible to reproduce experimental findings from Yasumatsu, Matsuzaki, Miyazaki, Noguchi, and Kasai (2008). The distribution of synaptic weight strengths in the SORN matched the lognormal distribution of EPSPs in the visual cortex of rats.

An extension to a SORN network is a *reward-modulated self-organizing recurrent neural network* (RM-SORN), which modulates the weight-changes in STDP (Aswolinskiy & Pipa, 2015). Equation (14) gets an additional *modulation factor* m, therefore

$$\Delta w_{ij}(t) = m \cdot \eta_{\mathrm{STDP}} \cdot \left(x_j(t-1)x_i(t) - x_j(t)x_i(t-1) \right). \tag{17}$$

Different modulation strategies can be used. The weight-change can be applied as normal, suppressed or inversed, depending on a favored output or readout. Within eight tasks Aswolinskiy and Pipa (2015) could show that RM-SORN is close to the performance of the original SORN from Lazar et al. (2009), even though the RM-SORN learns more self-organized than the SORN. It shows the ability of the SORN network to learn even more complex tasks very well in a highly self-organized way.

Another huge research effort was done by Hartmann et al. (2015). First, they were showing that SORN has many characteristics which can be observed in experimental findings. The following list contains four such findings, where the experimental studies are cited in brackets.

- The inter-spike-interval (ISI) distribution of a neuron can be fitted by an exponential decay. Furthermore, the coefficient of variation (CV) follows a Poisson distribution with an event rate around 1 (Kenet, Bibitchkov, Tsodyks, Grinvald, & Arieli, 2003; Goris, Movshon, & Simoncelli, 2014).
- The fraction of excitatory-to-excitatory connections, where the weight stays $w_{ij} > 0$ during plasticity, converges to a stable fraction (Yasumatsu et al., 2008).
- Sequences of four distinct activity patterns can be learned. In the original experiment with mice, gratings were used to train the visual cortex V1 (Gavornik & Bear, 2014).
- Those sequences can be predicted, using the stored information of the network (Gavornik & Bear, 2014).

The sequence task, where specific patterns or ‘words’ were learned, was also part of the inspiration for the present research. The words were represented surprisingly stable after a phase of plastic training, using STDP.

In a very interesting approach Hartmann et al. (2015) gave ambiguous input to the network, after it was trained. While the network was trained with several pre-defined input clusters, in the ambiguous case, some neurons from one cluster and some neurons from another cluster were activated. With a Bayesian approach, using the initial training information as a prior and the ambiguous input as data, they could predict the decision of the network by computing the posterior. They finally concluded, that SORN is able to store information from earlier input and that this previous stored information, which is often seen as noise, could be information which systematically influences decisions of networks instead.

5 Markov chain Monte Carlo (MCMC)

In this thesis, Markov chains were used to train the presented self-organizing recurrent neural network. Before the specific methods are addressed, some theory about Markov chains is introduced.

5.1 Markov chains

Markov chains contain a number of states which are changing with a specific probability. The *markov property* states that the probability of the current state only depends on the immediate previous state. Therefore, the Markov property in the discrete case can be defined as follows:

Definition 1 (Markov property). *Let* $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space and* $(S, \mathcal{S})$ *a measurable space, where* $S = \{x_1, x_2, ...\}$ *is a countable set, called state space. A stochastic process* $X = \{X_t : \Omega \to S\}_{t \in \mathbb{N}}$ *is said to possess the Markov property if and only if for each* $x_0, ..., x_{t+1} \in S$ *and* $t \in \mathbb{N}$

$$\mathbb{P}(X_{t+1} = x_{t+1} | X_t = x_t, X_{t-1} = x_{t-1}, ..., X_0 = x_0) = \mathbb{P}(X_{t+1} = x_{t+1} | X_t = x_t). \tag{18}$$

The Markov property is also called *memorylessness*, since it is sufficient to memorize the probability of the state before, no further information is necessary.

The probability of changing from one state to another is called *transition probability*, which is defined as

$$p_{ij}(t) = \mathbb{P}(X_{t+1} = x_j | X_t = x_i), \tag{19}$$

where $x_i, x_j \in S$ and $i, j \in \mathbb{N}$. The transition probabilities can be summarized in a *transition matrix*. A transition matrix for a finite Markov chain where $i, j \in \{1, ..., n\}$ with $n \in \mathbb{N}$ has the form

$$M(t) = (p_{ij}(t))_{i,j\in\{1,\dots,n\}} = \begin{bmatrix} p_{11}(t) & p_{12}(t) & \dots & p_{1n}(t) \\ p_{21}(t) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{n1}(t) & \dots & \dots & p_{nn}(t) \end{bmatrix}. \tag{20}$$

5.2 Properties of Markov chains

a) Homogeneous / Stationary Markov chain If the transition probabilities $p_{ij}(t)$ from state x_i to state x_j are independent from time t, i.e. $p_{ij}(t) = p_{ij}$ for all $t \in \mathbb{N}$, the Markov chain is called *homogeneous* or stationary Markov chain. It means that

$$p_{ij} = \mathbb{P}(X_{t+1} = x_j | X_t = x_i) = \mathbb{P}(X_t = x_j | X_{t-1} = x_i). \tag{21}$$

b) Irreducible Markov chain In *irreducible* Markov chains, any state is reachable in finite time, independent of the present state. Formally a Markov chain is irreducible if there exists any $m < \infty$ with

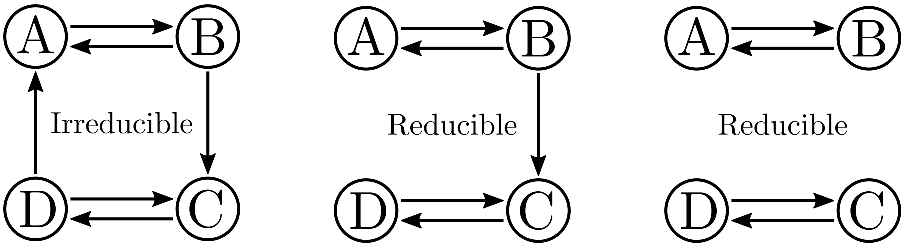
$$\mathbb{P}(X_{t+m} = x_j | X_t = x_i) = p_{ij}^{(t+m)} > 0 \tag{22}$$

for all i, j. Examples for reducible and irreducible Markov chains are given in figure 10a.

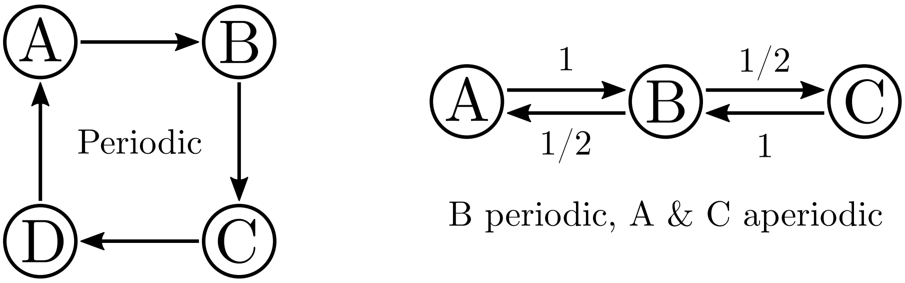
c) Aperiodic Markov chain If a state x_i returns with a multiple of k again to state x_i, it is called *periodic*, otherwise *aperiodic*. Formally a *period* is defined as

$$k = \gcd\{t > 0 \, : \, \mathbb{P}(X_t = x_i | X_0 = x_i) = p_{ii}^{(t)} > 0\}, \tag{23}$$

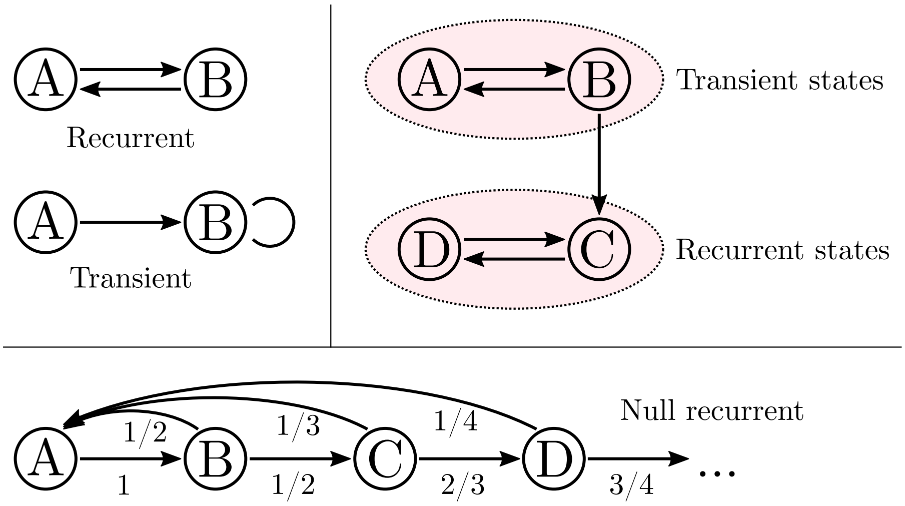
where gcd is the greatest common divisor. If the set is empty, the period is not defined. In case



(a) In irreducible Markov chains every state is reachable in finite time, independent of the present state. The first chain on the left side fulfills this property, while the center and right chain do not. The center chain may start in A or B, but if C is reached, A and B are not reachable any more. In the right chain, states A and B are totally separated from C and D.



(b) A state is periodic, if it returns back to the same state periodically, if not, the state is aperiodic. In the chain on the left side every state is periodic, since every state returns after 4 steps. Therefore, the whole chain is periodic. On the right side, only state B returns periodically every second time, states A and C are not periodic.



(c) If a state will be reached again almost surely, it is recurrent, otherwise transient. Two simple examples are given on top left. In the first small chain, both states A and B will be reached again all the time. In the second example A directly goes to B and will not be active any more. State A is transient. On the top right, states A and B are transient. If C is activated the first time, the chain will never reach A and B again. On the other hand, C and D are recurrent, both can be reached again all the time. The last example on the bottom shows a null recurrent case. Shown is a Markov chain with infinite states. Since the probability of going back to A is decreasing with a sequence $1/n$ for $n \to \infty$, the time to go back to A is $\mathbb{E}[T_A] = \infty$. Regarding the last chain, more details can be found in a proof in appendix in section A.

Figure 10: Three properties of Markov chains are shown: irreducibility, periodicity and recurrence. If no probability is written at an arrow, the probability is simply $p > 0$.

that $k = 1$ the state x_i is aperiodic and otherwise, if $k > 1$, x_i is periodic.

If all states are aperiodic, the Markov chain is called aperiodic. Examples for periodicity and aperiodicity are given in figure 10b.

d) Recurrent Markov chain A state x_i of a Markov chain is called *recurrent* if it will be reached again almost surely. Therefore, the probability to come back to state x_i is one. Let

$$T_{x_i} = \inf\{t \geq 0 \,:\, X_t = x_i | X_0 = x_i\} \tag{24}$$

be a random variable, which defines the number of steps until state x_i is reached again for the first time and let $\mathbb{P}(T_{x_i} = t)$ the probability to reach x_i again for the first time after exactly t steps. Then state x_i is recurrent, if and only if

$$\mathbb{P}(T_{x_i} < \infty) = \sum_{t=1}^{\infty} \mathbb{P}(T_{x_i} = t) = 1, \tag{25}$$

If a state is not recurrent, it is called *transient*, which is the case if $\mathbb{P}(T_{x_i} < \infty) < 1$. It means that the state will not be reached again almost surely.

Furthermore, if the state is recurrent and the expected value is finite, thus

$$\mathbb{E}[T_{x_i}] = \sum_{t=1}^{\infty} t \cdot \mathbb{P}(T_{x_i} = t) < \infty, \tag{26}$$

the state is called *positive recurrent.* Otherwise, if $\mathbb{E}[T_{x_i}] = \infty$, it is called *null recurrent.*

If all states are (positive) recurrent, the Markov chain is called (positive) recurrent. Examples for recurrent and transient Markov chains are given in figure 10c. A proof for the null recurrent chain in the figure can be found in the appendix in section A.

5.3 Stationary distribution

Definition 2 (Stationary distribution)**.** *Given a homogeneous Markov chain in discrete time* $t \in \mathbb{N}$ *on state space* $S = \{x_1, x_2, ...\}$*, the distribution* $\mathbb{P}_\pi$ *is called stationary distribution if and* *only if*

$$\boldsymbol{\pi}_{x_j} = \mathbb{P}_\pi(X_t = x_j) = \sum_{x_i \in S} \mathbb{P}_\pi(X_t = x_i)\, p_{ij} \overset{(21)}{=} \sum_{x_i \in S} \mathbb{P}_\pi(X_t = x_i)\, \mathbb{P}(X_t = x_j | X_{t-1} = x_i) \tag{27}$$

for all $x_j \in S$.

The stationary distribution should not be confused with the stationary Markov chain. While the stationary Markov chain is stationary regarding the transition probabilities, the stationary distribution is stationary regarding the probability of the states. Furthermore, the presence of a stationary Markov chain (homogeneous Markov chain) is a condition for the existence of a stationary distribution.

It is not always given that a stationary distribution exists. Furthermore a stationary distribution is not necessarily unique. In the following, first, the general Perron-Frobenius-Theorem (Seneta, 2006; Pillai, Suel, & Cha, 2005) is presented. Based on this theorem, a statement about the stationary distribution can be derived. While this theorem is highly useful, it assumes aperiodic Markov chains, which are not given in any case. Hence, afterwards another theorem is introduced, which lowers the assumption. At that point I want to remind, that a *homogeneous* Markov chain is assumed in the following.

Definition 3 (Primitive matrix). *Given a squared matrix* $Q = (q_{ij})_{i,j \in \mathbb{N}}$, *it is called a primitive matrix if all items are non-negative,* $q_{ij} > 0\, \forall i, j$, *denoted by* $Q > 0$ *and if a* $k \in \mathbb{N}$ *exists, such that all items of the matrix* Q^k *(k-th power of* Q*) are positive, denoted by* $Q^k > 0$.

Theorem 1 (Perron-Frobenius). *Suppose* Q *is an* $n \times n$ *non-negative primitive matrix. Then there exists an eigenvalue* r, *called Perron-Frobenius eigenvalue, such that:*

1. r *is real and positive,* $r > 0$.
2. r *can be associated with strictly positive left and right eigenvectors.*
3. $r > |\lambda|$ *for any other eigenvalue* $\lambda \neq r$.
4. *The eigenvectors associated with* r *are unique to constant multiples.*
5. *If* $0 \leq B \leq Q$ *and* β *is eigenvalue of* B, *then* $|\beta| \leq r$. *Moreover,* $|\beta| = r$ *implies* $B = Q$.
6. r *is a simple root of the characteristic equation of* Q.

A proof is given in Seneta (2006, Theorem 1.1). Using the Perron-Frobenius theorem it is possible to state in which cases a unique solution for the stationary distribution can be found.

Theorem 2. *Let M be a transition matrix and $\mathbf{1}$ a vector where all entries are one. An irreducible and aperiodic Markov chain has a unique stationary distribution given by the solution $\boldsymbol{\pi}$ of $\boldsymbol{\pi}^T M = \boldsymbol{\pi}^T$, where $\boldsymbol{\pi}^T \mathbf{1} = 1$.*

This theorem will be proven, but two corollaries are necessary.

Corollary 1. *An irreducible and aperiodic matrix is primitive.*

Corollary 2. *Let Q be a positive square matrix. Then the minimal row sum is a lower bound and the maximal row sum is an upper bound of the the largest eigenvalue of Q.*

Proof for theorem 2. Since any transition matrix M is per definition square and non-negative, the statements of the Perron-Frobenius theorem hold for transition matrix M, if the Markov chain is primitive. Since corollary 1 states that any irreducible and aperiodic matrix is primitive, the assumptions of the theorem are clarified.

Furthermore, it holds that

$$M\mathbf{1} = \mathbf{1},$$

since every row sums to one by definition of the transition matrix. Hence, M has an eigenvalue of 1 and an eigenvector of $\mathbf{1}$. With corollary 2, the largest eigenvalue is exactly the row sum 1, since all rows of M have the same sum. Therefore, the eigenvalue $r = 1$ is the Perron-Frobenius eigenvalue, since property (3) of the theorem states that all other eigenvalues are smaller in absolute. The corresponding right Perron-Frobenius eigenvector is $\mathbf{1}$.

Assuming that a vector $\boldsymbol{v}^T$ is normed, such that $\boldsymbol{v}^T \mathbf{1} = 1$, the eigenvalue problem for the left eigenvector $\boldsymbol{v}^T$ has the following form:

$$\boldsymbol{v}^T M = r\boldsymbol{v}^T = \boldsymbol{v}^T$$

From statement (2), we know that the left eigenvector regarding r is strictly positive. Therefore, $\boldsymbol{v}^T = \boldsymbol{\pi}^T$ is a distribution, the stationary distribution.

Finally, statement (4) of the Perron-Frobenius theorem states that $\boldsymbol{\pi}$ is unique. $\square$

As a side note, using theorem 2, the stationary distribution can be calculated as a right eigenvalue problem, just by transposing both sides $(\boldsymbol{\pi}^T M)^T = (\boldsymbol{\pi}^T)^T$, which results in $M^T \boldsymbol{\pi} = \boldsymbol{\pi}$.

Theorem 2 is limited to aperiodic Markov chains. Another theorem lowers the assumptions.

Theorem 3. *An irreducible and positive recurrent Markov chain has a unique stationary distribution* $\boldsymbol{\pi}$ *given by*

$$\pi_{x_i} = \frac{1}{\mathbb{E}[T_{x_i}]} \tag{28}$$

A proof is given in Bladt and Nielsen (2017, Corollary 1.2.29). The theorem uses the definition of the random variable T_{x_i} from equation (24), which indicates the number of steps, necessary to come back to state x_i again for the first time. Furthermore $\mathbb{E}[T_{x_i}]$ was defined in equation (26).

Theorem 4. *An irreducible and finite Markov chain is positive recurrent.*

This theorem is proven in Brémaud (2013, Theorem 3.3). It shows, that theorem 3 is highly practical for finite Markov chains. Since, in case of finite Markov chains, only the property of irreducibility needs to be fulfilled.

The Perron-Frobenius theorem has a slight advantage, compared to theorem 3. Using the Perron-Frobenius theorem, it can be shown that even a *limiting distribution* can be obtained. It is defined by $\pi_i = \lim_{t\to\infty} \mathbb{P}(X_t = x_i)$. In case of theorem 3, this is not necessarily the case. If we assume for example a periodic Markov chain with just two states, which are alternating $(0, 1, 0, 1, 0, 1, ...)$, the Markov chain has no limiting distribution, but a unique stationary distribution. However, for calculating stationary distributions, in most cases theorem 2 was used.

5.4 Measures

To evaluate the stationary distribution many measures can be used, which are common for any distribution, not only for Markov chains. Two types are introduced, which are used in the results. At first, information measures, namely variance and Kullback-Leibler divergence, are defined and afterwards a measure for the concentration of a distribution is presented, where the Lorenz curve and the Gini coefficient are introduced. Note that the Gini coefficient is directly derived from the Lorenz curve.

Information measures For simplification, the state space can be defined as $S = \{1, ..., n\}$ where $n \in \mathbb{N}$. A simple measure for the stationary distribution is the deviation of the probabilities from their mean probability. Therefore an empirical variance measure is used by calculating

$$\sigma_\pi^2 = \frac{1}{n} \sum_{k=1}^{n} (\pi_k - \bar{\pi})^2, \tag{29}$$

where $\bar{\pi} = \frac{1}{n} \sum_{k=1}^{n} \pi_k$. In the following it is just called *variance.*

Another measure uses the Kullback-Leibler divergence. If $\boldsymbol{\pi}' \in \{\pi_1', ..., \pi_n'\} = \{1/n, ..., 1/n\}$ is a stationary distribution where all states are equally probable, the Kullback-Leibler divergence can be obtained by

$$D_{\mathrm{KL}}(\boldsymbol{\pi} \| \boldsymbol{\pi}') = \sum_{i=1}^{n} \pi_i \ln\left(\frac{\pi_i}{\pi_i'}\right). \tag{30}$$

The Kullback-Leibler divergence can be understood as relative entropy. It describes the increase in information which is necessary to describe $\boldsymbol{\pi}$, when $\boldsymbol{\pi}'$ is given. It is important to note that the Kullback-Leibler divergence is an asymmetric measure. The value of the measure depends on the order of the distributions and therefore the divergence does not fulfill the properties of a *metric.* It is in particular different from the *total variation distance* $\delta(\boldsymbol{\pi}, \boldsymbol{\pi}')$. A link between the total variation distance and the Kullback-Leibler divergence is given by *Pinsker's inequality.* However, *Gibb's inequality* states that

$$D_{\mathrm{KL}}(\boldsymbol{\pi} \| \boldsymbol{\pi}') \geq 0 \tag{31}$$

and $D_{\mathrm{KL}}(\boldsymbol{\pi} \| \boldsymbol{\pi}') = 0$ if and only if $\boldsymbol{\pi} = \boldsymbol{\pi}'$, which means that a divergence of 0 indicates a match between the distributions $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$.

Concentration measures Another way of evaluating the properties of a Markov chain is to plot the *Lorenz curve* and calculate the *Gini coefficient*. Traditionally, both is often applied to indicate inequality regarding income, used in social sciences. But the Lorenz curve shows the concentration of a distribution in general, whereas the Gini coefficient is a related measure for the intensity of the concentration of the given distribution.

First, assume that $\boldsymbol{\pi} = (\pi_1, ..., \pi_n)^T$ is sorted in a sense that $\pi_1 \leq ... \leq \pi_n$ and denote $r_i = 1/n$ as the share of a state with probability π_i, where $n \in \mathbb{N}$ is the number of states. Furthermore $R_k = \sum_{i=1}^{k} r_i = \sum_{i=1}^{k} 1/n = k/n$ is the cumulative share of the states. On the other hand, $P_k = \sum_{i=1}^{k} \pi_i$ is the cumulated probability of $\boldsymbol{\pi}$. The *Lorenz curve* shows the cumulated share of states R_k at the x-axis and the cumulated probability P_k at the y-axis. If all probabilities are equal, it will result in a straight diagonal line. The more the distribution $\boldsymbol{\pi}$ is concentrated, the more the line will appear in the right bottom corner. An illustration is given in figure 11.

Finally, the *Gini coefficient* G is the ratio of the area that lies between the diagonal line, where all states are equal probable, and the area between the actual Lorenz curve and the diagonal. The more the Lorenz curve tends to the corner and, therefore, the more the stationary distribution is unequal, the more the Gini coefficient will tend to 1, where 1 represents a perfect concentrated situation. It means that the whole probability is concentrated at one state, the other states are zero. If the distribution is perfectly equal, the Gini coefficient will reach 0. It can be shown that $G \in [0, 1]$ is half of the *relative mean absolute difference*. Hence, for the

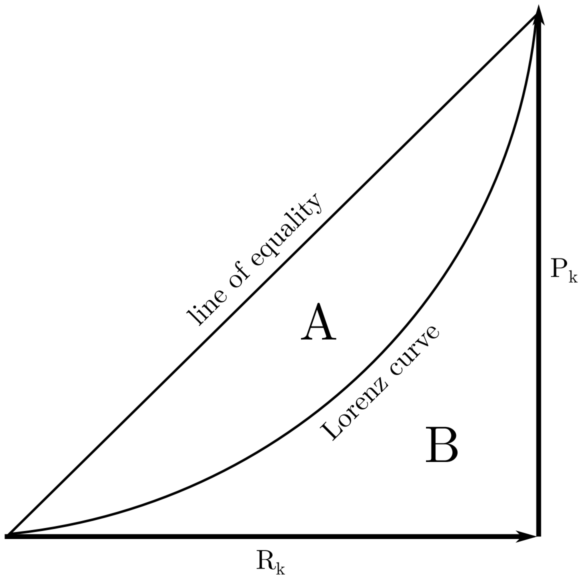


Figure 11: Illustration of a Lorenz curve. On the x-axis is the cumulative share of the probabilities from the states of an equally distributed stationary distribution. On the y-axis is the cumulative share of the probabilities of the actual stationary distribution. If the actual distribution is equally distributed, the Lorenz curve reaches the line of equality. The Gini coefficient is calculated by $G = A/(A + B)$.

discrete case, the Gini coefficient is defined as

$$G = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |\pi_i - \pi_j|}{2n \cdot \sum_{i=1}^{n} \pi_i}. \tag{32}$$

5.5 Markov Chain Monte Carlo (MCMC)

A *Monte Carlo simulation* is a stochastic procedure where random samples are drawn many times. As a result, numerical solutions for analytically difficult or even impossible problems are obtained. Thomopoulos (2012) described the process as follows:

> To apply the Monte Carlo method, the analyst constructs a mathematical model that simulates a real system. A large number of random sampling of the model is applied yielding a large number of random samples of output results from the model.

Specifically, if random samples are drawn from the distribution of a Markov chain, it is called *Markov chain Monte Carlo* (MCMC) method which includes a broad class of algorithms. Famous and well known algorithms are the *Metropolis–Hastings algorithm* (Hastings, 1970) or the *Gibbs sampler* (S. Geman & D. Geman, 1984). These techniques are often used in Bayesian inference to evaluate posterior distributions.

There are many different possibilities to understand the behavior of neural networks under the perspective of MCMC sampling. For example, there are applications for Boltzmann machines (Osogami, 2017) or energy-based models in general (Goodfellow et al., 2017). In SORN, it is possible to view the learned time-dependent input patterns under the perspective of MCMC sampling. The approach will be introduced in methods (section 6) and many illustrative implementations will be given in results (section 7).

6 Methods

A self-organizing recurrent neural network (SORN) was used to learn specific input patterns, given by a defined Markov chain. It was checked how well the network was able to represent the given information. SORN uses three plasticity rules to learn the input patterns:

- Spike-timing-dependent plasticity (STDP)
- Synaptic normalization (SN)
- Intrinsic plasticity (IP)

The time in the network is discrete and there is no refractory period modeled for the neurons. In most cases $N_{\text{sim}} = 20$ networks were simulated and results were averaged.

The source code for all the simulation experiments is based on source code from Hartmann et al. (2015), which is written in `Python 2.7`. The team provided a small framework for SORN simulation experiments (https://github.com/chrhartm/SORN). A lot of code was added or rewritten, especially regarding the evaluation of the network behavior, such that it fitted to the experiments, performed in this thesis.

6.1 Network structure

The reservoir consists of N^E excitatory and N^I inhibitory neurons. Additionally, some excitatory neurons where randomly connected to a specific number of input neurons $N^U = n$, which equals the number of Markov states. The weights $w_{ij} \in [0, 1]$ between the neurons were initialized randomly. The network is build with three types of connections:

- Excitatory-excitatory weights: W^{EE}, where $\sum_j w_{ij}^{EE} = 1$
- Inhibitory-excitatory weights: W^{EI}, where $\sum_j w_{ij}^{EI} = 1$
- Excitatory-inhibitory weights: W^{IE}, where $\sum_j w_{ij}^{IE} = 1$
- Input-excitatory weights: W^{EU}, where $w_{ij}^{EU} = 1$ for all i, j

As already explained in section 4.4, there are no connections modeled between inhibitory neurons. On average, 10% of the possible connections in W^{EE} were realized. Hence, the matrix W^{EE} is sparsely connected. The matrices W^{IE} and W^{EI} are fully connected. Note that the weights are normalized using equation (6), such that the incoming weights to a neuron sum up to one.

In section 4.4, figure 9 shows a sketch of the structure of the reservoir network. In this thesis $n = 4$ states were used. Hence, the state space consists of $S = \{x_1, x_2, x_3, x_4\} = \{A, B, C, D\}$. Three states are no enough to test a variety of combinations, while five or more states are already to complicated for a systematic approach. Therefore, four states seemed to be a good compromise. The four states are represented with neuron populations, called clusters. In most cases the network consists of $N^E = 200$ and $N^I = 40$. Furthermore $N^U_{x_i} = 10$ input connections are used per state or input neuron.

6.2 Modification of the intrinsic plasticity

While the STDP and the SN where applied analog to Lazar et al. (2009), the IP rule was modified. In the original SORN model, the target rate H_{IP} was fixed. Hartmann et al. (2015) found out that the model is more robust, if the average firing rate differs slightly between neurons. Therefore, the H_{IP} value from equation (7) was replaced by an individual H^{IP}_i target rate for every neuron i, resulting in

$$T_i(t+1) = T_i(t) + \eta_{\mathrm{IP}} \left(x_i(t) - H^{\mathrm{IP}}_i \right). \tag{33}$$

To get individual values, an average value $\bar{H}_{\mathrm{IP}} = 2 \cdot N^U / N^E$ was used. It equals the original heuristic from Lazar et al. (2009). Additionally, a stochastic term $\varepsilon^{\mathrm{IP}}_i$ was added to the target rate of every neuron. It is uniformly i.i.d. on an interval of $a = \sigma^{\mathrm{IP}} = 0.01$ and $b = -a = -\sigma^{\mathrm{IP}} = -0.01$. Therefore, $\varepsilon^{\mathrm{IP}}_i \sim \mathrm{Uni}[a, b] = \mathrm{Uni}[-\sigma^{\mathrm{IP}}, +\sigma^{\mathrm{IP}}] = \mathrm{Uni}[-0.01, 0.01]$. Together,

$$T_i(t+1) = T_i(t) + \eta_{\mathrm{IP}} \left(x_i(t) - \bar{H}^{\mathrm{IP}} + \varepsilon^{\mathrm{IP}}_i \right) \tag{34}$$

is the rule that was applied in the used network.

6.3 Network training and testing

The network was trained in 3 phases:

- Self-organizing phase (plastic training phase)
- Training phase (no-plastic training phase)
- Testing phase

Self-organizing phase In the self-organizing phase (or plastic training phase) the input neurons are active, depending on the current sample x_i from the Markov chain. In this phase all three plasticity rules are active. While STDP actively adjusts the weights, SN stabilizes the activity around every neuron without loosing the relative relationships between the neurons. IP also helps to ensure the stability of the network, since it adjusts thresholds of neurons depending on their past activity. The update of the network activity follows equation (15) for the excitatory neurons and equation (16) for the inhibitory neurons.

During self-organizing, the information from the Markov chain is stored in the network and is coded in the weights between the neurons. Biologically, the encoding in the weights corresponds to the encoding in the strength of the synapses. In most cases the self-organizing phase was applied for $T_{\text{plastic}} = 50,000$ steps. As shown below in the results, e.g. in figure 15, latest after $50,000$ steps no further improvement can be noticed, in most cases much sooner.

Training phase In the training phase (or no-plastic training phase) the input neurons are still active and they still depend on the current state x_i of the Markov chain. In this phase only IP is active. Therefore, the stability of the network is ensured further on. STDP is not active and thus also SN is not necessary. Hence, the weights stay fixed in the training phase, meaning that $W_{ij}^{EE}(t) = W_{ij}^{EE}$ in equation (15), where the rest of the network update equals the self-organizing phase.

The main role of the training phase is to find out how the states are represented in the activity patterns of the network after the weight structure was learned. After a burn in phase, Hartmann et al. (2015) used the last $\tilde{T}_{\text{compare}} = 2500$ steps of the no-plastic training phase for classification (see section 6.4). In the results section, it is shown that this fixed value causes a small bias, which was solved when choosing $\tilde{T}_{\text{compare}}$ more flexible, which is explained below in section 6.4.

The training phase needs enough steps to cover $T_{\text{noplastic}} > \tilde{T}_{\text{compare}}$. However, the training was mostly chosen as $T_{\text{noplastic}} = 50,000$ no-plastic training steps. There are four reasons for this high value:

- The computational effort is very low in the training phase.
- The number of no-plastic training steps do not influence the performance of the network (see appendix C).
- After switching off the self-organizing phase (namely the STDP and SN), there should be a short burn-in period. Hartmann et al. (2015) used $T_{\text{noplastic}} = 20,000$, which seemed to be enough in all cases.
- The updated classification algorithm, mentioned above, implies a flexible size of $\tilde{T}_{\text{compare}}$, depending on the Markov chain. This change caused that a longer training phase is necessary in some cases. $T_{\text{noplastic}} = 50,000$ was enough for all applications.

Testing phase In the testing phase the input neurons are not active any more, there is no input at all. It means that $\boldsymbol{v}(t) = 0$ for all $t \in \{T_{\text{evoked}} + 1, ..., T_{\text{evoked}} + T_{\text{test}}\}$, where $T_{\text{evoked}} = T_{\text{plastic}} + T_{\text{noplastic}}$. Also STDP and SN is still switched off, as it was in the training phase before. IP is still active. Beside the stabilizing character of the IP, especially in the testing phase, it is responsible for spontaneous activity (see section 3.5). In the testing phase, the spontaneous activity patterns of the network are observed over time.

Since no input is given in this phase, it is necessary to classify the spontaneous activity patterns to a Markov state. For that purpose, the representations from the no-plastic training phase are used. A detailed explanation of the classification mechanism is given below.

The number of steps while testing has to take into account a short period of burn in. After the training phase, the input is switched off and therefore the network has to find equilibrium again, using just spontaneous activity from intrinsic plasticity. Mathematically, the burn in time depends on the learning rate of the IP mechanism η_{IP}. This relation is evaluated in appendix F. In most cases the burn in phase is relatively short and $T_{\text{test}} > 20,000$ is sufficient.

6.4 Classification of states

In the last phase, the testing phase, it will be necessary to classify a Markov state x_i to the activity pattern $\boldsymbol{x}(t)$ in every step of time. The activity patterns from the testing phase are compared to some representative activity patterns in the end of the no-plastic training phase.

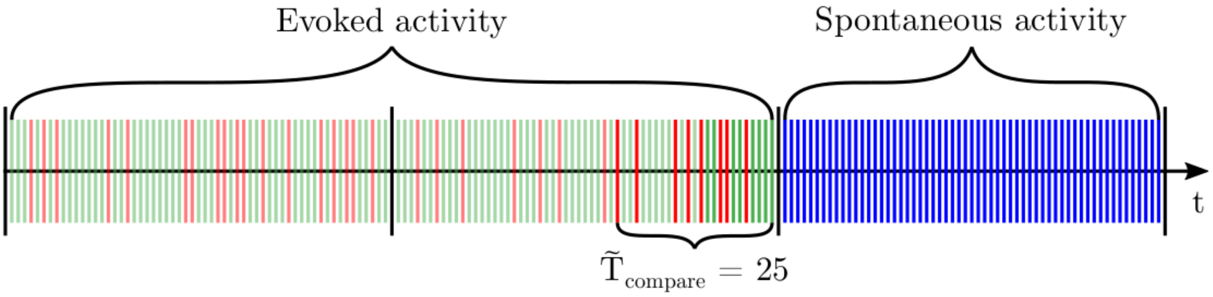
Selection mechanism from Hartmann In Hartmann et al. (2015) a fixed amount of $\tilde{T}_{\text{compare}} = 2500$ steps in the training phase was used as representative activity patterns. From those steps the number of occurrences for every state $\tilde{T}^{x_i}_{\text{compare}}$ was calculated. Let

$$\begin{aligned}\tilde{R}_{\text{compare}} &= \Big\{\boldsymbol{x}_u(T_{\text{plastic}} + T_{\text{noplastic}} - \tilde{T}_{\text{compare}}), ..., \boldsymbol{x}_u(T_{\text{plastic}} + T_{\text{noplastic}})\Big\} \\ &= \Big\{\boldsymbol{x}_u(T_{\text{evoked}} - \tilde{T}_{\text{compare}}), ..., \boldsymbol{x}_u(T_{\text{evoked}})\Big\}\end{aligned} \tag{35}$$

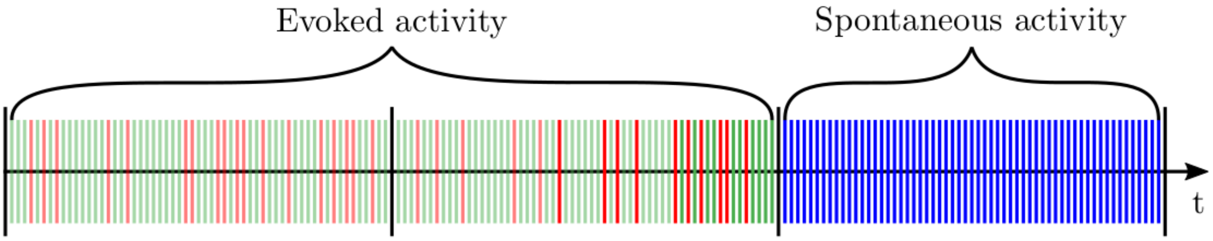
be a set of the last $\tilde{T}_{\text{compare}}$ activity patterns in the no-plastic phase, where $u \in \{x_1, ..., x_n\}$ indicates the given input for the specific step. In order to avoid confusion, it is important to remind that the vector $\boldsymbol{x}_u(t)$ denotes the activity pattern of the excitatory neurons at time t with input u and the scalar x_i denotes a state of the Markov chain. The obtained activity patterns $\tilde{R}_{\text{compare}}$ are filtered by Markov state, denoted by $\tilde{R}^{x_i}_{\text{compare}} = \{\tilde{R}_{\text{compare}} | u = x_i\}$. The size of this set is defined as $\tilde{T}^{x_i}_{\text{compare}} = |\tilde{R}^{x_i}_{\text{compare}}|$, which was mentioned above.

The minimum of those occurrences $T^{\min}_{\text{compare}} = \min\{\tilde{T}^{x_1}_{\text{compare}}, ..., \tilde{T}^{x_n}_{\text{compare}}\}$ was chosen as the number of steps for every state. If the number of representative activity patterns per state would differ, those with a higher amount of steps have a higher chance to be classified, since the amount of representative patterns cover more possible cases. Altogether, $n \cdot T^{\min}_{\text{compare}}$ steps are available for comparison, where n is the number of Markov states or input neurons. The corresponding activity pattern for a specific state is denoted by $R^{x_i}_{\text{compare}}$. To clarify how it works, a short example is provided:

Assume just two possible states A and B, thus $n = 2$. Further, assume that a Markov chain exists which provides a stationary distribution with $\pi_A = 2/5$ and $\pi_B = 3/5$. A likely sampling would result in $\tilde{T}^{A}_{\text{compare}} = 997$ (where $\tilde{T}_{\text{compare}} \cdot \pi_A = 2500 \cdot 2/5 = 1000$ would be expected) and $\tilde{T}^{B}_{\text{compare}} = 1503$ (where $\tilde{T}_{\text{compare}} \cdot \pi_B = 1500$ would be expected). Finally, in testing phase are more representative activity patterns of B than A available to compare with. Hence, B has an advantage, compared to A, and activity patterns in the testing state will eventually more likely be classified as B. To avoid this possible bias, the minimum of both values is taken and $T^{\min}_{\text{compare}} = \min\{\tilde{T}^{A}_{\text{compare}}, \tilde{T}^{B}_{\text{compare}}\} = \min\{997, 1503\} = 997$ is the number of steps which are used for *every* state to compare with, meaning that $T^{\min}_{\text{compare}} = T^{A}_{\text{compare}} = T^{B}_{\text{compare}}$. Figure 12a illustrates a similar example.



(a) A fixed number of potential activity patterns is chosen. In this example $\tilde{T}_{\text{compare}} = 25$, meaning that the last 25 steps in time are taken into account. In this period, state A occurs $\tilde{T}^{A}_{\text{compare}} = 8$ times and state B $\tilde{T}^{B}_{\text{compare}} = 17$ times. To ensure that the later classification of the spontaneous activity pattern have the same chance to be classified, $T^{\text{min}}_{\text{compare}} = T^{A}_{\text{compare}} = T^{B}_{\text{compare}} = 8$ is chosen for both, state A and B.



(b) Since the $T^{\text{min}}_{\text{compare}}$ from the solution from Hartmann, Lazar, Nessler, and Triesch (2015) varies between input patterns with different probabilities, another bias needs to be taken into account when different learning patterns should be compared. Therefore, independent of the probability structure, $T_{\text{compare}} = T^{A}_{\text{compare}} = T^{B}_{\text{compare}} = 10$ are chosen for all states. The search for those states begins at the end of the no-plastic training phase.

Figure 12: Selection mechanism for activity pattern representations. In both illustrations, the first block is the plastic training phase, the second block represents the no-plastic training phase and the last block is the testing phase. Since input is given in the first two blocks, they perform with evoked activity. The last block has no input anymore and runs only with spontaneous activity from IP. The example network is trained with two states: A, represented by the red lines, and B, drawn in green, where $\pi_A = 2/5$ and $\pi_B = 3/5$. Every line represents a point in time. The opaque lines are the selected steps, which are used as representative activity patterns for the specific state of the Markov chain. The activity of the chosen steps is collected in R_{compare} in general and in R^{A}_{compare} & R^{B}_{compare} specifically. The first graphic shows the method provided by Hartmann, Lazar, Nessler, and Triesch (2015), the second graphic illustrates the method which is applied in the network of this thesis.

Updated selection mechanism While evaluating the network, another bias was found. The algorithm from Hartmann et al. (2015) makes sure that the classification remains unbiased, if the same input pattern is chosen. But if different patterns are compared, the classification mechanism depends on the probabilities of the states. If one state in the input pattern is very rare, $T^{\text{min}}_{\text{compare}}$ will be very small and therefore all activity patterns in the testing phase have just few representative patterns in the no-plastic training phase to compare with. On the other hand, if all states are equally probable, $T^{\text{min}}_{\text{compare}}$ is quite high compared to the rare case. The performance between those two cases will eventually differ, just because of the number of available representative states in no-plastic training phase. In terms of Markov chains: If Markov chains with different stationary distributions are learned by the network and compared with each other, it is necessary to keep the number of $T^{\text{min}}_{\text{compare}}$ constant for all chains. In section 7.1 this effect is illustrated and discussed in detail.

An approach, which was chosen in this thesis, is to fix the number of representing steps for every state. A value of $T_{\text{compare}} = 500$ was chosen. Independent of the probability of the state, starting from the end, 500 representing activity patterns for every state were collected. If a state is very unlikely, for example with a probability of $\pi_{x_i} = 0.05$, for 500 occurrences, $10,000$ steps are necessary in average to find enough representations for that state. Therefore, the $T_{\text{noplastic}} = 50,000$ was chosen, as indicated before. An illustration of this selection mechanism is shown in figure 12b.

Classification mechanism After the representational activity patterns R_{compare} are collected with equal length $T^{x_1}_{\text{compare}} = ... = T^{x_n}_{\text{compare}}$, independent of the probabilities of the Markov chain, it remains to describe the classification mechanism for the activity patterns in the testing phase. First, a distance measure, the *Hamming distance* is defined.

Definition 4 (Hamming Distance). *Let Ω be an alphabet and $\boldsymbol{x} = (x_1, ..., x_N) \in \Omega^N$, $\boldsymbol{y} = (y_1, ..., y_N) \in \Omega^N$ two words with length $N \in \mathbb{N}$. Then the* hamming distance *between those words $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as:*

$$d(\boldsymbol{x}, \boldsymbol{y}) := |\{i \in \{1, ..., N\} : x_i \neq y_i\}| \tag{36}$$

For the network holds that $\Omega = \{0, 1\}$. The Hamming distances between every step in the testing phase $\boldsymbol{x}(t_{\text{test}}) \in R_{\text{test}}$, where $R_{\text{test}} = \{x(T_{\text{evoked}} + 1), ..., x(T_{\text{evoked}} + T_{\text{test}})\}$, and all steps in $\boldsymbol{x}(t_{\text{compare}}) \in R_{\text{compare}}$ are calculated. In that case the Hamming distance is defined as

$$d(\boldsymbol{x}(t_{\text{test}}), \boldsymbol{x}(t_{\text{compare}})) = |\{i \in \{1, ..., N^E\} : x_i(t_{\text{test}}) \neq x_i(t_{\text{compare}})\}|. \tag{37}$$

The activity pattern $\boldsymbol{x}(t_{\text{compare}})$ with the smallest distance $d(\boldsymbol{x}(t_{\text{test}}), \boldsymbol{x}(t_{\text{compare}}))$ for all t_{compare} is chosen and denoted by $\boldsymbol{x}^{\text{min}}_{\text{compare}}(t_{\text{test}})$. Corresponding to $\boldsymbol{x}^{\text{min}}_{\text{compare}}(t_{\text{test}})$ exists a specific input

x_i	$\boldsymbol{x}(t_{\text{compare}})$										$d\,(\boldsymbol{x}(t_{\text{test}}), \boldsymbol{x}(t_{\text{compare}}))$
A	**1**	**0**	0	**0**	**1**	0	**0**	0	**1**	0	6
A	0	**0**	0	**0**	0	**1**	**0**	**1**	0	**1**	6
A	0	1	0	**0**	0	**1**	**0**	**1**	**1**	0	5
A	0	**0**	**1**	**0**	0	**1**	1	0	**1**	**1**	6
A	0	**0**	0	1	0	0	**0**	**1**	**1**	**1**	5
B	0	1	**1**	**0**	0	0	1	0	0	0	2
A	0	1	0	1	0	**1**	**0**	**1**	**1**	**1**	6
B	0	1	**1**	1	0	0	**0**	0	0	0	2
A	**1**	**0**	0	**0**	0	**1**	1	**1**	**1**	**1**	6
B	**1**	1	**1**	1	0	0	1	0	0	**1**	2
B	0	**0**	**1**	1	0	0	**0**	0	0	0	3
A	**1**	1	0	**0**	0	0	1	**1**	**1**	**1**	5
A	0	**0**	0	**0**	0	**1**	**0**	**1**	**1**	**1**	7
B	0	1	**1**	1	0	0	1	0	0	0	**1**
B	0	1	**1**	**0**	0	0	**0**	**1**	**1**	0	5
A	0	**0**	0	**0**	0	0	**0**	0	0	**1**	4
B	0	**0**	**1**	1	0	0	**0**	0	0	0	3
B	**1**	1	**1**	1	0	0	**0**	**1**	0	0	4
B	0	1	**1**	1	0	0	**0**	**1**	0	0	3
B	0	**0**	0	1	0	0	1	0	0	0	**1**
$\boldsymbol{x}(t_{\text{test}})$	0	1	0	1	0	0	1	0	0	0	
$\boldsymbol{u}_A$	0	0	0	0	0	1	0	1	1	1	
$\boldsymbol{u}_B$	0	1	1	1	0	0	1	0	0	0	

Table 1: State classification using Hamming distance. Shown are $T^A_{\text{compare}} = T^B_{\text{compare}} = 10$ representative states from the no-plastic training phase, corresponding to figure 12b. An activity vector $\boldsymbol{x}(t_{\text{test}})$ and examples of input vectors $\boldsymbol{u}_A$ and $\boldsymbol{u}_B$, corresponding to the Markov states A and B are shown in the bottom of the table. On the right side the Hamming distance between any $\boldsymbol{x}(t_{\text{compare}})$ and the $\boldsymbol{x}(t_{\text{test}})$ is calculated. The differing digits in the vectors are bold. The smallest distance is given by two vectors corresponding to an B input, with a distance of just 1, which is highlighted in bold on the right side. Therefore, the chosen step in testing phase would be classified as B. This procedure is repeated for every $\boldsymbol{x}(t_{\text{test}}) \in R_{\text{test}}$.

$u \in \{x_1, ..., x_n\}$ which is chosen as the Markov state for the corresponding step t_{test} in the testing phase. A simple example, which gives an intuition is shown in table 1. The process is in principle a k-nearest neighbors algorithm with $k = 1$, where the distance between the vectors is measured using a hamming distance.

The question arises what happens if two or more patterns from training phase having equal distances, assuming that the patterns in the training phase belong to different Markov states. Hartmann et al. (2015) solved this problem by shuffling the training states, before calculating the distances and then taking the first element which has minimum distance. With this method a systematical bias is prevented.

However, the algorithm is relatively simple, which is its strength, at least regarding computational effort and simplicity in implementation. The classification method will be examined in the results and further discussed in the discussion.

6.5 Measuring performance

The reservoir network is trained with a specific Markov chain, where the states in the Markov chain represent input clusters in the neural network, modeled by input neurons. Ideally, the network finds a good representation of the transition probabilities and the stationary distribution, stored in the weights, to reproduce the Markov chain. Statistically, the network can be seen as an estimator of the true probability structure of the Markov chain. An estimator in that sense can be optimized by finding better parameter configurations of the network. The approach does not include an analytic study of the estimator characteristics, rather it is a simulation study.

Let M denote the true transition matrix and $\hat{M}$ the estimator of the transition matrix. Analogously, $\boldsymbol{\pi}$ is the true stationary distribution of the Markov chain and $\hat{\boldsymbol{\pi}}$ the estimator of the stationary distribution. To quantify the performance of the estimators $\hat{M}$ and $\hat{\boldsymbol{\pi}}$, a common mean squared error (MSE) can be implemented. For a general parameter $\boldsymbol{\vartheta}$ and associated estimator $\hat{\boldsymbol{\vartheta}}$, the MSE is defined as

$$\text{MSE}(\hat{\boldsymbol{\vartheta}}) = \mathbb{E}_{\hat{\boldsymbol{\vartheta}}}[(\hat{\boldsymbol{\vartheta}} - \boldsymbol{\vartheta})^T(\hat{\boldsymbol{\vartheta}} - \boldsymbol{\vartheta})]. \tag{38}$$

In both cases, $\hat{M}$ and $\hat{\boldsymbol{\pi}}$, the analytic solution is probably impossible or at least very hard to obtain, since both estimators are highly non-linear random variables. But for both estimators an estimator for the MSE can be introduced. It is based on a Monte Carlo approximation. The simulation was repeated N_{sim} times. For the vector parameter $\boldsymbol{\vartheta} \in \mathbb{R}^d$

$$\widehat{\text{MSE}}(\hat{\boldsymbol{\vartheta}}) = \frac{1}{d \cdot N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} (\hat{\boldsymbol{\vartheta}}_i - \boldsymbol{\vartheta}_i)^T (\hat{\boldsymbol{\vartheta}}_i - \boldsymbol{\vartheta}_i) \tag{39}$$

will be the general formulation of the used performance measure.

In case of the transition matrix M, a stochastic matrix, the rows need to be concatenated. Therefore, $\boldsymbol{m} = (p_{11}, ..., p_{1n}, p_{21}, ..., p_{2n}, p_{31}, ..., p_{nn})^T$ (compare with equation (20)) defines a vector, containing all entries from the transition matrix. Finally, the estimator of the MSE for the transition matrix estimator $\hat{M}$ is defined as

$$\varepsilon_M = \widehat{\text{MSE}}(\hat{\boldsymbol{m}}) = \frac{1}{n^2 \cdot N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} (\hat{\boldsymbol{m}}_i - \boldsymbol{m}_i)^T (\hat{\boldsymbol{m}}_i - \boldsymbol{m}_i). \tag{40}$$

In case of the stationary distribution, the calculation is straight forward. The estimator of the MSE for the stationary distribution estimator $\hat{\boldsymbol{\pi}}$ is defined as

$$\varepsilon_\pi = \widehat{\text{MSE}}(\hat{\boldsymbol{\pi}}) = \frac{1}{n \cdot N_{\text{sim}}} \sum_{i=1}^{N_{\text{sim}}} (\hat{\boldsymbol{\pi}}_i - \boldsymbol{\pi}_i)^T (\hat{\boldsymbol{\pi}}_i - \boldsymbol{\pi}_i). \tag{41}$$

It remains to show how $\hat{M}$ and $\hat{\boldsymbol{\pi}}$ are obtained. Since an analytic representation is highly complex, the estimators are empirically obtained from the output of the network. Assuming that the states of the Markov chain in the testing phase are already classified (using the method from section 6.4), the stationary distribution can directly be calculated by taking the frequencies of the states. If n_{x_i} is the number of occurrences for state x_i in the testing phase, the estimator is defined by

$$\hat{\boldsymbol{\pi}} = \left(\frac{n_{x_1}}{N_{\text{test}}}, ..., \frac{n_{x_n}}{N_{\text{test}}} \right)^T, \tag{42}$$

where $N_{\text{test}} = |\{\boldsymbol{x}(t_{\text{test}}) \,:\, \boldsymbol{x}(t_{\text{test}}) \neq 0\}|$ for all $t_{\text{test}} \in \{T_{\text{evoked}}+1, ..., T_{\text{evoked}}+T_{\text{test}}\}$ is the number of non-silent steps in the training phase, where $\boldsymbol{x}(t_{\text{test}})$ is an activity vector of the network, as used before. Note that $\boldsymbol{x}(t_{\text{test}}) \neq 0$ is equivalent to $x_i(t_{\text{test}}) \neq 0$ for all $i \in \{1, ..., N^E\}$. Equivalently $N_{\text{test}} = n_{x_1} + ... + n_{x_n} = \sum_{j=1}^{n} x_{x_j}$ is just the sum of all non-zero state occurrences.

To track changes in the course of the testing phase, the testing phase is split into chunks of $\Delta T_{\text{chunk}} = 5000$ steps. Hence, $\hat{\boldsymbol{\pi}}$ is calculated for every chunk. The number of chunks $K_{\text{chunks}} = T_{\text{test}}/\Delta T_{\text{chunk}}$ is simply the number of overall steps in the testing phase divided by the chunk size. Again, $N_{\text{test}}^k = |\{\boldsymbol{x}(t_{\text{test}}^k) \,|\, \boldsymbol{x}(t_{\text{test}}^k) \neq 0\}|$ for all $k \in \{1, ..., K_{\text{chunks}}\}$ describes the number of non-silent steps in every chunk k. The calculation of the estimator $\hat{\boldsymbol{\pi}}^k$ for a specific chunk k, changes to

$$\hat{\boldsymbol{\pi}}^k = \left(\frac{n_{x_1}^k}{N_{\text{test}}^k}, ..., \frac{n_{x_n}^k}{N_{\text{test}}^k} \right)^T . \tag{43}$$

For the calculation of the estimator for the transition matrix, the silent steps are ignored, too. If n_{x_i,x_j}^k denotes the number of transitions from state x_i to state x_j inside of chunk k, $\hat{M}^k$ is calculated as

$$\hat{M}^k = \left(\frac{n_{x_i,x_j}^k}{N_{\text{test}}^k} \right)_{x_i,x_j \in S} = \begin{bmatrix} \frac{n_{x_1,x_1}^k}{N_{\text{test}}^k} & \frac{n_{x_1,x_2}^k}{N_{\text{test}}^k} & \cdots & \frac{n_{x_1,x_n}^k}{N_{\text{test}}^k} \\ \frac{n_{x_2,x_1}^k}{N_{\text{test}}^k} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{n_{x_n,x_1}^k}{N_{\text{test}}^k} & \cdots & \cdots & \frac{n_{x_n,x_n}^k}{N_{\text{test}}^k} \end{bmatrix} . \tag{44}$$

Finally, the MSE estimator can be applied. As the estimator of the stationary distribution $\hat{\boldsymbol{\pi}}$ just contains information about how often a state occurs, the estimated transition matrix also contains information about the transition properties. Therefore, the $\widehat{\text{MSE}}(\hat{\boldsymbol{m}})$ is preferred in the following.

As a remark, it should be stated that also the hamming distance can be used as a performance measure. If the mean of the hamming distance of all steps in testing phase is calculated, it should be smaller for networks which perform better, since the learned states should be closer to the initial states. Since the differences in the hamming distance means do not differ much between different settings (see appendix D), this kind of performance measure was not further investigated.

A last problem was the amount of silent passes. The more silent passes exist, the worse will be the estimator. This problem was solved by just implementing a threshold. If the number of silent steps exceeded a threshold of 25% of ΔT_{chunk}, which means that if $1 - N^k_{\text{test}}/\Delta T_{\text{chunk}} > 0.25$ for any k, an error is thrown and the network simulation is canceled.

7 Results

In the last sections the basics of neurons and some important plasticity rules were presented. Afterwards, an introduction to artificial neural networks led to SORN, a reservoir network with dynamic self-organizing behavior using three plasticity rules. Finally, the principle of Markov chains was introduced and all components were combined in the methods section. In this section specific Markov chains are chosen and fed into the network during training. The first hypothesis is that the Markov chain approach works and is able to show that a lot of different input patterns can be learned by the network. This approach unveils an interesting behavior of the network regarding specific properties of the Markov chain. Therefore, based on the first results, new consecutive hypothesis will be formulated and tested.

7.1 Input dependent performance

Markov chain selection Since there are many combinations of possible Markov chains, in the beginning a discrete state space of size $n = 4$ is chosen with $S = \{A, B, C, D\}$. Furthermore 5 different Markov chains — called *models* in the following — were chosen, shown in figure 13. The idea was to start from a periodic Markov chain and systematically introduce more and

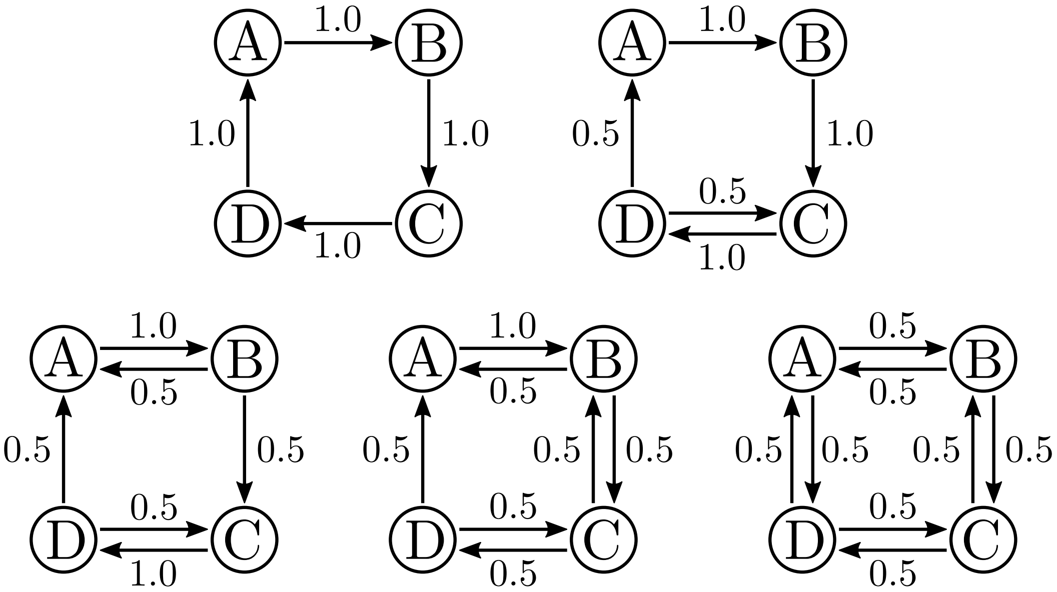


Figure 13: The first 4 state Markov models, denoted as model series I. Starting point is a periodic Markov chain, where from model to model one more state is split with a probability of $p_{ij} = 0.5$. The models are enumerated from left to right and from top to down from 1 to 5.

more backward flow with probability of $p_{ij} = 0.5$. It was hypothesized that:

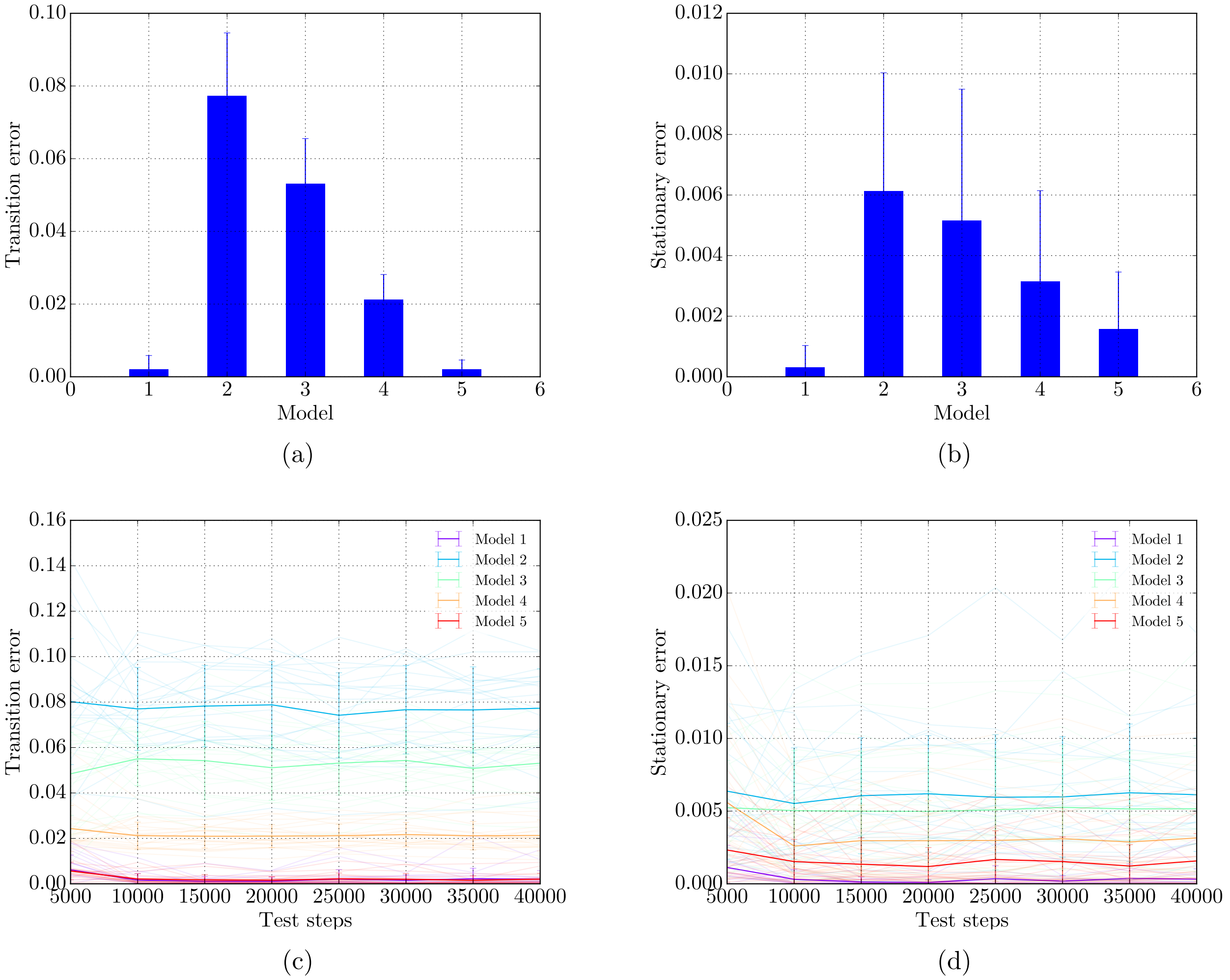
- A periodic Markov chain should be learned easily.
- The more states having probability backward, the harder it can be learned.

Performance In figures 14a and 14b the performance of the 5 models is shown. While the first hypothesis is true, the periodic Markov chain can be learned with high performance, the second hypothesis is not true. There is no relation between the number of backward directions and the performance. Furthermore, the performance differences between the models seem high. Results of t-tests show that all models differ highly significant with $p < 0.0001$ in transition performance ε_M, except between model 1 and 5 where $p = 0.986$, where the null hypothesis cannot be rejected. Table 2 shows the results of all t-test pairs. An example for a learned transition matrix is shown in table 3.

Additionally the performance over time was plotted in figures 14c and 14d, showing that the performance is stable over time in testing phase. This is reasonable, since spike-timing-dependent plasticity is not active in testing phase and the weights stay constant. To prove that STDP is the cause for the stable behavior, another simulation was done in appendix B, where STDP was switched on in the testing phase. And indeed, the information is lost by time and the network 'forgets' if STDP is active and the weights are still plastic. Furthermore, after the first testing chunk from 5,000 to 10,000 steps, there is a slight drop in error. This is due to a burn

Model	2	3	4	5
1	$1.303 \cdot 10^{-20}$ (18.51)	$1.803 \cdot 10^{-19}$ (17.14)	$4.656 \cdot 10^{-13}$ (10.73)	**0.986** (0.02)
2		$1.561 \cdot 10^{-05}$ (4.95)	$1.099 \cdot 10^{-15}$ (13.12)	$8.475 \cdot 10^{-21}$ (18.75)
3			$6.310 \cdot 10^{-12}$ (9.78)	$8.091 \cdot 10^{-20}$ (17.55)
4				$6.284 \cdot 10^{-14}$ (11.49)

Table 2: p-values of transition performance differences between Markov models. The upper value is the p-value, the lower value in brackets is the T value. Obviously all values are highly significant, except the bold value which represents the p-value of the performance difference between model 1 and 5.



(a) (b)

(c) (d)

Figure 14: Performance of first Markov models. The transition error ε_M of the first and last model is very low, as can be seen in in **a)**. The same holds for the stationary error ε_π in **b)**. The error bars in **a)** and **b)** indicate standard errors. The ε_M and ε_π correspond to the last chunk in the testing phase. The error over all chunks in the testing phase is shown in **c)** and **d)**.

in phase, which depends on η_{IP}. It takes some time until the spontaneous activity reaches a stable state. Details regarding the burn in phase are given in appendix F.

In conclusion, the main interesting effect is that the models seem to perform differently and there is currently no explanation why they behave in that way. It is striking, that model 1 and 5 seem very simple. Model 1 represents just a deterministic input pattern and Hartmann et al. (2015) already showed that such an input is represented quite well in the network. On the other hand, model 5 has kind of a ‘random’ structure. Except connections between A & C and B & D, everything else has no strong restrictions, whereas the other models have more restrictions. For example model 4: If the network reaches state A, it can only change to state B, which is a strong restriction, since all the other transitions are allowed to change to two directions. The results, shown in table 3, seem to support this explanation. It seems to be problematic to learn the connection between A and B, a lot of probability is ‘lost’ in this row.

The mean squared error from the transition matrix seems to be a sufficient measure. It is able to reflect the performance of the network and captures more information than the stationary distribution. Hence, in the following, only the performance measures obtained from the transition matrices ε_M are shown.

Plastic training It is very likely that the network performs better if the time for training is increased. If absolutely no plastic training is applied, meaning that $T_{\mathrm{plastic}} = 0$, the network can be seen as a reservoir network. The weights are initialized, but not adapted to the input in that case. The more steps T_{plastic} are chosen, the better the performance should be, due to the STDP

	Initial transition					Learned transition			
	A	B	C	D		A	B	C	D
A	0	1.0	0	0	A	0.02	**0.75**	0.02	0.21
B	0.5	0	0.5	0	B	**0.48**	0.04	**0.47**	0.01
C	0	0.5	0	0.5	C	0.02	**0.61**	0.03	**0.34**
D	0.5	0	0.5	0	D	**0.46**	0.05	**0.48**	0.01

Table 3: Example for a learned transition matrix. On the left side, model 4 is shown. The transition probabilities were used to train the network. On the right side, the learned transition matrix of model 4 is shown. The probabilities are calculated using equation (44). Bold values were initially non-zero. While most transitions are represented quite well by the network, the transition from A to B differs a lot.

mechanism. It could be that the $T_{\text{plastic}} = 50,000$ training steps are not enough for all models. Perhaps, model 2, 3 and 4 need a longer training to perform well. In order to understand the influence of the parameter, T_{plastic} was varied from $T_{\text{plastic}} = 0$ to $T_{\text{plastic}} = 80,000$. There are three hypothesis regarding the plastic training phase:

- All models improve in performance when training is extended and the performance increase is decreasing, the longer the training is conducted.
- Different models need different self-organizing time for a good performance.
- The longer the training period is, the closer the models get, regarding their performance.

The results in figure 15 show that the self-organizing phase has an important influence. The first hypothesis is partly true, since the performance increases, when more training is performed. But the increase in performance is saturating after about $T_{\text{plastic}} \approx 20,000$ steps of plastic

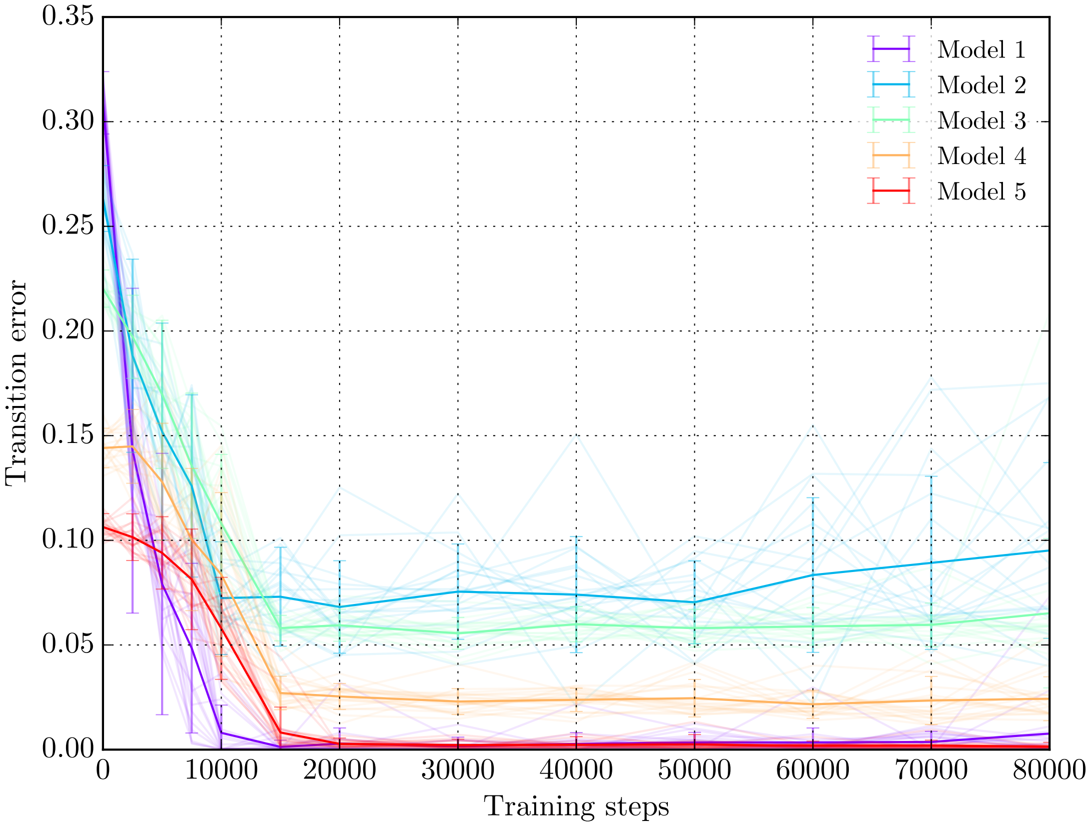


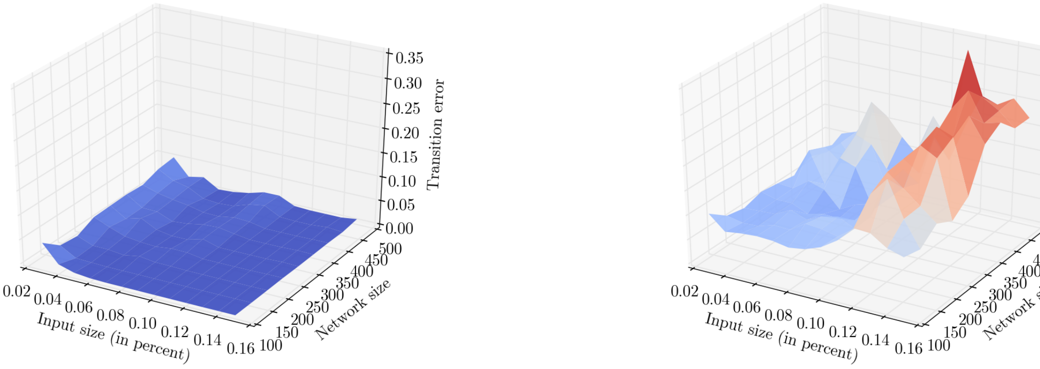
Figure 15: Influence of training to first Markov models. The transparent lines are all $N_{\text{sim}} = 20$ simulations per model. The opaque lines represent the error ε_M of a model per line for all simulations. Error bars indicate the standard error. Up to $T_{\text{plastic}} = 20,000$ the steps were increased by $5,000$, after that, they were increased in steps of $10,000$. Model 1 and 5 saturate earlier than the other models, but finally all models reach a specific performance level and do not approximate each other. Training is not able to explain the performance differences between the models.

training.

Model 1 and 5 seem to saturate earlier (at about $T_{\text{plastic}} \approx 10,000$) than the other models (at about $T_{\text{plastic}} \approx 15,000$ to $T_{\text{plastic}} \approx 20,000$). This behavior is in line with the second hypothesis. But since all of them are saturating, it contradicts the third hypothesis. At which point the network saturates, seems to depend slightly on the model, but still the previous differences remain after all. Differences in the length of the plastic training phase are not able to explain performance differences between models.

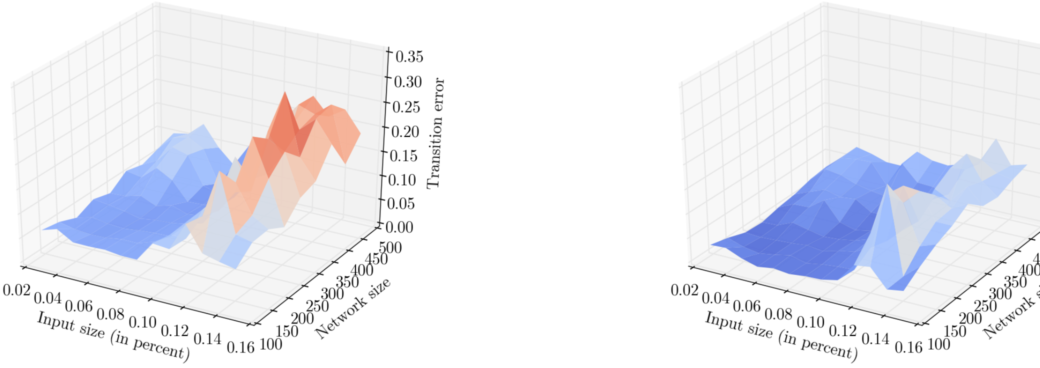
Size of the network and size of input cluster It is necessary to develop further ideas why the models have different performance in the same network. Another idea is to check the size of the network. It could be that — for some reason — some models need more neurons to find a good representation. It is important to also control the number of input connections to the network. If the network size is increased, but the number of connections between input neurons and excitatory neurons stays constant, it could be that the input information has too little influence, compared to ongoing spontaneous activity. In case of small input clusters, most of the network would just end up in spontaneous activity. Therefore, both parameters were varied at the same time. The network size was tested between $N^E = 100$ and $N^E = 500$ in steps of $\Delta N^E = 50$ and the number of input connections varied from $N^U_{x_i} = f_U \cdot N^E = 0.03 \cdot N^E$ to $N^U_{x_i} = 0.15 \cdot N^E$ in steps of $\Delta N^U_{x_i} = 0.01 \cdot N^E$. Since the number of inhibitory neurons depends on the number of excitatory neurons by $N^I = 0.2 \cdot N^E$, also the number of inhibitory neurons are changed.

In figure 16 the results are shown. Both parameters have an influence on the performance, but the mean performance is still different for the models. The initial parameters, proposed by Hartmann et al. (2015), of $N^E = 200$, $N^U = 0.2 \cdot 200 = 40$ and $N^U_{x_i} = 0.05 \cdot 200 = 10$ seem to have a good performance, compared to the other combinations. There are interesting effects worth to research on. There is a decay in performance for bigger networks, independent of the number of input connections. Furthermore, the performance decreases for more input connections, but the performance becomes better again for $f_U \geq 0.14$. But since the thesis focuses on the performance discrepancies between the different models, no further investigation was done. The main result is that the differences between the models could not be explained by the network size, nor by the size of the input clusters. While the optimal parameter combination



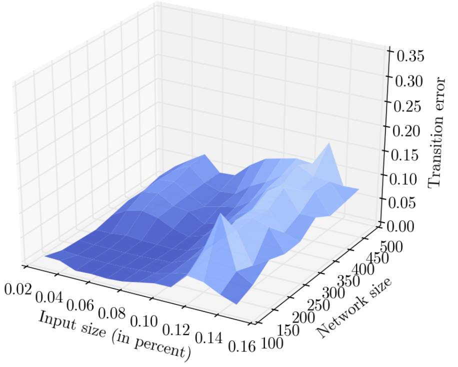
(a) Model 1

(b) Model 2



(c) Model 3

(d) Model 4



(e) Model 5

Figure 16: Shown is the performance of the 5 models, depending on the number of excitatory neurons N^E and the size of the input cluster for every input $N^U_{x_i}$. N^E was chosen from 100 to 500 in steps of $\Delta N^E = 50$. The size of the input cluster or equivalently the number of input-to-excitatory weights per input was chosen as a percentage of N^E for every state. It is defined as $N^U_{x_i} = f_U \cdot N^E$, where f_U was chosen between 0.03 and 0.15 in steps of $\Delta f_U = 0.01$. The two parameters do not explain the differences between the models.

is very similar for all models, the models still differ in their scales.

Classification method In section 6.4, the classification method was introduced. It was already mentioned that the original selection mechanism from Hartmann et al. (2015), to find the activity pattern representations R_{compare}, was changed. The original method was taking the last $\tilde{T}_{\text{compare}} = 2500$ steps of the no-plastic training phase. While working with the network, a bias in the classification method was found. The number of available comparison steps $T^{\text{min}}_{\text{compare}}$ differed a lot between the models. Perhaps, the classification algorithm is connected to the differences in the performance. While the updated mechanism was already explained in the methods section theoretically, the original and the updated method will be applied to the case of the 5 models in this section.

In models 1, 3 and 5 every state is equally probable, which can be seen by the stationary distribution, provided in table 4. The stationary distributions for all 5 models were calculated. The states A and B in model 2 are less equal, than the others. The same holds for state D in model 4. In the no-plastic training phase, those states will occur less often, which is elaborated in the following. Note that for the calculation of the stationary distribution of model 1, theorem 2 cannot be applied, since the model is periodic. In that case only theorem 3 can be applied and model 1 has no limiting distribution.

In average it is expected that every state occurs $\tilde{T}_{\text{compare}} \cdot \pi_{x_i}$. If all states have the same probability, it results in $\tilde{T}_{\text{compare}}/n = 2500/4 = 625$ steps for every state in the comparison phase. The calculation leads to $T^{\text{min}}_{\text{compare}} = \min\{\tilde{T}^{A}_{\text{compare}}, \tilde{T}^{B}_{\text{compare}}, \tilde{T}^{C}_{\text{compare}}, \tilde{T}^{D}_{\text{compare}}\} = \min\{625, 625, 625, 625\} = 625$ states in average, which are available as representative states

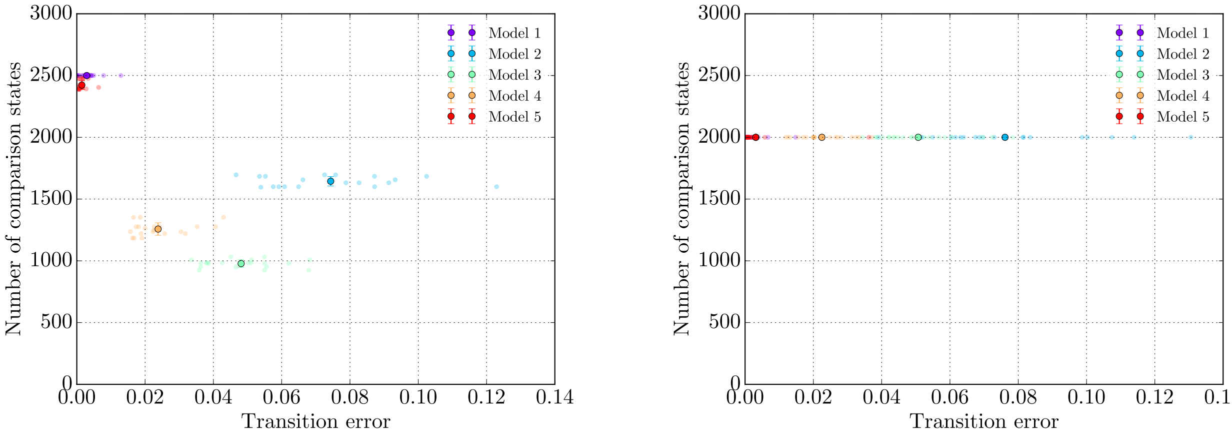
Model	π_A	π_B	π_C	π_D
1	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
2	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{2}{6}$	$\frac{2}{6}$
3	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
4	$\frac{2}{7}$	$\frac{3}{7}$	$\frac{2}{7}$	$\frac{1}{7}$
5	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

Table 4: Stationary distributions of first Markov models. In models 1, 3 and 5 all states have equal probabilities, in models 2 and 4 the probabilities of the states differ.

for the classification. This expected number of steps for every state holds for models 1, 3 and 5. But, for example, for model 2, the state with the lowest probability is state A and B with $\pi_A = \pi_B = 1/6$. Hence, both states are expected to occur in $2500 \cdot 1/6 \approx 417$ cases. In average the number of available steps for every state is given by $T^{\text{min}}_{\text{compare}} = \min\{417, 417, 833, 833\} = 417$. With an analogous calculation it can be shown that for model 4 only 357 steps are available to compare with. The number of available samples for classification is nearly half of the available samples, compared to the models where all states are equally probable.

It was hypothesized, that the original selection mechanism influences the performance systematically. There should be a correlation between $T^{\text{min}}_{\text{compare}}$ and the performance ε_M of the specific model. In figure 17a the overall number of available comparison steps $T^{\text{min}}_{\text{compare}} \cdot n$ is shown at the y-axis. The error ε_M is at the x-axis. The plot shows that there is indeed a slight negative effect. Of course, there is a ceiling effect since the highest possible $T^{\text{min}}_{\text{compare}}$ is 625, and therefore also $T^{\text{min}}_{\text{compare}} \cdot n$ has a limit. Additionally, the effect seems not perfectly clear, but still, the correlation is $r_{T\text{compare}} = -0.63$.

The updated mechanism fixes the number of $T_{\text{compare}} = 500$. An algorithm was searching for the last 500 representatives for every state in the no-plastic training phase, starting from the end. In that case, independent of the model, the same number of steps is available for classification. The number of necessary no-plastic training $\tilde{T}_{\text{compare}}$ steps is flexible in that case,



(a) Original method

(b) Updated method

Figure 17: Available steps for comparison $n \cdot T^{\text{min}}_{\text{compare}}$ with the original method and the updated method in relation to the performance. Transparent points are the single simulations, the opaque points represent their means. Error bars are added for the standard errors of $n \cdot T^{\text{min}}_{\text{compare}}$, but they are very small. The evaluation of the original method is shown in **a)**. The performance of the transition error seems to depend negative on the transition error, in tendency. Figure **b)** shows the result if $n \cdot T_{\text{compare}}$ is fixed. There is no effect by design.

which influences the choice of $T_{\text{noplastic}}$. In the old method the number of no-plastic training steps could easily be estimated, since they never exceeded $\tilde{T}_{\text{compare}} = 2500$. In the new solution the overall number $\tilde{T}_{\text{compare}}$ depends on the probability of that state which occurs least. An example was given in section 6.4.

The new classification method was also tested. The results with the updated method is shown in figure 17b. This time there is no correlation any more, which is expected, since the ceiling is always reached from every model by design. Lastly, the performance measures of all models were compared again with the new method. Surprisingly, even though the correlation is controlled, there is no clear change in the behavior. Hence, there has to be another mechanism responsible for the performance differences between the models. However, the idea to concentrate on the properties of the Markov chain, in terms of the stationary distribution, seemed to be an interesting approach to focus on.

Information of the stationary distribution In the previous section, the stationary distribution was used to obtain the probabilities for every state. While the research was focused on the parameters of the network only, like the training or size, it would also be possible to focus on the statistical properties of the Markov chain in order to understand what kind of Markov chain can be learned better. In other words, it would be interesting to focus on the properties of the input instead of the network. Since the probabilities indirectly had an effect on the performance in the classification mechanism, it could also be that they directly have an influence. In the previous section, those Markov chains had an influence, where some states

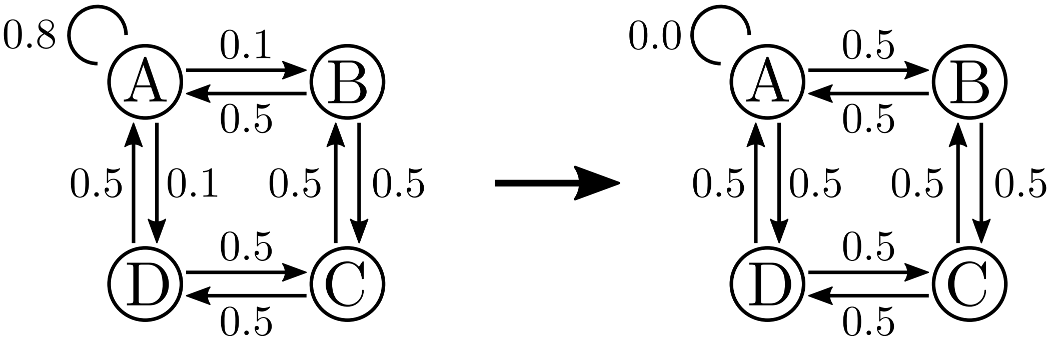


Figure 18: Model series II. This time, it is very probable that state A stays in state A, whereas B, C and D change between each other with equal probability (except B and D). Starting from this model, in several steps the probability p_{AA} is decreased to $p_{AA} = 0$ and p_{AB} and p_{AD} are increased to $p_{AB} = p_{AD} = 0.5$. The stationary distributions of all models are shown in table 5.

Model	π_A	π_B	π_C	π_D	σ_π^2	$D_{\mathrm{KL}}(\boldsymbol{\pi}\|\boldsymbol{\pi}')$
1	0.625	0.125	0.125	0.125	0.047	0.31
2	0.526	0.158	0.158	0.158	0.025	0.17
3	0.455	0.182	0.182	0.182	0.014	0.098
4	0.4	0.2	0.2	0.2	0.0075	0.054
5	0.357	0.214	0.214	0.214	0.0038	0.028
6	0.323	0.226	0.226	0.226	0.0018	0.013
7	0.294	0.235	0.235	0.235	0.0006	0.005
8	0.27	0.243	0.243	0.243	0.0001	0.001
9	0.25	0.25	0.25	0.25	0	0

Table 5: Stationary distributions of the Markov models from the second approach. Shown is also the variance and the Kullback-Leibler divergence of every stationary distribution.

had a lower probability than others. This is finally due to a higher variance or due to a higher deviation from the perfectly equally stationary distribution. The latter could be measured by a Kullback-Leibler divergence between a perfectly equally distributed stationary and the present stationary distribution. Both measures were introduced in section 5.4. It is hypothesized that Markov chains with different variance and Kullback-Leibler divergence vary in their performance systematically.

The initially chosen models do not systematically increase the variance σ_π^2 and Kullback-Leibler divergence $D_{\mathrm{KL}}(\boldsymbol{\pi}\|\boldsymbol{\pi}')$ in their stationary distribution, which can be seen in table 4. Therefore, a new bunch of models was invented to test the hypothesis. The new models are shown in figure 18. The first model has a high probability that state A stays in state A, namely $p_{AA} = 0.8$. The probability to leave A, p_{AB} and p_{AD}, is very small in the first model. Model by model p_{AA} decreases and p_{AB} as well as p_{AD} increase until all transitions have the same probability. The stationary distribution, variance and Kullback-Leibler divergence are shown in table 5 for every model. State A is very probable in the first model, compared to the probability of the other states, and increases the variance of the stationary distribution and the Kullback-Leibler divergence. Model by model it becomes less probable, until all states are equally probable, which results in $\sigma_\pi^2 = 0$ and $D_{\mathrm{KL}}(\boldsymbol{\pi}\|\boldsymbol{\pi}') = 0$.

The performance results are shown in figure 19a and 19b. The simulation was done with $T_{\text{plastic}} = 50,000$ plastic training steps and with $T_{\text{plastic}} = 0$ steps. The latter equals a static

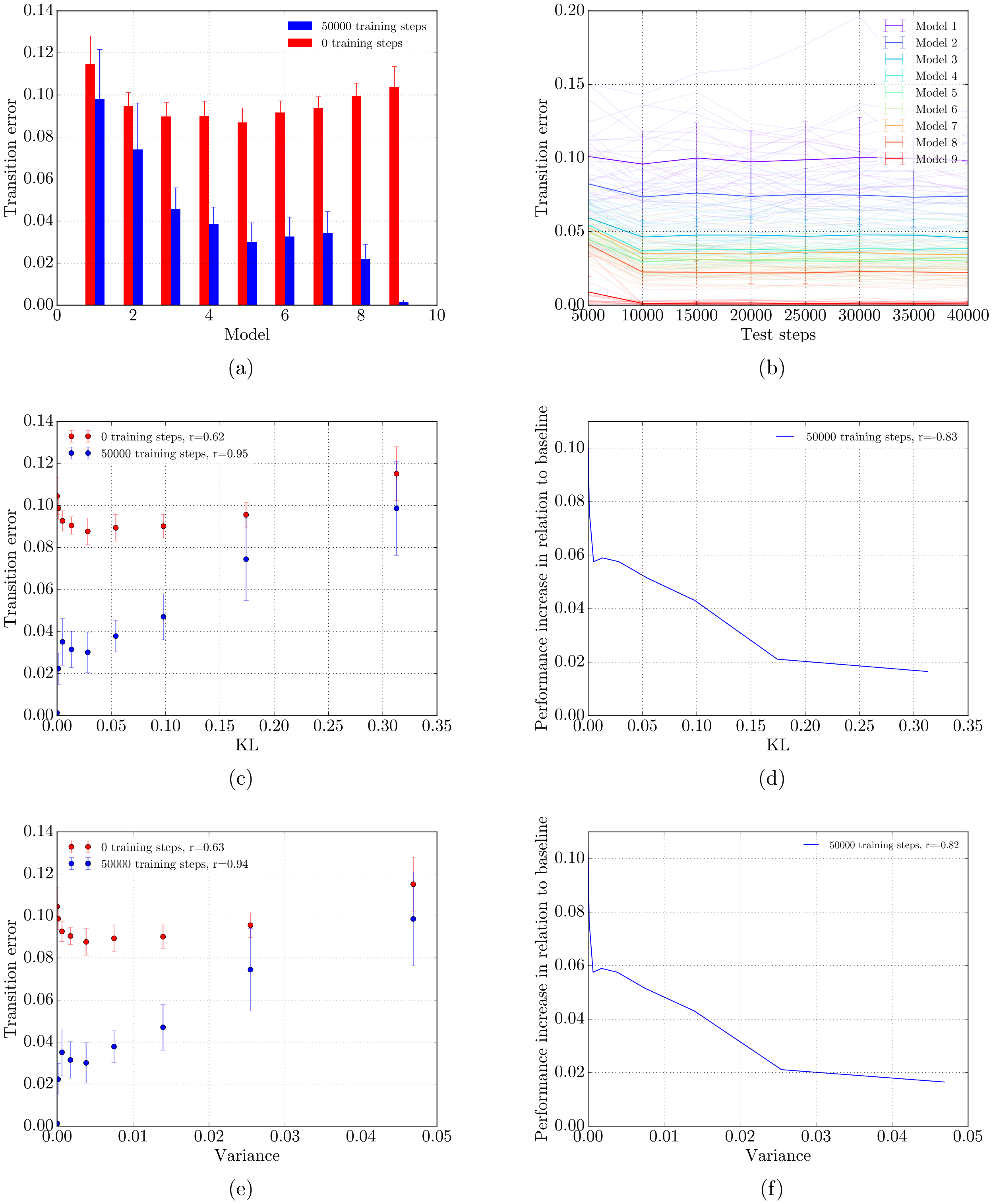


Figure 19: Performance and information of model series II. **a)** shows the performance of the 9 models. Blue bars show the performance with $50,000$ plastic training steps, red bars with no training, which equals a static reservoir. The performance behavior over the testing phase for the trained network is shown in **b)** and is constant, as it was expected from previous results. Plot **c)** shows the correlation between the Kullback-Leibler divergence and the transition error ε_M. Correlations are given in the legend. In **d)** the increase in performance from the models with $50,000$ plastic training steps in relation to the baseline with 0 training steps is presented. The correlation is shown in the legend. Plot **e)** and **f)** are analog to **c)** and **d)** but with the variance of the stationary distribution instead of the Kullback-Leibler divergence.

reservoir network, since the weights are just randomly initialized and not adapted using STDP. Interestingly, the static reservoir network behaves nearly equal for every model, which is opposite to the hypothesis of a systematic influence. On the other hand, the networks, where the weights are trained, show a clear rise in performance, when the stationary distribution becomes more equal. Figure 19b shows that the performance of the network stays constant in the testing phase. It reproduces the result from the previous models.

In figure 19c the MSE ε_M is plotted, depending on $D_{\mathrm{KL}}(\boldsymbol{\pi}\|\boldsymbol{\pi}')$. For the static reservoir, there is no clear linear tendency for a relationship. Regarding the trained network, a linear tendency can be seen and the correlation is $r_{D_{KL}} = 0.95$. The same holds for a relation between the performance and the variance of the stationary distribution with a correlation of $r_\sigma = 0.94$, shown in figure 19e.

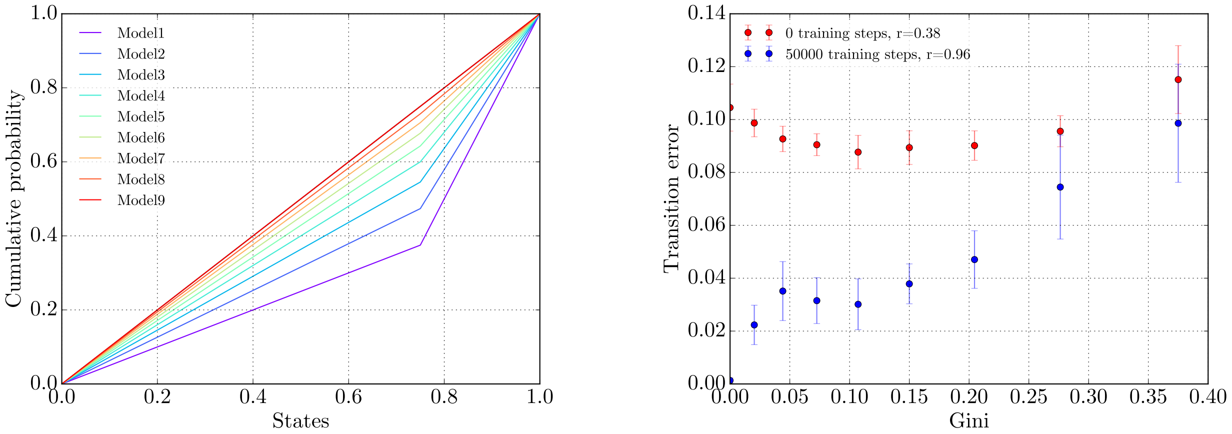
The performance of the static reservoir network seems to be kind of a *baseline*, where training is able to improve the performance as long as the stationary distribution is not too unequal. In figures 19d and 19f, the increase in performance is shown in relation to the information measures. The more equal the stationary distribution, the higher is the gain in performance.

Finally, the relation between ε_M and σ_π^2 as well as $D_{\mathrm{KL}}(\boldsymbol{\pi}\|\boldsymbol{\pi}')$ seems to be non-linear for small variances and small values of Kullback-Leibler divergence. The behavior in that area is evaluated in appendix E.

Concentration of the distribution While focusing on the information of the stationary distribution in the previous section, it could also be that the concentration of the distribution plays a major role. The results should be similar, since a high concentration normally also includes an increase in variance.

To quantify the concentration of a distribution, the Lorenz curve can be used, which was introduced in section 5.4 and illustrated in figure 11. While the Lorenz curve is a qualitative assessment, the Gini coefficient G, shown in equation (32), quantifies the concentration.

Figure 20a shows the Lorenz curves for the models used above. The amount of concentration obviously differs between the different models. The relation between the performance ε_M and the Gini coefficient G is shown in figure 20b. The effect is quite the same as it was before with the variance and the Kullback-Leibler divergence. Using the Gini coefficient, the correlation



(a) (b)

Figure 20: Concentration of the stationary distribution. **a)** shows the Lorenz curves of all models from the second approach (figure 18). The models differ in their concentration. In **b)** the Gini coefficients are calculated to quantify the concentration measure. They are put in relation with the performance of the models.

is $r_G = 0.96$. Therefore, the concentration of the distribution is just another perspective, but it does not explain more than the former approaches were doing. However, to evaluate the observations under a perspective of concentration had a decisive impact on the idea of the following IP-hypothesis.

7.2 IP-hypothesis

The results from the previous section show a clear relation between the properties of the Markov chain and the performance. It is necessary to develop an explanation for this behavior and it remains to show that this behavior is robust, also for other Markov chains.

The hypothesis, which is suggested to explain the effect, focuses on the intrinsic plasticity (IP). Therefore, the hypothesis is called *IP-hypothesis* in the following. Assume just 3 states, A, B and C. They correspond to input clusters at the excitatory neurons. The case is shown in figure 21. Inhibitory neurons, input neurons and weights are not shown. Further, assume a stationary distribution

$$\pi = (\pi_A, \pi_B, \pi_C)^T = (0.8, 0.1, 0.1)^T. \tag{45}$$

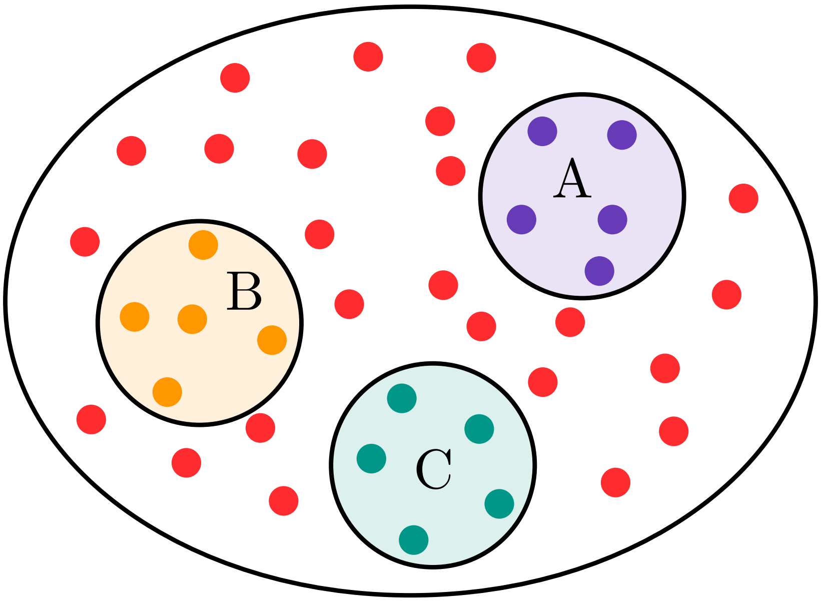


Figure 21: Three input clusters as an example to illustrate IP-hypothesis. If one cluster is not active for a longer time, neurons of that cluster will be active spontaneously and the chance for a misclassification increases.

A simulation of a Markov chain with such a distribution and the three states could look like

$$A\ A\ A\ A\ A\ A\ A\ C\ B\ A. \tag{46}$$

It is necessary to remind the property of IP regarding spontaneous activity, as it was shown in figure 7 in section 3.5. If a neuron is not active for a long time, the threshold of that neuron is going down. If the threshold falls below zero, the neuron fires spontaneously. In the simulation, state A and therefore neuron cluster A is repetitively active in the network for a long time, due to its higher probability π_A. The threshold of the other neurons is going down at that time, because they are ideally not excited. That affects those neurons of cluster B and C in particular. Therefore, the chance rises that cluster B or C will be active spontaneously after some time.

Models with regular input patterns, meaning that the probabilities of the states are similar, have a better performance than those with irregular input patterns, where some states have a smaller probability than others. The IP-hypothesis suggests that intrinsic plasticity is a cause for that observation. This hypothesis was tested in three steps:

- Apply different models and evaluate if the effects can be explained by the hypothesis.
- Vary parameter $\bar{H}^{\mathrm{IP}}$ and evaluate if different levels of intrinsic plasticity influence the

performance.

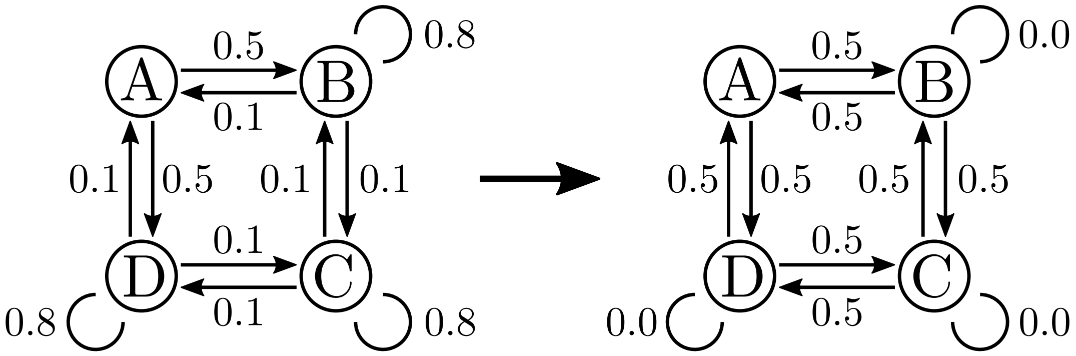
- Vary parameter σ_{IP} and evaluate if the performance becomes more robust, independent of the model.

More models After the effect of the stationary distribution was shown, a bunch of different models was developed to test the IP-hypothesis. In figure 22 four series of Markov chains are shown. They were all simulated with $T_{\text{plastic}} = 50,000$.

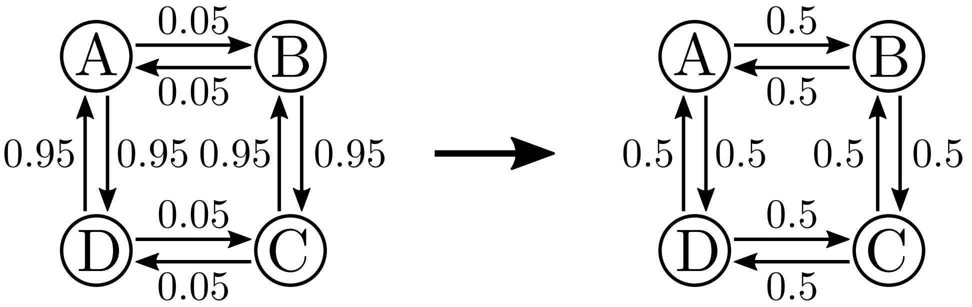
The first series in figure 22a, model series III, is similar to the series of models II, presented in figure 13. But while the previous models concentrated their probability at state A, in this case the probability is concentrated at states B, C and D. Therefore, the chance that A will be active spontaneously is increased, since this state is less active than the other states. The performance plot should show similar results to figure 19a and indeed, figure 23a shows that the results are very similar.

Model series IV, shown in figure 22b, separates the states A & D and B & C and join them step by step again. The interesting aspect of this approach is that the stationary distribution is absolutely equal for every model in this series, such that $\boldsymbol{\pi} = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)^T$. At that point the predictors, namely the variance, Kullback-Leibler divergence and Gini coefficient, are the same for every model and would predict no differences in performance. On the other hand, the IP-hypothesis predicts a change in performance. In the beginning, the probability to switch between the right and the left side is very small. Therefore, if one side is active, the chance increases that the other side will be active spontaneously. The more both sides are connected again, the better the performance should become. Figure 23b shows a clear decay in the MSE ε_M, which strongly supports the IP-hypothesis. The down side of this result is, that the information and concentration measures do not longer hold as predictors. The information, contained in the stationary distribution is obviously not sufficient to predict the behavior.

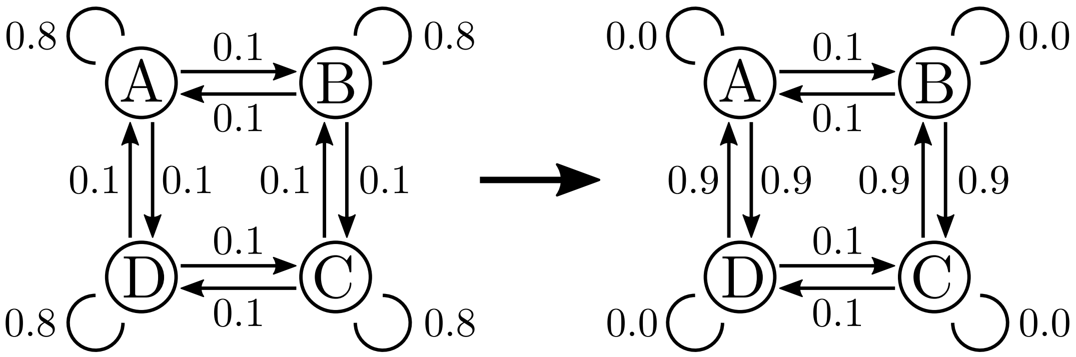
Series V is more complex. In figure 22c the construction of the models is shown. It starts with a model where all four states are quite disconnected. It is expected that the performance is low in this case. The last model of this series equals the first model of the previous series. The model transforms from four separated parts to two separated parts. Comparing the performance of the first models and the last models in figure 23b, the error is quite similar. Therefore, it seems not very important if the chain is separated in four parts or in two parts. It seems to be more



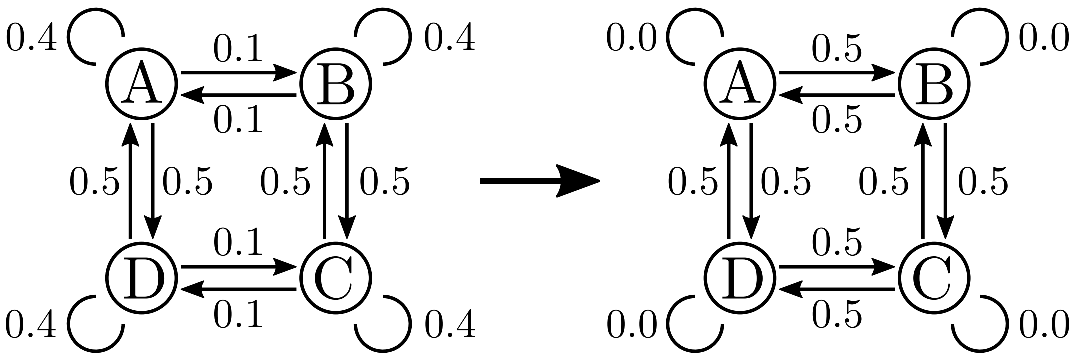
(a) Model series III



(b) Model series IV

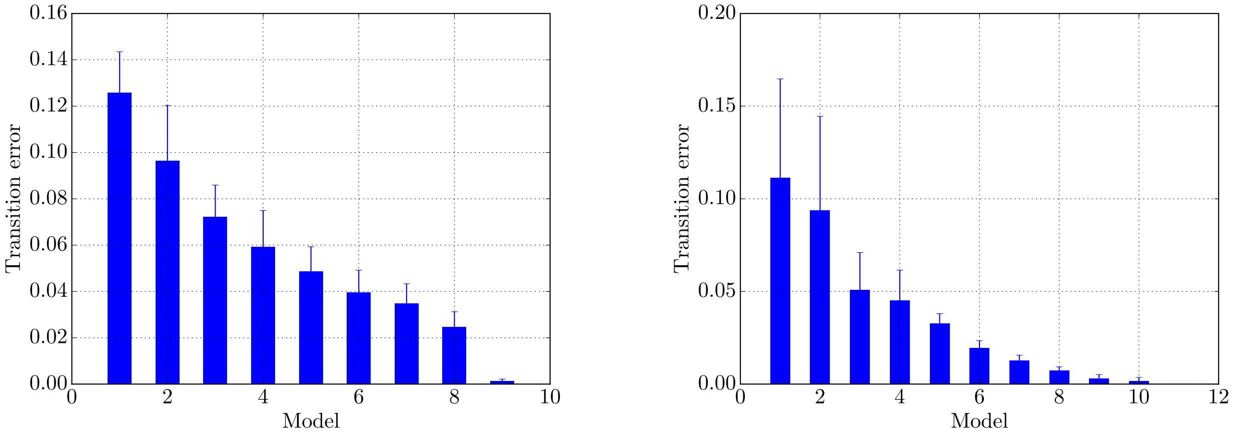


(c) Model series V



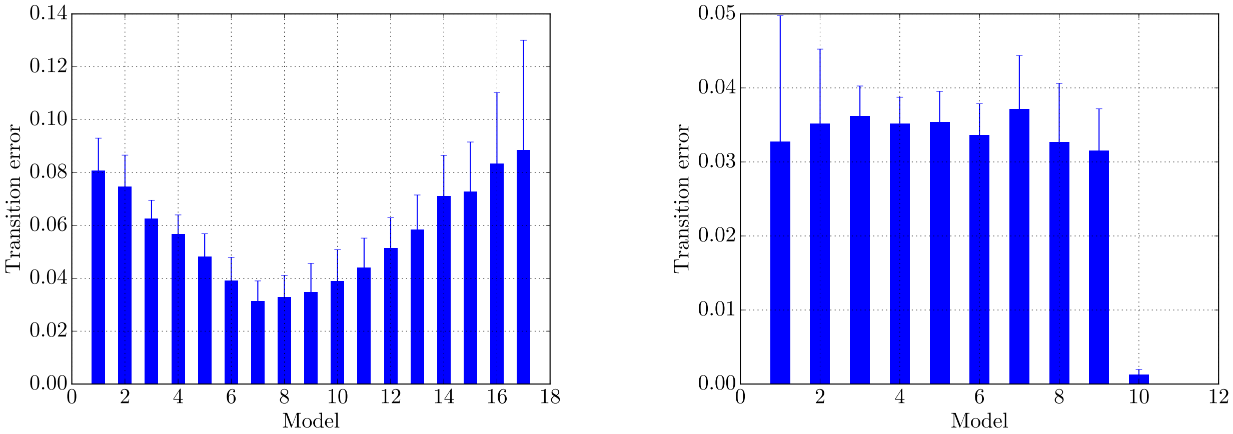
(d) Model series VI

Figure 22: Four Markov model series to test the IP-hypothesis. The first series in **a)** prefers state B, C and D. Model by model, the self loop is decreasing and the probability to reach A is increased. **b)** starts with a model which separates between the left and the right side. Series **c)** starts with a model where all four states are separated and changes step by step to a model where the right and left side are separated, which was the starting point in **b)**. Finally, in **d)** the first model separates between right and left and has some self loops additionally. This equals a middle model from **c)**. In series **b)**, **c)** and **d)** all models have the same stationary distribution with equal probability for every state.



(a) Model series III

(b) Model series IV



(c) Model series V

(d) Model series VI

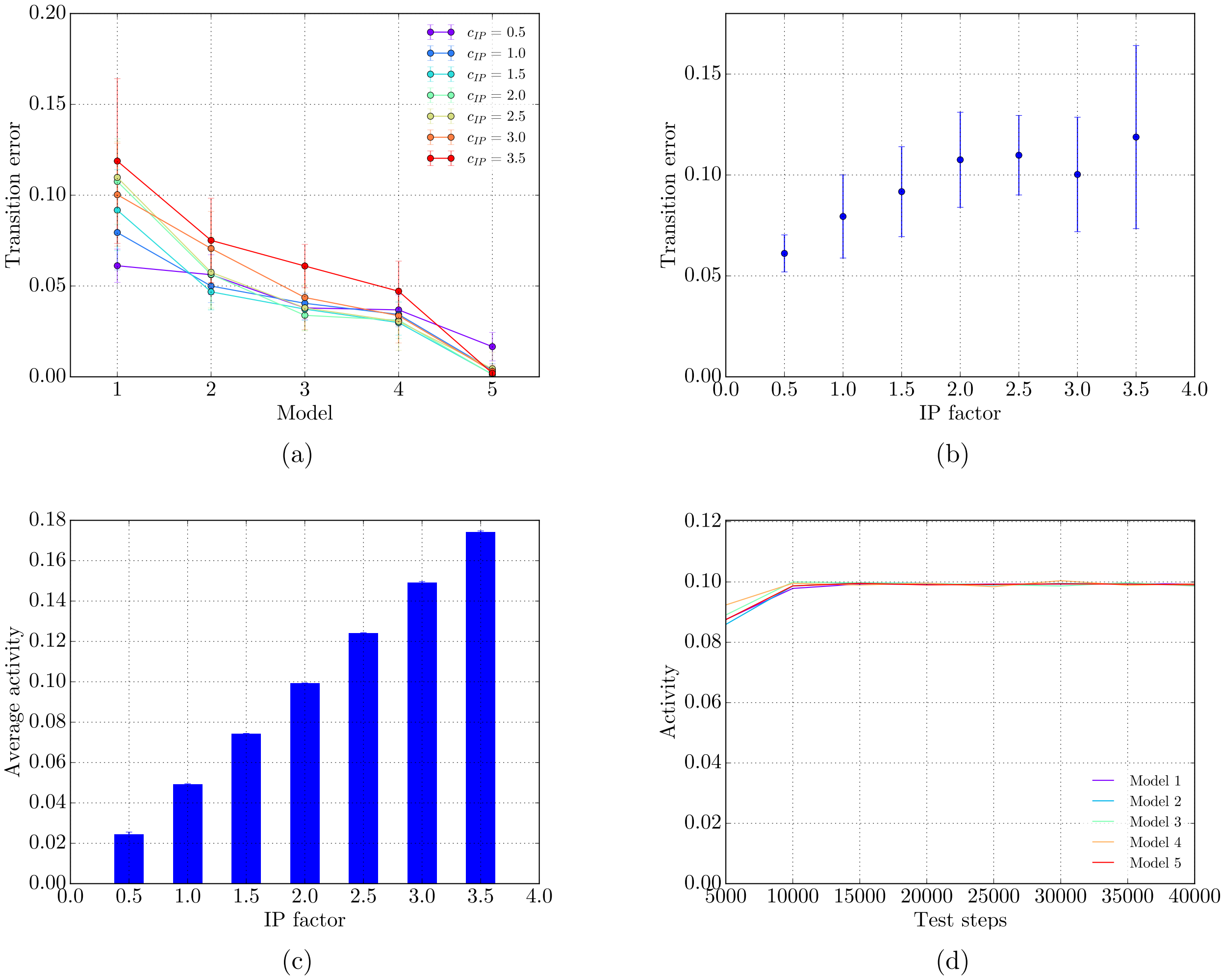
Figure 23: Performance of the four Markov model series from figure 22. Model series III was divided into 9 models, model series IV into 11 models, model series V into 17 models and model series VI into 11 models. All four series follow the predicted behavior according to the IP-hypothesis. The variance, the Kullback-Leibler divergence and the Gini coefficient, do not hold for prediction in cases **b)**, **c)** and **d)**, since every information and concentration measure is equal within every model series, while the performance differs significantly.

important how strongly the different parts are connected. Interestingly for the middle part, the error decreases. The middle model has a probability of $p_{x_i x_i} = 0.4$ for the self loop and $p_{AD} = p_{BC} = 0.4$. Probably the self loop increases the probability that the states can change between the right and the left side, since in some cases it stays in the same state and has another chance to switch. The first model suffers from a strong disconnection between all states. For those 'islands' the self loop is not helpful. The last model has no self loops any more, which separates the the sides, as it was observed in the model series above. Note that the stationary distributions are the same for all models again, where each state has equal probability.

The last example, model series VI, is shown in figure 22d and again, all models in this series have the same distribution as before. In this case, the first model starts, where the third series was in its middle. At that point the performance was relatively good. The last model was already used before in many cases and is already known as a model which can be learned with high precision. It is predicted from what was observed before, that the self loop in the first models increase the probability that the states can switch the sides. The more the self loop probability is decreased, the more the direct connection between the two sides increase at the same time. This behavior should compensate the decrease in the self loops. It should result in a relatively similar performance for all models. Indeed, figure 23d shows that all models have a relatively constant and low error, where the last model is a little bit outstanding. Comparing this outlier model for example with 23a, indicates that this effect already happened before. A possible explanation is given in appendix E.

In conclusion, the IP-hypothesis holds for all tested model series. In the following, the implementation of the intrinsic plasticity mechanism from equation (34) is tested for different parameters to come closer to an understanding of how IP works.

Influence of parameter $\bar{H}^{\mathbf{IP}}$ If the intrinsic plasticity is responsible for the performance differences between the models, then a change of the parameters of the IP mechanism should be reflected in the performance of the models. $\bar{H}^{\mathrm{IP}} = 2 \cdot N^U / N^E$ is the main factor inducing spontaneous activity. The lower the target rate $\bar{H}^{\mathrm{IP}}$, the less spontaneous activity is present and vice versa. Models, where specific states are less probable, should profit from a lower target rate. Those states can stay silent for a longer time until intrinsic plasticity will spontaneously activate them. Since the IP is the only source for activity in the testing phase, it is somehow



(a) (b) (c) (d)

Figure 24: Influence of the target rate on model series II, divided into 5 models. The target rate factor c_{IP} was varied from 0.5 to 3.5 in steps of $\Delta c_{\text{IP}} = 0.5$. **a)** shows the behavior for all models, where the performances of the first models seem to be influenced by the target rate. In **b)** a further investigation of the very first model was done, since effects are expected to be most clear for that model. It shows that the target rate has an effect on the performance for that model, where p-values are given in table 6. The average activity of the network for the first model is plotted in **c)**, showing that the target rate directly influences the activity in the network. In plot **d)** the activity for $c_{\text{IP}} = 2.0$ is shown over the whole testing phase for all models. After a short burn in phase, the activity stays very constant. Note that this plot bases only on a single simulation. In all cases, errorbars indicate standard errors.

problematic to change the parameter. The IP mechanism directly changes the average activity in the network. It cannot be excluded that it is the activity level in general rather than the threshold of the mechanism which influences changes in performance. For the simulation a parameter c_{IP} was used, such that $\bar{H}^{\text{IP}} = c_{\text{IP}} \cdot N^U / N^E$ where $c_{\text{IP}} = 2$ is the default value.

However, the results with different c_{IP} values are shown in figure 24. It was applied to the systematic model series II from figure 18. For simplicity of the presentation of the results in the plots, the series was created with just 5 models in opposite to results shown in figure 19c,

c_{IP}	1.0	1.5	2.0	2.5	3.0	3.5
0.5	$\mathbf{1.08 \cdot 10^{-3}}$ (∗∗)	$\mathbf{2.46 \cdot 10^{-6}}$ (∗ ∗ ∗)	$\mathbf{1.12 \cdot 10^{-9}}$ (∗ ∗ ∗)	$\mathbf{6.61 \cdot 10^{-12}}$ (∗ ∗ ∗)	$\mathbf{1.37 \cdot 10^{-6}}$ (∗ ∗ ∗)	$\mathbf{3.44 \cdot 10^{-6}}$ (∗ ∗ ∗)
1.0		0.085 (ns)	$\mathbf{3.62 \cdot 10^{-4}}$ (∗ ∗ ∗)	$\mathbf{4.01 \cdot 10^{-5}}$ (∗ ∗ ∗)	**0.014** (∗)	$\mathbf{1.41 \cdot 10^{-3}}$ (∗∗)
1.5			**0.040** (∗)	**0.011** (∗)	0.310 (ns)	**0.025** (∗)
2.0				0.750 (ns)	0.395 (ns)	0.343 (ns)
2.5					0.236 (ns)	0.433 (ns)
3.0						0.139 (ns)

Table 6: p-values of performance differences for simulations with different target rate factors c_{IP} for model 1. Bold values indicate significant differences with $p \leq 0.05$.

where the series was applied with 9 models. The first plot in figure 24a shows the performance of all 5 models and all c_{IP} values. As expected from the results before, the error decreases when state A is less isolated. In those models, where state A is highly isolated, the IP target rate influences the performance of the models. For evaluation, the first model was chosen. This model has the lowest probability to reach state A and therefore the target rate should have the highest influence to that model. In figure 24b the performance results for the first model are shown, depending on the IP factor c_{IP}. It seems that there is a positive correlation between the parameter and the error ε_M, at least between $c_{\mathrm{IP}} = 0.5$ and $c_{\mathrm{IP}} = 2.0$. In table 6, t-tests between all errors value were applied, showing that the intuition is correct. The main effect happens before $c_{\mathrm{IP}} = 2.0$, afterwards no significant changes can be observed. This results supports the IP-hypothesis, even though the effects seem to decay very fast with further models.

Figure 24c shows the average activity for the first model, depending on the target rate. It is clear from that plot that $\bar{H}^{\mathrm{IP}}$ directly influences the amount of activity, which is defined as the number of spiking excitatory neurons $n_{\mathrm{spiking}}(t) = |\ \{i \in \{1, ..., N^E\}\ :\ x_i(t) = 1\}|$ divided by the total number of excitatory neurons N^E at a specific point in time t. Additionally, for a fixed c_{IP}, the activity is very stable and independent of the model, as shown in figure 24d. It shows the activity over the testing phase for $c_{\mathrm{IP}} = 2.0$ for a single simulation. Note that this activity is not averaged. The activity differences could potentially influence the performance directly.

To exclude effects from the amount of activity in the networks, noise could be implemented to hold a specific level of activity. This approach is considered in the discussion in section 8.6.

However, even though the effects are not very strong and possibly influenced by the activity, in tendency, they support the IP-hypothesis.

Influence of parameter $\sigma^{\mathbf{IP}}$ Hartmann et al. (2015) have shown that the network performs more robust if the target rate is not the same for every neuron. Therefore, in this thesis the parameter σ^{IP} was introduced in section 6.2 and chosen as 0.01, pursuant to their suggestion. The parameter determines the range, how much a target rate of a specific neuron can differ from the average target rate $\bar{H}^{\mathrm{IP}}$. According to Hartmann et al. (2015), the performance should decrease if σ^{IP} is lower than 0.01 or even zero. The latter case with $\sigma^{\mathrm{IP}} = 0$ corresponds to the rule Lazar et al. (2009) used in the initial SORN. Beside reproducing the behavior for $\sigma^{\mathrm{IP}} < 0.01$, it also remains to show what happens if σ^{IP} is increased above 0.01 and to evaluate the results in context of the IP-hypothesis.

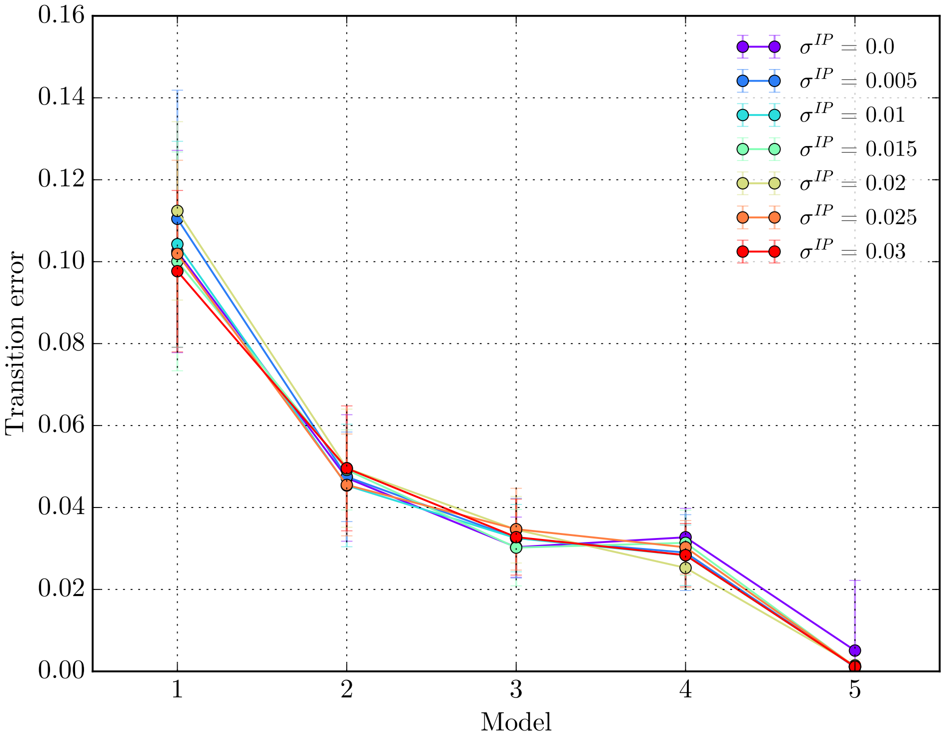


Figure 25: Variation of IP target rate range σ^{IP} applied to model series II, divided into 5 models. The target rate range was varied from 0.0 to 0.03 with steps of $\Delta\sigma^{\mathrm{IP}} = 0.05$. Error bars indicate standard errors. The performance is independent from the chosen target rate range for all models.

To keep it comparable with the variation of the $\bar{H}^{\mathrm{IP}}$ parameter, also the systematic model series II from figure 18 was applied, divided into 5 models. Figure 25 shows the performance of all 5 models for all σ^{IP} values. The differences in the performance are not significant ($p > 0.05$) for all models, except one, where the difference between $\sigma^{\mathrm{IP}} = 0.02$ and $\sigma^{\mathrm{IP}} = 0.03$ in the first model is significant with $p = 0.035$. But since the significance level is low and all other 20 comparisons for the first model alone are non-significant, the results from Hartmann et al. (2015) could not be reproduced. Contrary to their findings, the present findings suggest no effect below $\sigma^{\mathrm{IP}} < 0.01$, nor is an effect above $\sigma^{\mathrm{IP}} > 0.01$. The result suggests that the initial IP rules from Lazar et al. (2009) is sufficient. Regarding the IP-hypothesis, the σ^{IP} parameter seems to have no or at best a very small influence on the less probable states. Since the arguments of Hartmann et al. (2015) for introducing σ^{IP} are plausible, perhaps further research is necessary to understand in which cases a variation of the parameter for the neurons can increase the robustness. A suggestion for another approach is given in the discussion section, in subsection 8.6.

Other parameters Two other aspects should be considered: the learning rate η_{IP} and the connectivity of the network.

First, η_{IP} is the learning rate of the intrinsic plasticity. The higher the learning rate, the faster the network reaches its target rate $\bar{H}^{\mathrm{IP}}$. It should not influence the performance of the models, since the burn in phase was always excluded from evaluation. However, the effect of η_{IP} regarding the burn in phase is considered in appendix F.

Beside effects from the intrinsic plasticity, it is also necessary that the neurons are adequately connected. It could be that this assumption does not hold. Under some conditions, the structure of the network may be rather good, under others rather bad. An approach to test if such effects possibly occur, is to vary the connectivity of the network. It is the average number of connections between the excitatory neurons, denoted by $\lambda^{W} = \rho \cdot N^{E}$, where $\rho = 0.1$ by default. If ρ is changed, the number of average connections changes. According to the appendix section of Lazar et al. (2009) a sparse network with about 10% of connections performs best. It means that every neuron is connected with 10% of the other excitatory neurons in the network. If the number of connections is too high, there is a higher risk that the activity in the network grows without bounds. In context of the IP-hypothesis, it is probably worth to vary the connectivity

ρ again in the setting of the Markov chain approach. The results are shown in appendix G. It was possible to reproduce former results and a connectivity of 10% seems to fit well, also in context of the IP hypothesis. If the connectivity is too low or too high, the error increases.

8 Discussion

Self-organizing recurrent neural networks are a promising approach in modeling brain processes. SORN extends randomly initialized reservoir networks with binary threshold neurons by using three biologically motivated plasticity mechanisms. It has already been shown that this self-organizing network outperforms static reservoir networks and is able to reproduce experimental findings, shown in section 4.5. Lazar et al. (2009) stated that many basic mechanisms, like plasticity mechanisms, are more and more researched. Even though those mechanisms still lack a deep understanding, it is highly interesting how they work together in a balanced system. The hope is to get a deeper understanding of basic processes of dynamic non-linear systems, like the brain.

Since the number of neurons in the network is relatively small, in a scale of hundreds, the applications are limited to sensory or short-term processes. Indeed, when spike-timing-dependent plasticity was active in the testing phase, the network tends to forget the stored information by time. It should be possible to build larger networks using SORN, for example in combination with cell assemblies. At present, cell assemblies are often build with static reservoir networks, but first dynamic approaches already exist (Dasgupta, Tetzlaff, Kulvicius, & Wörgötter, 2015; Tetzlaff, Dasgupta, Kulvicius, & Wörgötter, 2015).

However, this thesis focuses on a small sized network and its properties. The combination of the plasticity rules led to the interesting finding, that different input patterns highly influence how well the network is able to store and reproduce this input. Furthermore, it was hypothesized that intrinsic plasticity plays in important role for those results.

8.1 Perspectives on SORN

There are severals mathematical perspectives to interpret SORN. Three variants could be collected.

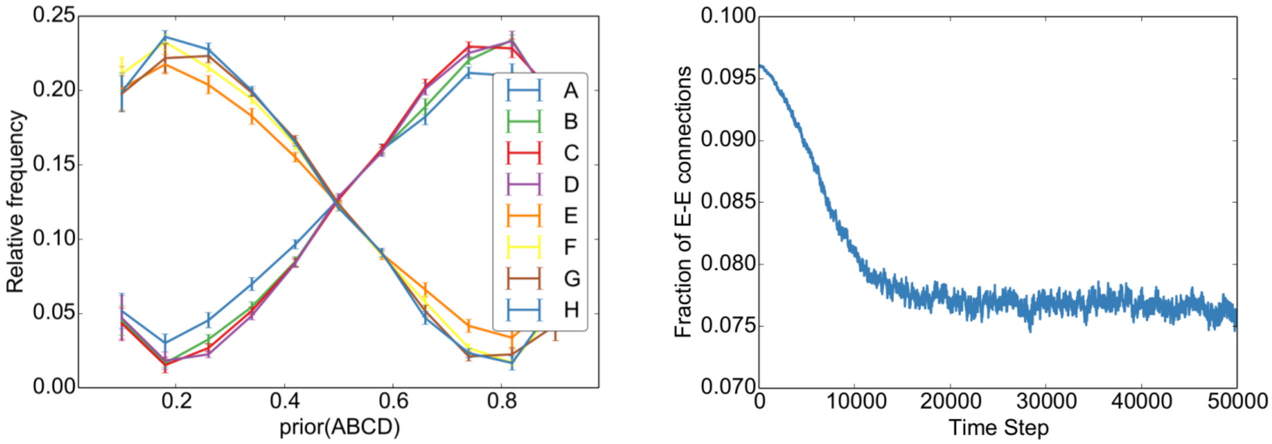
- Trajectory in high dimensional space: The (spontaneous) activity in the network, with its N^E neurons, can be seen as a trajectory in an N^E-dimensional space. This perspective has similarities with kernels. The input is transformed in a high dimensional time-dependent feature space, which was already discussed in section 4.3.

- Markov chain Monte Carlo sampling: If the time-dependent input pattern is modeled with a Markov chain, the network learns from Markov chain samples. Furthermore, in testing phase, the network reproduces those samples.
- Statistical estimator: Again, if SORN is trained, using a distribution for the input neurons, the network can be seen as an estimator for this distribution. The parameters of the network determine the quality of the estimator. If the underlying Markov chain is unknown, it can be estimated by the (spontaneous) activity patterns of the network.

Regarding the sampling perspective, the samples from the Markov chain could be biased while training. According to the law of large numbers, after a long time the average converges to the Markov distribution. But in finite time steps, it is possible to have large runaways. Therefore, it is not guaranteed that the network was trained with a perfect Markov chain. This effect could be stronger in those networks, where some states are repeated very often. To put it in other words, in testing phase, the network does not reproduce the Markov chain, it reproduces the *samples* from the training phases, which were samples from the Markov chain. However, since the results are averaged over N_{sim} runs and the standard error was calculated, this influence is assessable.

8.2 Reverse learning effect

Hartmann et al. (2015) applied different probabilities to 8 input states. States A, B, C and D occurred with a probability between $\frac{0.1}{4} = 0.025$ and $\frac{0.9}{4} = 0.225$ for each state. In contrary, states E, F, G and H were active with the reverse probability between $\frac{0.9}{4}$ and $\frac{0.1}{4}$. The probabilities were changed in steps of 0.1. In figure 26a the results are shown. Between 0.2 and 0.8, the network was able to reproduce the initial probabilities with a small amount of over representation. But at 0.1/0.9, the effect was reversed and the variance increased. Hartmann et al. (2015) argued that this effect is due to ‘pathological network dynamics’, while a specific explanation was missing. It is suggested that the IP-hypothesis is able to explain the reverse effect. Probably, the states were changing fast enough and avoiding distorting spontaneous activity in the range between 0.2 and 0.8. After that, it is likely that the threshold becomes smaller than zero, before the state is activated again according to the rule.



(a) States A, B, C & D and E, F, G & H were learned with probabilities between 0.1 and 0.9. At the extreme points, a reverse effect in the representation was observed (Hartmann, Lazar, Nessler, & Triesch, 2015, figure 4c). It can possibly be explained with intrinsic plasticity.

(b) The fraction of excitatory-to-excitatory connections, where $w_{ij} > 0$ during plastic training, stabilizes after about $20,000$ steps (Hartmann, Lazar, Nessler, & Triesch, 2015, figure 1d). It is assumed to correspond to the error decay in the first $20,000$ steps of training.

Figure 26: Reverse learning effect and fraction of excitatory-to-excitatory connections from Hartmann, Lazar, Nessler, and Triesch (2015).

8.3 Biological plausibility

The question arises if the differences in the learning performance are due to SORN alone or if there are indications that this behavior can also be observed in experiments where responses from mammal brains are recorded. In psychology, an experimental design, called *oddball paradigm*, is often used for recording the neural behavior of time-dependent stimulus patterns. It consists of a *standard* stimulus, which is repeated very often, and a *deviant*, which breaks the previous stimuli. If the electrophysiological or magnetophysiological activity of the brain is recorded, a deviant elicits an *event-related potential* (ERP) or more specific a *mismatch negativity* (MMN). The electro- or magnetophysiological activity differs significantly from normal activity when a deviant stimulus is presented (Näätänen, Gaillard, & Mäntysalo, 1978).

The processes, underlying the MMN mechanism, is assumed to be pre-attentive and on a sensory memory level (Tiitinen, May, Reinikainen, & Näätänen, 1994). While the MMN was obtained with simple stimulus rules, Paavilainen, Arajärvi, and Takegata (2007) and Bendixen, Prinz, Horváth, Trujillo-Barreto, and Schröger (2008) were able to show that also more complex stimulus rules can be learned and detected by the brain. Such a rule is shown in figure 27. The learning of those complex rules is not accessible to attentive processing, which emphasizes that

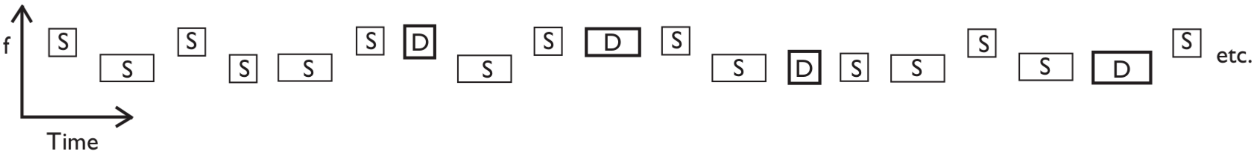


Figure 27: Complex rule with auditory stimuli used in oddball paradigm (Paavilainen, Arajärvi, & Takegata, 2007, figure 1). Long tones are followed by high tones (high frequency f) and short tones are followed by low tones. S indicates a standard stimuli, fitting to the rules, D indicates a deviant, breaking the rule.

the detection mechanisms is connected to processing on sensory level. As already stated above, SORN has also characteristics of a sensory or short-term memory. Thus, a parallel between those experiments with humans and the simulation can be drawn. Furthermore, in a more broad perspective, the difference between standard and deviant stimuli can be seen as stimuli which are more probable or less probable, respectively. The input patterns can therefore be modeled as Markov chains, which was done by Mill, Coath, Wennekers, and Denham (2011). They found that the way the deviants, or less probable states, are represented in the network, highly depends on the parameters of the system, which was also found in this thesis.

To the best of the authors knowledge, there are no experimental findings which either clearly supports the exact findings, nor rejects it. However, there is a lot of evidence that less probable stimuli are processed in a special way. In future research, it would be an interesting approach to use the oddball paradigm to train different stimuli patterns to the brain using Markov chains. Results from such an experiment would probably reveal if the effect from the simulation also appears in natural networks.

Another biological relation can be found regarding the plastic training. In the results section, figure 15 shows that the network needs about $20,000$ steps of plastic training, until the performance becomes stable and further training does not influence the performance any more. Interestingly, there seems to be a parallel to the observation that the fraction of excitatory-to-excitatory connections stabilizes after some time. The fraction designates how many weights stay with a weight of $w_{ij} > 0$ during plasticity training. Many weights, which are not necessary for the input pattern, decay to zero or close to zero, due to STDP. This effect is known to occur in natural networks (Yasumatsu et al., 2008). In figure 26b a plot from Hartmann et al. (2015) is shown. It takes about $20,000$ steps of training until the stable fraction was reached. This corresponds to the training time of the network and possibly explains the ceiling effect of training steps. As long as non-necessary connections exist, the input pattern cannot be reproduced very well. A ceiling is reached, when the fraction stabilizes. At that point no further increase in performance should be possible.

As already mentioned in section 4.5, Zheng et al. (2013) extended a SORN with structural plasticity. After every step, with a low probability, new connections were established. With this mechanism it should be possible to extend the learning phase slightly. It could possibly also reduce the effects due to the IP mechanism. Furthermore, while initializing the connections, a stochastic component to the connectivity of every neuron could be added. The connectivity λ^W would stay the same as before, but it would become the mean connectivity per neuron instead of a fixed connectivity per neuron. A stochastic component with a specific distribution could add variation to every neuron.

8.4 Markov chains

Directly connected to the former suggestion, it would be interesting to apply multi-state Markov chains. Currently, only one state is active at a specific point in time. This holds for the plastic and no-plastic training as well as for the testing phase, where only one state is classified for each step. But in more complex rules, a stimulus can have more than one property. For example in the experiments from Paavilainen et al. (2007) and Bendixen et al. (2008), the auditive stimuli were different in their pitch *and* in their length. Therefore, it would be interesting to define an input cluster for low and high tones as well as an input cluster for short and long tones. At a specific point in time always two clusters are active: low/short, low/long, hight/short, hight/long. Using such a Markov chain would enable the network to possibly reproduce the experimental findings.

According to the IP-hypothesis, also the number of states n of the Markov chain should be important. In the present thesis, only Markov chains with $n = 4$ states were tested. But it can be predicted that if the number of states increase, the performance should decrease, due to intrinsic plasticity. If only one state is active at a time and many states are available, it takes possibly a long time until the current state will be activated again. Therefore, spontaneous activity will activate neurons in different clusters of the network and the classification will be at least more ambiguous. Markov chains with $n > 4$ should be tested in combination with $k > 1$ in the k-nearest neighbor classification algorithm (see below) in order to observe if the ambiguous behavior happens as predicted or not.

Another aspect is the prediction problem. The stationary distribution was only appropriate in some cases. A better predictor could possibly be derived from the transition matrix directly, since it contains much more information than the stationary distribution. A clustering algorithm for Markov chains, as it was developed by Van Dongen (2001), could possibly be an interesting candidate. If some states have a high probability to each other, but a low probability to the others, the chain ends up in lower performance, due to intrinsic plasticity. This structure could be reflected by a clustering algorithm.

8.5 Classification

In the current network simulation, the classification was done with an algorithm which is similar to a k-nearest neighbor algorithm with $k = 1$. The distance between an activity pattern in the testing phase and every activity patterns in the no-plastic training phase was calculated. The activity pattern from the no-plastic training phase with minimum distance was used for classification. An extension to this algorithm could be a k-nearest neighbor algorithm with $k > 1$, such that k closest activity patterns from the no-plastic training phase are taken for classification. The majority of appearing states would be the classified state. This has two advantages:

- It is easier to evaluate ambiguous classifications. In appendix D the classification method was evaluated. While the hamming threshold criterion was able to exclude ambiguous states, it is not able to classify which state is the second closest. With $k > 1$ it is very easy to classify specific input patterns as ambiguous and to get information about which states are involved in that case.
- It allows classification of two or more states. If a multi-state Markov chain would be used, it is necessary to classify which states are currently active. A classification with $k = 1$ only considers one activity pattern. Therefore, $k > 1$ is necessarily needed to classify multiple states. To be precise, k needs to be at least as high as the number of simultaneously active states.

8.6 IP mechanism

Regarding the IP mechanism, the distribution for the target rate range, introduced in equation (34), could be modified. For stability reasons the target rate H^{IP} is not fixed (Hartmann et al., 2015). A term $\varepsilon_i^{\text{IP}}$ was added, introduced in section 6.2, which is uniformly distributed with $\varepsilon_i^{\text{IP}} \sim \text{Uni}[a, b] = \text{Uni}[-\sigma^{\text{IP}}, +\sigma^{\text{IP}}] = \text{Uni}[-0.01, 0.01]$. σ^{IP} was already systematically changed, shown in figure 25. Since it was not possible to reproduce the previous results, perhaps another approach could be interesting. Possibly, if $\varepsilon_i^{\text{IP}}$ is sampled from another distribution, the results would change. A uniformly distributed $\varepsilon_i^{\text{IP}}$ has the advantage that it has no infinite tails. Therefore, there are no outliers or, in other words, the range of the outliers can easily be controlled. But on the other hand, the uniform distribution does not concentrate its probability mass around the expectation value, like a normal distribution would do. An interesting candidate would be a Beta distribution which can be centralized and scaled to fit to the requirements. It has the advantage, that is has finite tails and a concentration around the expectation value if the parameters are chosen in an appropriate way with $\alpha = \beta > 1$. Note that the uniform distribution is just a special case of the Beta distribution with $\alpha = \beta = 1$. An example for a

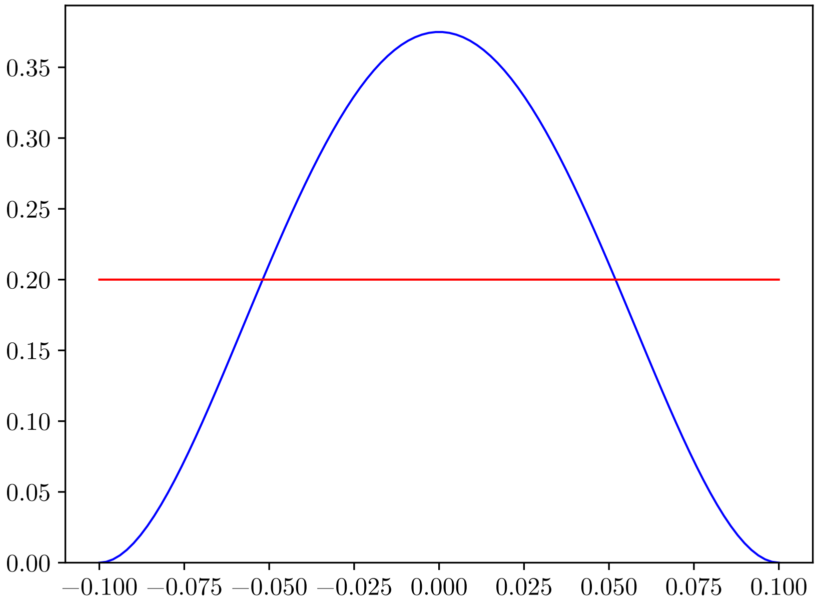


Figure 28: Beta distribution which can be used for $\varepsilon_i^{\text{IP}}$, instead of a uniform distribution. The blue line shows the probability density function of a Beta distribution, where $\alpha = \beta = 3$. The distribution is centralized and divided by a scaling factor of 5. The red line is the probability density function of a uniform distribution with $a = -0.01$ and $b = 0.01$, as it was already used in Hartmann, Lazar, Nessler, and Triesch (2015) and in this thesis. The same results from a Beta distribution with $\alpha = \beta = 1$, which is centralized and divided by 5 again.

suitable Beta distribution is shown in figure 28. It is compared to a uniform distribution.

Finally, the variation of the target rate $\bar{H}^{\mathrm{IP}}$ in section 7.2 lacks of the activity interaction. The higher $\bar{H}^{\mathrm{IP}}$, the higher is the activity in the network and vice versa. To keep the overall activity at a specific level, noise could be implemented. Additional activation can be sampled by a normal distribution with zero expectation value and some variance for every neuron. After every step, this additional activation is added to the actual activation of all neurons. The network would loose its deterministic character in favor of a stable average firing rate. The tricky task is to find a variance for that distribution, which would be able to provide the desired activity level.

8.7 Conclusion

SORN seems to be an attractive approach for neural simulations, since it allows to research on dynamic behavior of networks. In previous work, simulations showed that SORN performs very well and is able to reproduce biological findings (Lazar et al., 2009; Zheng et al., 2013; Aswolinskiy & Pipa, 2015; Hartmann et al., 2015). The input of those simulations was chosen in relation to specific biological experiments. In the present approach the input patterns were generalized using Markov chain Monte Carlo. The network was trained with samples from systematically chosen Markov chains. The performance was measured with a mean squared error between the initially chosen transition matrix from the Markov chain and the learned transition matrix, calculated by the occurring pattern of states in the network.

It was shown that the performance of the network depends on the chosen Markov chain. If the states of the Markov chain are very regular, which means that the probability between the states is relatively equal, the performance was very high, significantly higher than the performance of static reservoir networks. On the other hand, if the states are irregular and the probability of some states differ a lot from others, the performance is low. It converges to the performance of a static reservoir network with increasing irregularity.

In order to find an explanation for this behavior, parameters and properties of the network were tested, such as the size of the network, the training time and the classification mechanism. It was finally suggested that the intrinsic plasticity plays an important role for the observed performance differences. The so called IP-hypothesis suggests that those states which were not active for a longer time will spontaneously be active, due to the decreasing threshold of the IP rule. This hypothesis was able to explain the behavior of many different series of Markov chains. Furthermore, when the target rate, the main parameter of the intrinsic plasticity, was decreased, the effect was reduced. That clearly shows the influence of the IP mechanism on the observed differences between the models. Effects from other parameters could be excluded, like the target rate range, the IP learning rate and the connectivity of the network.

Further improvements of the network and future research was discussed. It was suggested to improve the intrinsic plasticity by choosing another distribution for the target rate range. To validate the influence of the target rate, noise could be included when varying the target rate parameter, in order to keep the activity level stable. Furthermore, the classification mechanism could be extended to a k-nearest neighbor algorithm with $k > 1$. Finally, in order to be able to reproduce more biological findings, a multi state Markov chain could be applied to SORN.

In sum, the Markov chain approach unveiled an interesting behavior of the network, showing that SORN does not outperform static reservoir networks in every case. The network behavior is not necessarily independent from its time-dependent input pattern. Further investigations showed that the intrinsic plasticity has a special role in the network regarding the observed effects. Finally it remains to be tested if indications for the present findings can be found by biological experiments.

Appendix

A Proof of null recurrent chain

In section 5.2 several properties of a Markov chain were introduced. An example of a *null recurrent* Markov chain was given in figure 10c. The null recurrent property of that chain will be proven in this appendix section. As a reminder the chain is shown again in figure 29. Also remind the definition of the random variable T_{x_i} in equation (24), which is the number of steps necessary to come back to state x_i for the *first time*. A state is recurrent if $\mathbb{P}(T_{x_i} < \infty) = 1$, which corresponds to equation (25) and null recurrent if $\mathbb{E}[T_i] = \infty$, which was defined in (26) and the following.

First, a homogen Markov chain is assumed and an infinite state space $S = \{x_1, x_2, x_3, x_4, ...\} = \{A, B, C, D, ...\}$ is given. $\mathbb{P}(T_{x_i} = t)$ will be denoted by $q^t_{x_i}$. It is the probability to come back to state x_i for the first time after t steps. Therefore, starting from state A, the probability to come back to state A in only one step is

$$q^1_A = \mathbb{P}(T_A = 1) = \mathbb{P}(X_1 = A|X_0 = A) = p_{AA} = 0. \tag{47}$$

Since A has no self-loop, it is impossible to come back to A in just one step. For two steps, a non-zero probability exists:

$$\begin{aligned} q^2_A &= \mathbb{P}(T_A = 2) \\ &= \mathbb{P}(X_1 = B|X_0 = A) \cdot \mathbb{P}(X_2 = A|X_1 = B) \\ &= p_{AB} \cdot p_{BA} = 1 \cdot \frac{1}{2} = \frac{1}{2} \end{aligned} \tag{48}$$

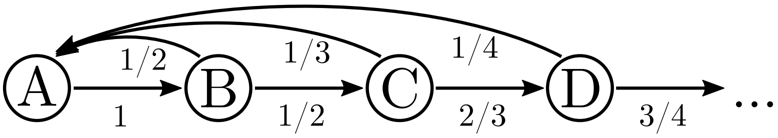


Figure 29: An example for a null recurrent Markov chain. It was introduced in figure 10c as an example of a null recurrent Markov chain.

Results are similar for $t = 3$, $t = 4$ and $t = 5$ steps:

$$q_A^3 = p_{AB} \cdot p_{BC} \cdot p_{CA} = 1 \cdot \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6} \tag{49a}$$

$$q_A^4 = p_{AB} \cdot p_{BC} \cdot p_{CD} \cdot p_{DA} = 1 \cdot \frac{1}{\cancel{2}} \cdot \frac{\cancel{2}}{3} \cdot \frac{1}{4} = \frac{1}{3} \cdot \frac{1}{4} = \frac{1}{12} \tag{49b}$$

$$q_A^5 = p_{AB} \cdot p_{BC} \cdot p_{CD} \cdot p_{DE} \cdot p_{EA} = 1 \cdot \frac{1}{\cancel{2}} \cdot \frac{\cancel{2}}{\cancel{3}} \cdot \frac{\cancel{3}}{4} \cdot \frac{1}{5} = \frac{1}{4} \cdot \frac{1}{5} = \frac{1}{20} \tag{49c}$$

It can easily be seen that q_A^t follows a specific rule, namely

$$q_A^t = \frac{1}{t-1}\frac{1}{t}, \tag{50}$$

for all $t > 1$. Note that it does not hold for $t = 1$.

Using this rule, the probability to come back to state A can be obtained by

$$\begin{aligned} \mathbb{P}(T_A < \infty) &= \sum_{t=1}^{\infty} \mathbb{P}(T_A = t) = \sum_{t=1}^{\infty} q_A^t \\ &= 0 + \sum_{t=2}^{\infty} \frac{1}{t-1}\frac{1}{t} = \sum_{t=1}^{\infty} \frac{1}{t}\frac{1}{t+1}. \end{aligned} \tag{51}$$

With induction it can be shown that

$$\mathbb{P}(T_A < \infty) = \sum_{t=1}^{\infty} \frac{1}{t}\frac{1}{t+1} = \lim_{n\to\infty} \frac{n-1}{n} = 1. \tag{52}$$

Hence, equation (25) is fulfilled and state A is recurrent. It remains to show that the state is also null recurrent. In order to verify this, the expectation value can be calculated using

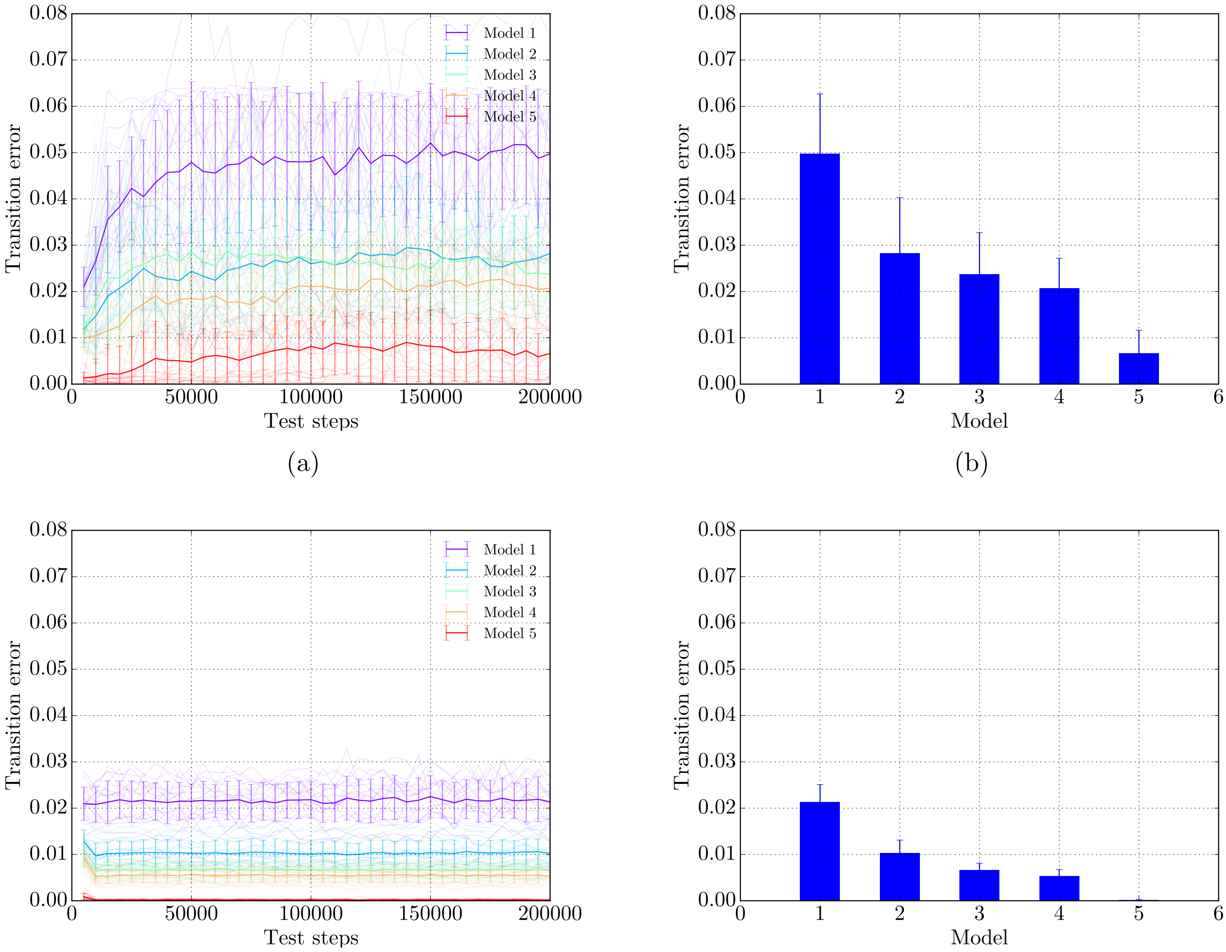
$$\begin{aligned}\mathbb{E}[T_A] &= \sum_{t=1}^{\infty} t \cdot \mathbb{P}(T_A = t) \\ &= 0 + \sum_{t=2}^{\infty} t \cdot \frac{1}{t-1}\frac{1}{t} = \sum_{t=2}^{\infty} \frac{1}{t-1} = \sum_{t=1}^{\infty} \frac{1}{t} = \infty,\end{aligned} \tag{53}$$

where the last step follows from the harmonic series. It was possible to show that the expected number of steps, necessary to come back to state A for the first time, are infinite. Equation (26) is fulfilled and state A is null recurrent.

□

B Testing phase with STDP

The spike-timing-dependent plasticity (STDP) is responsible for the dynamics in the weights. It was applied while training to adjust the weights with respect to the time-dependent input patterns. Afterwards, while no-plastic training and while testing, STDP was switched off in order to sustain the learned structure. In results, e.g. in figure 14c, it was shown that the static structure in the testing phase results in a constant performance. If STDP is active in testing phase and, therefore, weights are further adapted, spontaneous activity should lead to random changes in the weights, which will increasingly occur independent from the initial Markov chain transitions. Subsequently, the stored state transitions should decay. The testing



(a) (b)

(c) (d)

Figure 30: Performance with and without STDP in testing phase. **a)** shows that the error increases quite fast in the testing phase with STDP, compared to **c)**, where STDP was switched off. **b)** and **d)** shows the performance at the end of the testing phase for both cases. As the error and the standard error is higher in the case with STDP, the differences between the models are mainly maintained. Note that the testing phase was extended to $T_{\text{test}} = 200,000$ steps and that the y-axis of the plots is fixed to 0.08 for comparison. Error bars indicate standard errors.

phase was extended to $T_{\text{test}} = 200,000$ steps to be able to observe the process for a longer time and the model series II from figure 18 was used with 5 models.

Figure 30a shows that the network ‘forgets’ the stored information and the effect seems to be stronger for those Markov models, where errors were initially higher. The results can be compared to figure 30c, where also $T_{\text{test}} = 200,000$ were simulated, but with STDP switched off, as it was done in all previous simulations. The error increases quickly in the beginning and seems to maintain at a specific level. Interestingly, the differences between the models remain over the whole $200,000$ steps. Finally, figures 30b and 30d show the performance for both cases at the end of the testing phase. Those plots support the prior analysis. As the error and the standard error is higher in the simulation with STDP, compared to the simulation without it, the differences between the models are basically maintained.

C No-plastic training

The length of the no-plastic training phase, which was introduced in section 6.3, should have no influence on the performance. Its objective is to classify the current state of the network, when only spontaneous activity is in the network. Even if the input is still active in the no-plastic training phase, the main training effect should come from the plastic training phase, when the network adjusts the weights using STDP and input is given for a long time of $T_{\text{plastic}} = 50,000$ steps in most cases.

The number of steps in the no-plastic training phase $T_{\text{noplastic}}$ was varied for model series II from figure 18 with 5 models. As can be seen in figure 31, there is no influence on the performance of the models. The performance is independent from the number of no-plastic training steps. There are some slight differences in model 1, but they are not significant ($p > 0.05$). Note that it is not possible to set the number of training steps below the number of steps to compare with. That actually means that necessarily $T_{\text{noplastic}} > \tilde{T}_{\text{compare}}$, as already discussed in the methods section.

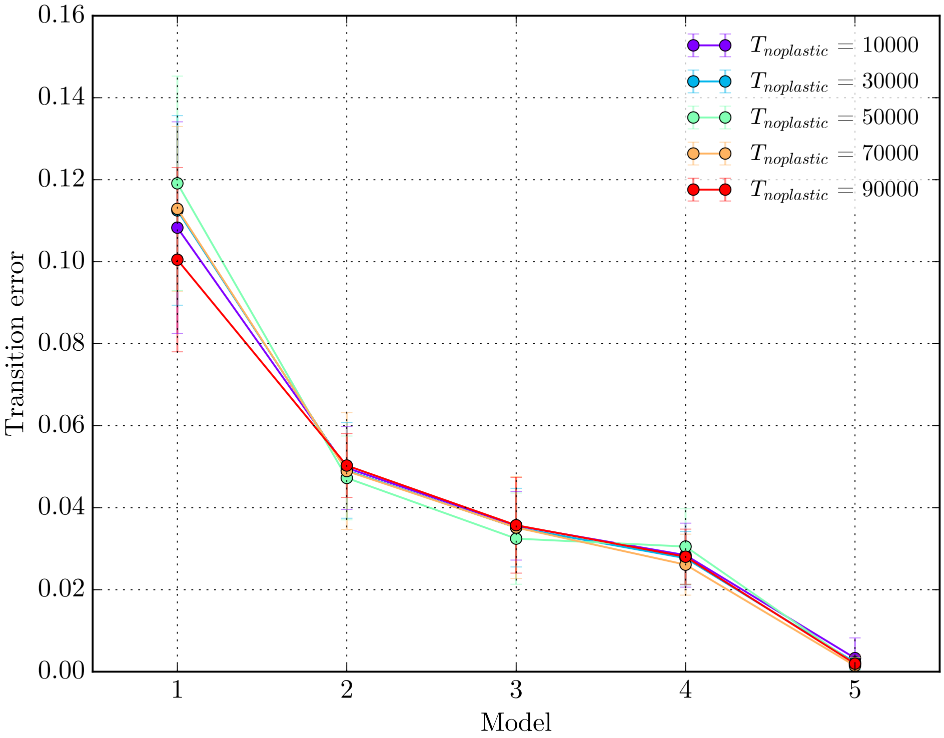


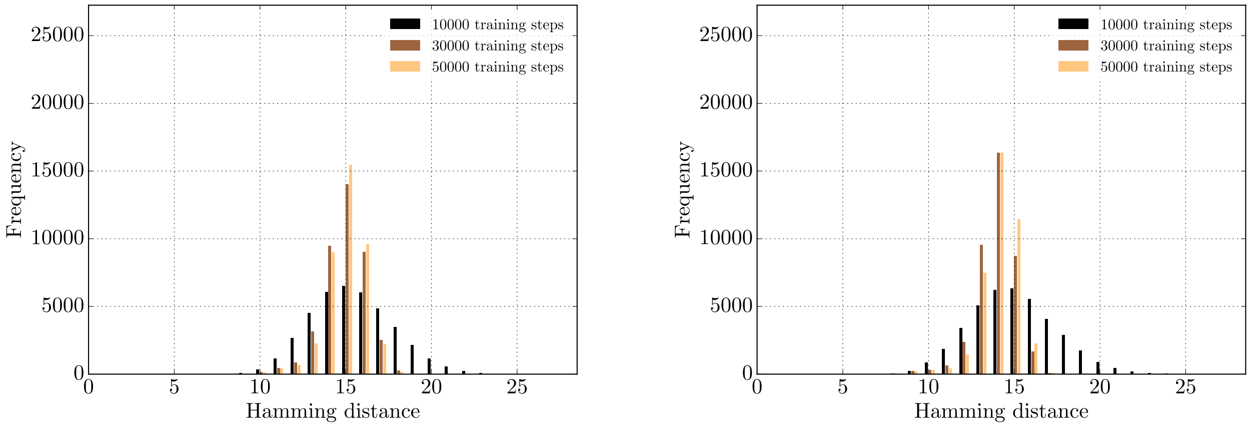
Figure 31: Influence of the length of the no-plastic training phase $T_{\text{noplastic}}$. The parameter was chosen between $T_{\text{noplastic}} = 10,000$ and $T_{\text{noplastic}} = 90,000$. The length of the no-plastic training phase does not influence the performance of the models.

D Hamming classification

The purpose of this section is to evaluate the classification mechanism, based on the hamming distance. The systematic model series II was used for evaluation,, shown in figure 18 in results. The series was realized with 5 models. The hamming distance calculation was introduced in section 6.4. In a first step, histograms were plotted, showing the frequencies of the smallest hamming distances for every step in the testing phase, where any testing phase step was compared with all comparison steps from the no-plastic testing phase. Results are shown in figure 32. There are three notable effects:

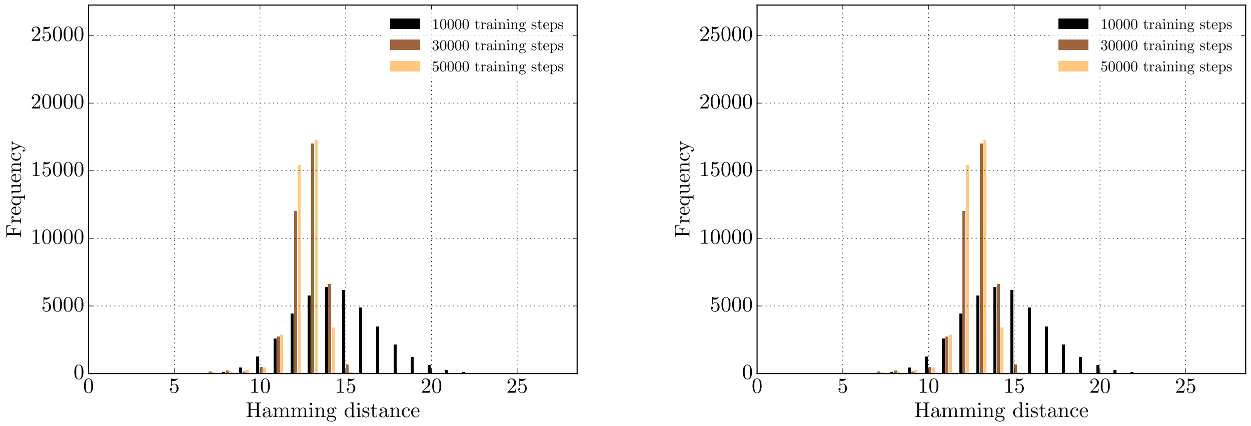
- A Kolmogorov-Smirnov test was performed, showing that all minimum hamming distances are normally distributed ($p > 0.05$).
- The more the model contains less probable states (model 1 has fewest transition probability to state A, model 5 is equal for every state), the higher is the mean of the minimum hamming distances in tendency.
- The more the network was trained, the lower is the variance and the mean of the minimum hamming distances in tendency. This effect is stronger for models with states which occur with similar probabilities.

Furthermore, it was checked if the classification method has an influence on the performance. In section 6.4 a parallel to the k-nearest neighbor algorithm with $k = 1$ was drawn. The classification mechanism does only search for the one activity pattern which is closest. The hamming distance could be the same for activity patterns that belong to different Markov states. In that case, the algorithm just chooses the first one. It would be possible to apply a threshold which classifies states as ‘unknown’ if the minimum hamming distance is too high. That is the case, if the state in the network is not very clear. It was checked if such a stricter rule, using a threshold, influences the behavior. Figure 33a shows that this is not the case. The transparent lines show the performance of different thresholds. The opaque line represents a threshold of infinity, which equals no threshold at all, and therefore represents the algorithm which was used by default. If a threshold is applied, its performance seems to be very similar to the performance of the infinite threshold. Hence, the original algorithm seems to be very robust.



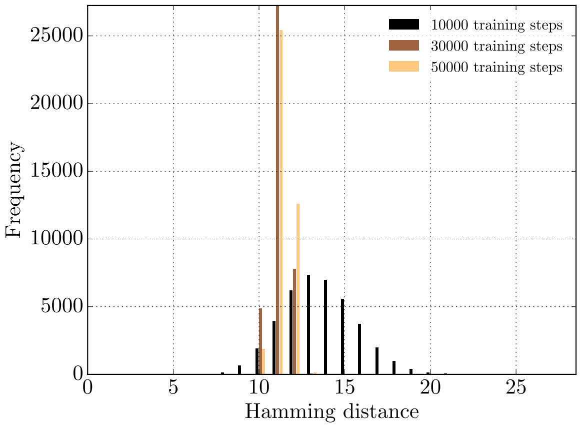
(a) Model 1

(b) Model 2



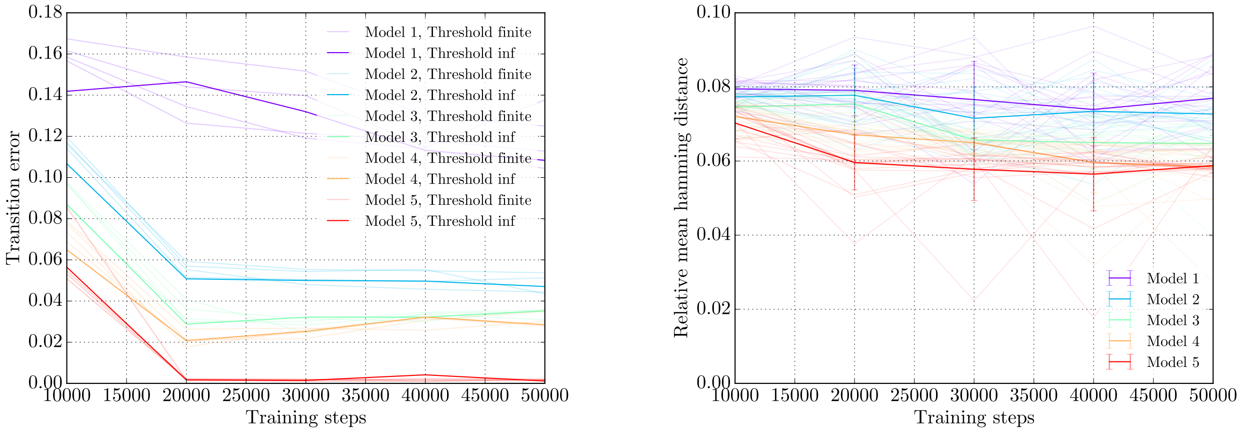
(c) Model 3

(d) Model 4



(e) Model 5

Figure 32: Histograms of minimum hamming distances for 5 models. The minimum hamming distances are normally distributed. The variances of the minimum hamming distances decrease if training is increased. The mean also decreases, as long as state A is not to unlikely. Furthermore, the mean decreases if state A is more and more equally connected to the other states and the states in the model becomes more regular.



(a)

(b)

Figure 33: Performance results associated with hamming distances. In **a)** thresholds of $20, 22, 24$ and 26 were applied. If the minimum hamming distance exceeded that threshold, the current pattern was classified as ‘unknown’. In the legend they are represented with the transparent lines for each model. They are compared to the default case, were no threshold was applied. In that case the threshold is infinity. It is represented by the opaque line for each model. In figure **b)**, the mean minimum hamming distance was calculated and divided by the maximum possible hamming distance N^E. The relative mean hamming distance can be used as a performance measure. Differences between the models are present and a training effect occurs. But since this measure is very robust and, therefore, effects are very small compared to the transition error, this performance measure was not used.

Finally, the relative mean hamming distance was calculated. The minimum hamming distances were averaged and divided by the maximum possible hamming distance N^E in order to provide another performance measure. The better the model is learned, the lower should be the minimum hamming distance. Indeed, figure 33b shows that the relative mean hamming distance behaves similar to the other performance measures. But the differences between the models and the decrease in performance, due to training, seems to be very low. Therefore, the relative mean hamming distance is not an appropriate performance measure. On the other hand, it shows again that the hamming distance is very robust between the models and regarding more or less training.

E Models close to equal probabilities

In most cases, the ‘equal’ model, where the probability of all available connections is 0.5, was outstanding in performance. There was a drop between this model and the very next model, e.g. in figure 19a, 23a or 23d in the results section. In this section, starting from the ‘equal’ model, the probabilities were varied in very small steps to be able to observe the behavior for very small changes. For evaluation, the model series II from figure 18 was adapted to a series which is shown in figure 34. This series was applied with 76 steps between the left and right model. Therefore, the probabilities were changed in very small steps.

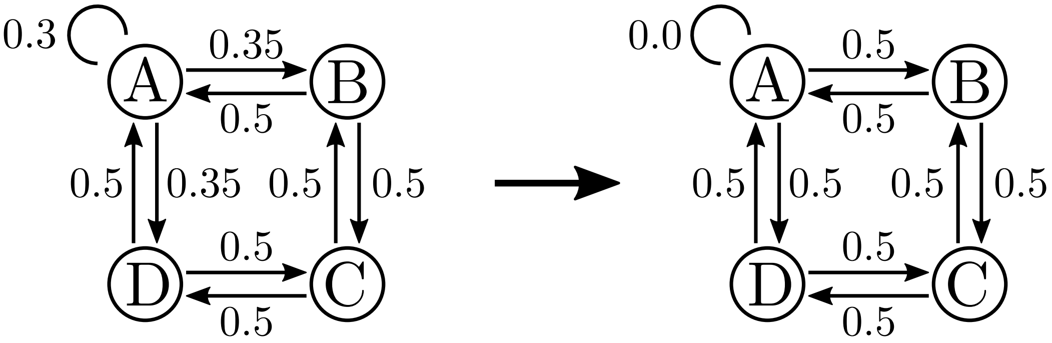


Figure 34: Model series close to equal probabilities. This series is adapted from the model series II in figure 18, where the starting probability is already chosen more close to the ‘equal’ model. This series was divided into 76 steps with very small probability changes.

The results are shown in figure 35. Starting from the right, were all probabilities are equal, the error starts to increase very fast. Small changes seem to influence the performance more strongly if the models are close to the ‘equal’ model. The performance change of the models on the left side is small. Therefore, those models seem to be more robust to changes of the probabilities. Whereas the models on the right are very sensitive to changes of the probabilities.

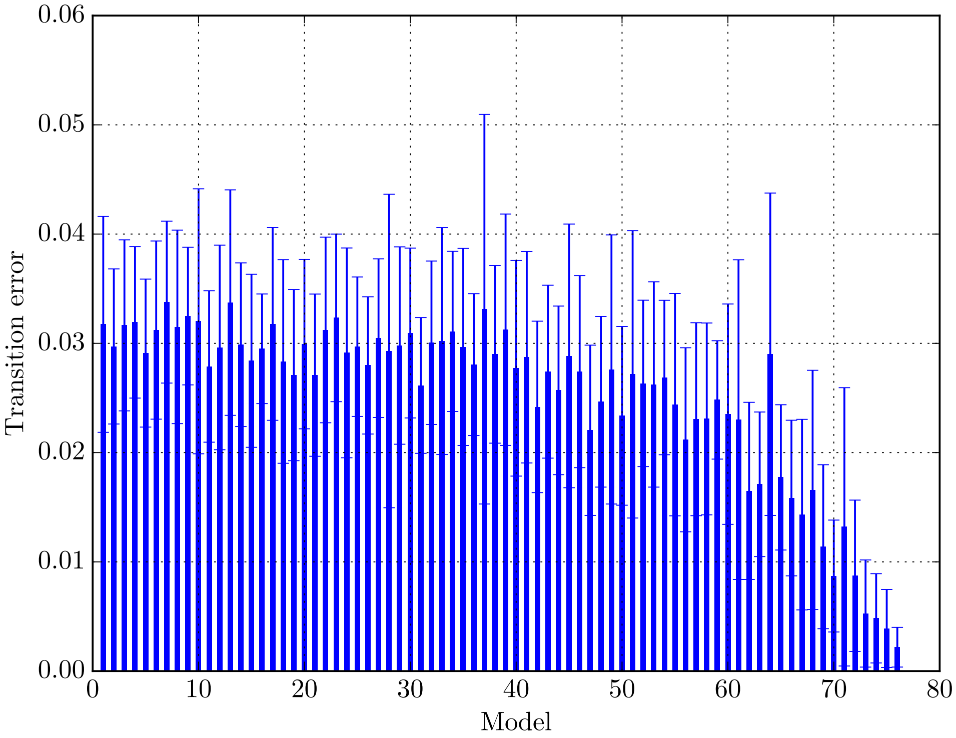
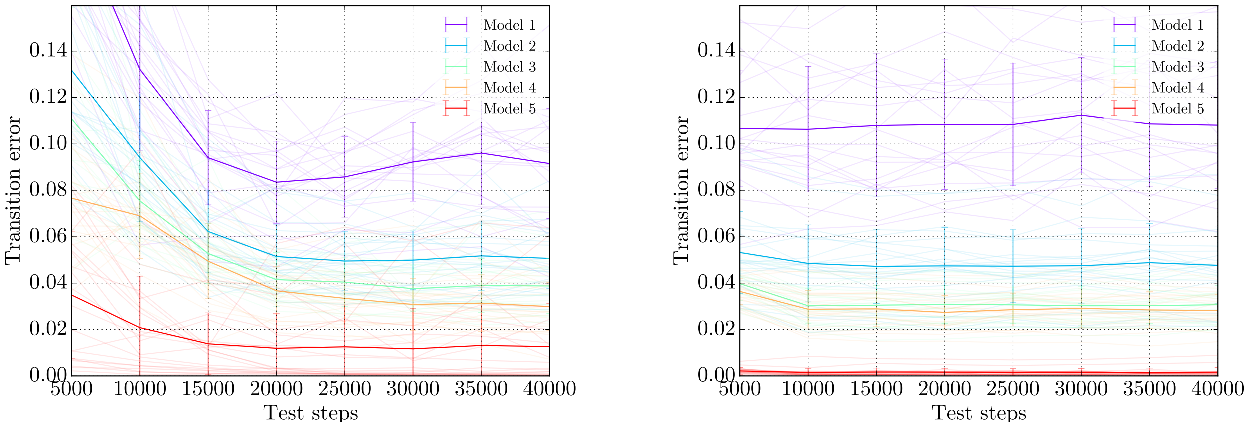


Figure 35: Performance of model series close to equal probabilities. 76 models were applied and the probabilities were changed in very small steps. The network is sensitive to changes around the ‘equal’ model (last model on the right) and becomes more robust towards the more unequal models (right to left).

F IP learning rate

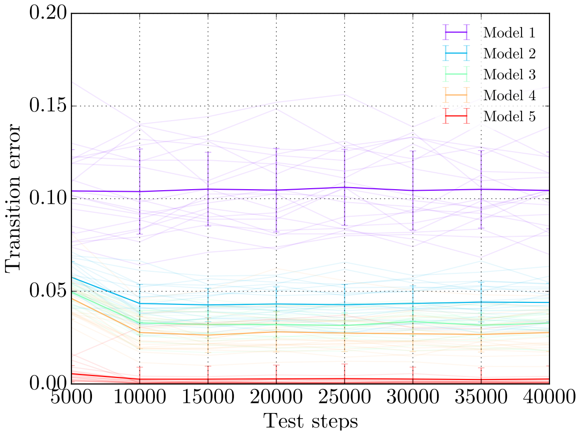
The IP mechanism was suggested to explain the performance differences of the different input patterns, sampled from the Markov chains. The intrinsic plasticity rule is build with three parameters, introduced in equation (34) in methods. $\bar{H}^{\mathrm{IP}}$ and σ^{IP} were already tested in section 7.2 in results, but the impact of the learning rate parameter η_{IP} remains to be checked. The model series II from figure 18 was used and applied with 5 models.

Figure 36a shows that a very small $\eta_{\mathrm{IP}} = 0.0002$ increases the burn-in time in the testing phase. The network needs a long time until it finds a stable performance. If the parameter is very high, like in figure 36b, where $\eta_{\mathrm{IP}} = 0.002$, the performance starts to be stable after the first few steps in testing phase. This also happens for the default case with $\eta_{\mathrm{IP}} = 0.001$, shown in figure 36c.



(a) $\eta_{\mathrm{IP}} = 0.0002$

(b) $\eta_{\mathrm{IP}} = 0.002$



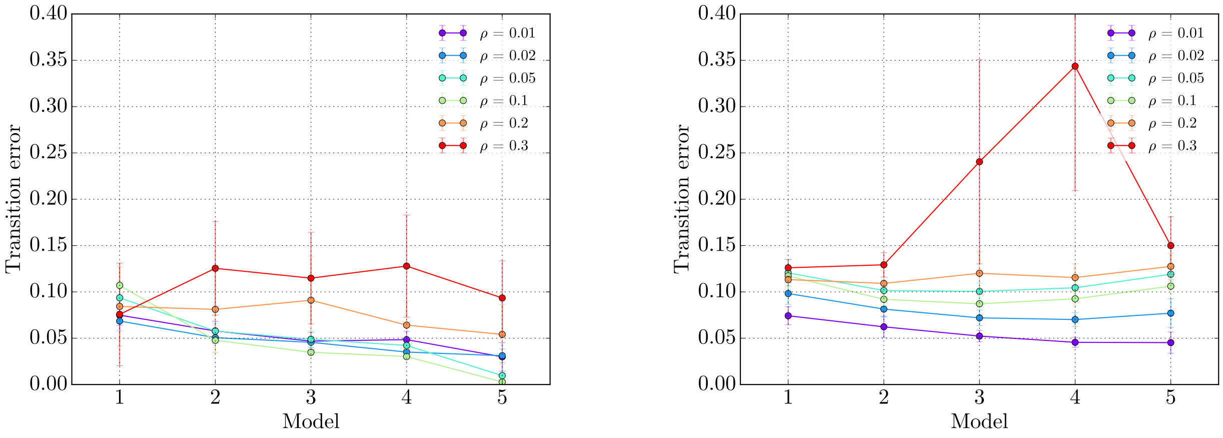
(c) $\eta_{\mathrm{IP}} = 0.001$ (default)

Figure 36: Influence of the IP learning rate η_{IP} on the burn-in time in the beginning of the testing phase.

G Network connectivity

As already mentioned at the end of results section 7.2, probably the connectivity could influence the performance, beside the intrinsic plasticity. The connectivity is defined by the number of connections of every neuron, which equals the number of weights with $w_{ij} > 0$ or — under a biological perspective — the number of synapses. Initially, every neuron is connected with $\rho = 0.1 \hat{=} 10\%$ of the other neurons. Since $N^E = 200$, every neuron has $\lambda^W = \rho \cdot N^E = 20$ synapses or weights greater zero.

Lazar et al. (2009) showed in their appendix that the connectivity influences the performance of the network. In particular, they concluded that if the network is too dense, the activity 'explodes'. To replicate the results in a context of Markov chain sampled input, model series II from figure 18 was divided into 5 models. The results for different values of ρ are shown in figure 37. On the left side the results for a SORN with $T_{\text{plastic}} = 50,000$ steps show that the performance is similar for $\rho \leq 0.1$ and increases for $\rho > 0.1$. On the left side, a static reservoir is shown with $T_{\text{plastic}} = 0$ steps. In that case, performance is increasing if ρ increases. For $T_{\text{plastic}} = 50,000$ steps and $\rho = 0.3$ the static reservoir network has very low performance in representing model 3 and model 4. Lazar et al. (2009) suggested that this behavior is due to seizure-like bursts in the network, due to the high connectivity. Comparing both plots, a sparse connectivity increases the performance for both networks, plastic and static. Furthermore, the plastic network outperforms the static reservoir network. Both findings are in line with previous



(a) Plastic reservoir ($T_{\text{plastic}} = 50,000$)

(b) Static reservoir ($T_{\text{plastic}} = 0$)

Figure 37: Variation of connectivity ρ. The left plot shows results for a plastic reservoir and the right plot for a static reservoir. For both holds that the more sparse the network is connected, the better the performance. Furthermore, the plastic reservoir outperforms the static reservoir network.

findings from Lazar et al. (2009).

References

Abbott, L. F. & Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature neuroscience*, *3*, 1178–1183.

Aswolinskiy, W. & Pipa, G. (2015). Rm-sorn: a reward-modulated self-organizing recurrent neural network. *Frontiers in computational neuroscience*, *9*, 36.

Awatramani, G. B., Price, G. D., & Trussell, L. O. (2005). Modulation of transmitter release by presynaptic resting potential and background calcium levels. *Neuron*, *48*(1), 109–121.

Baddeley, H. (2008). *Physics and the human body: stories of who discovered what.* Physics and the Human Body.

Bendixen, A., Prinz, W., Horváth, J., Trujillo-Barreto, N. J., & Schröger, E. (2008). Rapid extraction of auditory feature contingencies. *Neuroimage*, *41*(3), 1111–1119.

Bertschinger, N. & Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural computation*, *16*(7), 1413–1436.

Bladt, M. & Nielsen, B. F. (2017). Matrix-exponential distributions. In *Matrix-exponential distributions in applied probability* (pp. 199–296). Springer.

Brémaud, P. (2013). *Markov chains: gibbs fields, monte carlo simulation, and queues.* Springer Science & Business Media.

Chamberland, S. & Topolnik, L. (2012). Inhibitory control of hippocampal inhibitory neurons. *Frontiers in neuroscience*, *6*.

Clements, J. (1996). Transmitter timecourse in the synaptic cleft: its role in central synaptic function. *Trends in neurosciences*, *19*(5), 163–171.

Cudmore, R. H. & Desai, N. S. (2008). Intrinsic plasticity. *Scholarpedia*, *3*(2), 1363. revision #89024. doi:10.4249/scholarpedia.1363

Dale, H. (1935). Pharmacology and nerve-endings. SAGE Publications.

Dan, Y. & Poo, M.-M. (2006). Spike timing-dependent plasticity: from synapse to perception. *Physiological reviews*, *86*(3), 1033–1048.

Dasgupta, S., Tetzlaff, C., Kulvicius, T., & Wörgötter, F. (2015). Self-organization of computation in neural systems by interaction between homeostatic and synaptic plasticity. *BMC Neuroscience*, *16*(1), O5.

Deetjen, P., Speckmann, E.-J., & Hescheler, J. (2005). *Repetitorium physiologie.* Urban & Fischer.

Eccles, J. (1976). From electrical to chemical transmission in the central nervous system. *Notes and records of the Royal Society of London*, *30*(2), 219–230.

Finger, S. (2001). *Origins of neuroscience: a history of explorations into brain function*. Oxford University Press, USA.

Fu, Y.-X., Djupsund, K., Gao, H., Hayden, B., Shen, K., & Dan, Y. (2002). Temporal specificity in the cortical plasticity of visual space representation. *Science*, *296*(5575), 1999–2003.

Gavornik, J. P. & Bear, M. F. (2014). Learned spatiotemporal sequence recognition and prediction in primary visual cortex. *Nature neuroscience*, *17*(5), 732–737.

Geman, S. & Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 721–741.

Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep learning*.

Goris, R. L., Movshon, J. A., & Simoncelli, E. P. (2014). Partitioning neuronal variability. *Nature neuroscience*, *17*(6), 858–865.

Hartmann, C., Lazar, A., Nessler, B., & Triesch, J. (2015). Where's the noise? key features of spontaneous activity and neural variability arise through learning in a deterministic network. *PLoS computational biology*, *11*(12), e1004640.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, *57*(1), 97–109.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

Jaeger, H. & Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *science*, *304*(5667), 78–80.

Kempter, R., Gerstner, W., & Van Hemmen, J. L. (1999). Hebbian learning and spiking neurons. *Physical Review E*, *59*(4), 4498.

Kenet, T., Bibitchkov, D., Tsodyks, M., Grinvald, A., & Arieli, A. (2003). Spontaneously emerging cortical representations of visual attributes. *Nature*, *425*(6961), 954–956.

Kirschstein, T. & Köhling, R. (2009). What is the source of the eeg? *Clinical EEG and neuroscience*, *40*(3), 146–149.

Lazar, A. (2009). *Self-organizing recurrent neural networks*. (Doctoral dissertation, Citeseer).

Lazar, A., Pipa, G., & Triesch, J. (2009). Sorn: a self-organizing recurrent neural network. *Frontiers in computational neuroscience*, *3*, 23.

Lukoševičius, M. & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149.

Maass, W. & Markram, H. (2004). On the computational power of circuits of spiking neurons. *Journal of computer and system sciences*, *69*(4), 593–616.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural computation*, *14*(11), 2531–2560.

Mill, R., Coath, M., Wennekers, T., & Denham, S. L. (2011). A neurocomputational model of stimulus-specific adaptation to oddball and markov sequences. *PLoS computational biology*, *7*(8), e1002117.

Miller, K. D. & MacKay, D. J. (1994). The role of constraints in hebbian learning. *Neural Computation*, *6*(1), 100–126.

Moore, G. E. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE*, *86*(1), 82–85.

Näätänen, R., Gaillard, A. W., & Mäntysalo, S. (1978). Early selective-attention effect on evoked potential reinterpreted. *Acta psychologica*, *42*(4), 313–329.

Norton, D. & Ventura, D. (2006). Preparing more effective liquid state machines using hebbian learning. In *Neural networks, 2006. ijcnn'06. international joint conference on* (pp. 4243–4248). IEEE.

Osogami, T. (2017). Boltzmann machines and energy-based models. *arXiv preprint arXiv:1708.06008*.

Paavilainen, P., Arajärvi, P., & Takegata, R. (2007). Preattentive detection of nonsalient contingencies between auditory features. *Neuroreport*, *18*(2), 159–163.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning* (pp. 1310–1318).

Peracchia, C. (1977). Gap junction structure and function. *Trends in Biochemical Sciences*, *2*(2), 26–31.

Pillai, S. U., Suel, T., & Cha, S. (2005). The perron-frobenius theorem: some of its applications. *IEEE Signal Processing Magazine*, *22*(2), 62–75.

Pozo, K. & Goda, Y. (2010). Unraveling mechanisms of homeostatic synaptic plasticity. *Neuron*, *66*(3), 337–351.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Savtchenko, L. P. & Rusakov, D. A. (2007). The optimal height of the synaptic cleft. *Proceedings of the National Academy of Sciences*, *104*(6), 1823–1828.

Schandry, R. (2007). *Biologische psychologie.*

Schrauwen, B., Verstraeten, D., & Van Campenhout, J. (2007). An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007* (pp. 471–482).

Schröger, E. (1997). On the detection of auditory deviations: a pre-attentive activation model. *Psychophysiology*, *34*(3), 245–257.

Schwartz, E. L. (1993). *Computational neuroscience.* Mit Press.

Seneta, E. (2006). *Non-negative matrices and markov chains.* Springer Science & Business Media.

Sjöström, J. & Gerstner, W. (2010). Spike-timing dependent plasticity. *Scholarpedia*, *5*(2), 1362. revision #151671. doi:10.4249/scholarpedia.1362

Skrebitskii, V. & Chepkova, A. (1999). Synaptic plasticity in learning and memory. *Uspekhi fiziologicheskikh nauk*, *30*(4), 3–13.

Sossin, W. S., Sweet-Cordero, A., & Scheller, R. H. (1990). Dale's hypothesis revisited: different neuropeptides derived from a common prohormone are targeted to different processes. *Proceedings of the National Academy of Sciences*, *87*(12), 4845–4848.

Tetzlaff, C., Dasgupta, S., Kulvicius, T., & Wörgötter, F. (2015). The use of hebbian cell assemblies for nonlinear computation. *Scientific reports*, *5*.

Thomopoulos, N. T. (2012). *Essentials of monte carlo simulation: statistical methods for building simulation models.* Springer Science & Business Media.

Tiitinen, H., May, P., Reinikainen, K., & Näätänen, R. (1994). Attentive novelty detection in humans is governed by pre-attentive sensory memory. *Nature*, *372*(6501), 90.

Turrigiano, G. (2012). Homeostatic synaptic plasticity: local and global mechanisms for stabilizing neuronal function. *Cold Spring Harbor perspectives in biology*, *4*(1), a005736.

Turrigiano, G. G. (2008). The self-tuning neuron: synaptic scaling of excitatory synapses. *Cell*, *135*(3), 422–435.

Turrigiano, G. G. & Nelson, S. B. (2004). Homeostatic plasticity in the developing nervous system. *Nature Reviews Neuroscience*, *5*(2), 97–107.

Van Dongen, S. M. (2001). *Graph clustering by flow simulation* (Doctoral dissertation).

Waldeyer, W. (1891). Ueber einige neuere forschungen im gebiete der anatomie des centralnervensystems1. *DMW-Deutsche Medizinische Wochenschrift*, *17*, 1213–1218, 1244–1246, 1267–1269, 1287–1289, 1331–1332, 1352–1356.

Williams, A. H., O'Leary, T., & Marder, E. (2013). Homeostatic Regulation of Neuronal Excitability. *Scholarpedia*, *8*(1), 1656. revision #140978. doi:10.4249/scholarpedia.1656

Yao, H. & Dan, Y. (2001). Stimulus timing-dependent plasticity in cortical processing of orientation. *Neuron*, *32*(2), 315–323.

Yao, H., Shi, L., Han, F., Gao, H., & Dan, Y. (2007). Rapid learning in cortical coding of visual scenes. *Nature neuroscience*, *10*(6), 772–778.

Yasumatsu, N., Matsuzaki, M., Miyazaki, T., Noguchi, J., & Kasai, H. (2008). Principles of long-term dynamics of dendritic spines. *Journal of Neuroscience*, *28*(50), 13592–13608.

Zhang, L. I., Tao, H. W., Holt, C. E., Harris, W. A., & Poo, M.-m. (1998). A critical window for cooperation and competition among developing retinotectal synapses. *Nature*, *395*(6697), 37–44.

Zheng, P., Dimitrakakis, C., & Triesch, J. (2013). Network self-organization explains the statistics and dynamics of synaptic connection strengths in cortex. *PLoS computational biology*, *9*(1), e1002848.