



Christoph Rieger, Bsc.

Hierarchical architectures for spiking Winner-Take-All networks

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Biomedical Engineering

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Robert Legenstein

Institute of Theoretical Computer Science

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Robert Legenstein

Graz, April 2023

Contents

1	Introduction	1
2	Biological background	3
2.1	Winner-Take-All networks	3
2.2	The hierarchical structure of the brain	4
2.3	Visual cortex	4
2.4	Probabilistic brain	6
2.5	Feedback in the visual cortex	6
2.6	Plasticity	7
2.7	Spiking neural networks	9
3	Theoretical background	11
3.1	Bayes' theorem	11
3.2	Bayesian inference	11
3.3	Network model	12
3.4	Mathematical link between the spiking Winner-Take-All network model and Bayesian inference	17
4	Experiments	19
4.1	Experiment 5: Mathematical analysis and simulation of a 1D network	19
4.1.1	Introduction	19
4.1.2	Methods	20
4.1.3	Results	23
4.1.4	Discussion	35
4.2	Experiment 6: Simulation of the 1D network with double size	38
4.2.1	Introduction	38
4.2.2	Methods	39
4.2.3	Results	39

Contents

4.2.4	Discussion	42
4.3	Experiment 7: Training of the 1D network with predetermined parameters	42
4.3.1	Introduction	42
4.3.2	Methods	43
4.3.3	Results	43
4.3.4	Discussion	43
4.4	Experiment 2: Horizontal and vertical bars	49
4.4.1	Introduction	49
4.4.2	Methods	49
4.4.3	Results	52
4.5	Experiment 4: Horizontal and vertical bars with adaptive inhibition	64
4.5.1	Introduction	64
4.5.2	Methods	64
4.5.3	Results	64
4.6	Experiment 1: Rotated Bars	67
4.6.1	Introduction	67
4.6.2	Methods	67
4.6.3	Results	78
4.6.4	Conclusion	89
4.7	Experiment 3: Rotated bars with adaptive inhibition	89
4.7.1	Introduction	89
4.7.2	Methods	91
4.7.3	Results	91
5	Discussion	95
	Bibliography	97

1 Introduction

Functional networks of the human brain have a hierarchical modular organization. This means the execution of one specific task, for example vision, involves several modules which are interconnected (Meunier et al., 2009). Traditional views of the visual pathway of the brain hypothesized that the sensory input is passed on from lower to higher visual areas in a feed-forward system. However, Lee TS (2003) shows neurophysiological experimental evidence that there is feedback from high-level to low-level areas of the visual cortex. That feedback information is thought to be "explaining away" information, or putting emphasis on specific information of the low-level area. Through this mechanism can attention be realized and different cortex areas are able to adopt the same interpretation of some input.

The brain needs to handle a high degree of uncertainty in sensory input. When it first receives input it does not know what the information could represent, thus not knowing which parts of the input are most relevant. Several experiments on animals show that the computation of sensory input that the cerebral cortex performs can be explained and modelled by Bayesian inference (Funamizu Akihiro, 2016; Lee TS, 2003; Parr and Friston, 2018). They hypothesize that the brain functions as a generative and probabilistic model to reach its conclusions. This means the brain expresses information via probability distributions, rather than utilizing static neural codes. Bayes theorem yields a posterior probability, by multiplying a likelihood with a prior probability. In the context of cortical computation one can postulate that the posterior is represented by the output of a neural network. The likelihood can be computed based on the sensory input to the network and the feedback it receives from other cortical areas. The feedback from high-level to low-level cortical areas can be modelled as the prior probability (Nessler et al., 2013).

1 Introduction

There are various models for the computational dynamics of biological neurons (Gerstner and Kistler, 2002).

TODO 1. discuss spiking neurons 2. discuss briefly synaptic plasticity 3. explain synaptic plasticity in biological background

The plasticity of neural networks is often modelled as Spike-timing-dependent plasticity (STDP), as it agrees with biological experiments (Feldman, 2012; Dan Yang, 2004).

TODO!!!!!! 1. Talk about the canonical cortical microcircuit (softWTA paper douglas) 2. Say that WTA was postulated as a key computational motif To select the output of a neural network the soft Winner-Take-All mechanism is well established. Neurons with this mechanism choose what information is forwarded vertically into another layer and inhibit their horizontal neighbours of the same layer. This mechanism resembles the way biological pyramidal neurons function (Rodney J Douglas, 2004). Furthermore Winner-Take-All modules have a higher computational power than threshold-gates (Maass, 2000).

Nessler et al. (2013) and Guo et al. (2019) created Winner-Take-All spiking neural networks which perform Bayesian inference. Both of those works did not include feedback that comes from a network higher up in the hierarchy. As such feedback is proven to exist by experiments this thesis aims to expand the model. The hierarchical network model of Nessler et al. (2013) is taken and expanded to simulate a functional neural network that receives visual input and also feedback from another network higher up in the hierarchy. TODO: 1. summarize what we do and the main results

In Chapter 2 the biological background for this thesis and the underlying hierarchical spiking Winner-Take-All network will be given. The theoretical background, explaining the mathematical model used for the neural networks in this thesis, will be explained in Chapter 3. (Chapter 4 is a work in progress). Finally, in Chapter 5, the results of the experiments will be summarized and their implications will be discussed.

2 Biological background

2.1 Winner-Take-All networks

It has been discovered that pyramidal neurons on layers 2/3 and 5/6 of the neocortex form selection networks. This means that one pyramidal neuron is allowed to fire, while others within the same layer are inhibited. This horizontal selection mechanism determines what is vertically sent on to the neurons of cortical layers higher up in the hierarchy. For example pyramidal neurons in layer 5 have a selection behaviour that determines the output to motor structures. Their inhibition is performed by basket and chandelier cells (Rodney J Douglas, 2004). Basket cells connect to the soma of other neurons and control the action potential discharge rate of those neurons. Their dendritic branches wrap around the target soma, forming a "basket" giving them their name (Jones, 1984). Chandelier cells perform inhibition directly at the axonal initial segment of pyramidal neurons where action potentials are initiated. (Contreras, 2004). The selection of one pyramidal neuron performed this way is called soft winner-take-all mechanism and is used in various neuronal network models (Rodney J Douglas, 2004). The mechanism is called "soft" because it uses a softmax function, which controls how many winners there can be, and how likely each neuron is to win (Arbib, 2003). It has been shown that winner-take-all mechanisms are computationally more powerful when compared to threshold gates (McCulloch-Pitts neurons) and sigmoidal gates. Furthermore, arbitrary continuous functions can be approximated by circuits that only include one soft winner-take-all gate as their only nonlinear operator (Maass, 2000).

2.2 The hierarchical structure of the brain

The brain is made up of modular structures that connect with each other in a hierarchical organization. These modules have a high level of connectivity within themselves and a low level to other modules. This means that there are different categories of neurons within a module, depending on their function in the network. For one there are provincial hub neurons that primarily connect to other neurons of the same module and are responsible for the function the module expresses. Then there are connector hub neurons that transfer information from the module to other modules. Such modules form networks which are connected in a hierarchical manner, where each module is adding to the output of the previous module (Meunier et al., 2009). Most of the information flows upwards along the hierarchy, representing bottom-up observations. But there is physiological experimental data that shows that information is also passed downwards and influences the activity of modules lower in the hierarchy, representing top-down context. This will be explained for the visual cortex.

2.3 Visual cortex

The visual cortex is the region of the brain that processes visual information coming from the retina. From the retina the information is sent to the lateral geniculate nucleus in the thalamus and then further to the primary visual cortex, also called visual area 1 (V1). The visual cortex consists of five visual areas (V1 to V5), which are divided by their function and structure. These areas are located in the occipital lobe of the cerebral cortex. The purpose of the visual areas is to process visual information to recognize objects, perform spatial tasks or to perform visual-motor skills. Neurons of the visual cortex often respond to stimuli within a specific receptive field. It is assumed that each subsequent visual area is more specialized than the previous and due to that neurons in different visual areas respond to the same receptive fields, but to different types of stimuli. There are various specialized cells in the visual cortex. For example simple cells and complex cells are well studied. Simple cells which mainly occur in V1 respond primarily to oriented edges

and lines within a receptive field. For example a simple cell would always generate action potentials when there is a horizontal line in its receptive field. Complex cells can be found in V₁, V₂ and V₃ and also respond primarily to oriented edges and lines. However their receptive fields are larger and an horizontal edge for example does not have to be at a specific location in the receptive field to activate the cell. Some complex cells even respond primarily to movement of edges (Huff T, 2024). This activation of neurons depending on the stimulus is called neuronal tuning. The stimuli the complex cells respond to get more complex, the higher up in the visual cortex hierarchy they are located. For example in the inferior temporal cortex, which receives visual information from V₄, there are complex cells that respond to faces (Riesenhuber and Poggio, 2002). According to Palmer (1999) the larger receptive fields of complex cells are due to the hierarchical convergent nature of visual processing. This property follows from the complex cell receiving input from many simple cells which is summed and integrated.

Visual information first enters V₁, which is the best-understood part of the visual cortex. It consists of six layers that function differently. Layer 4 receives the input from the lateral geniculate nucleus. It has the most simple cells of the six layers and thus processes visual information of small receptive fields. On layers 2, 3 and 6 there are complex cells, which combine the result of layer 4 into larger receptive fields. Through that V₁ outputs simple visual components with their orientation or direction. The processed information is then sent on to V₂ which further processes it and thus responds to more visual complex patterns. V₂ was also found to respond to differences in color and spatial frequency, additional to the more complex patterns and object orientation. After processing V₂ sends its information on to V₃, V₄ and V₅. Furthermore it also has feedback connections to V₁, which will be discussed later. After V₂ the visual information is split up into the dorsal and ventral streams, which each specialize in processing different features of the visual information. The dorsal stream is involved in guidance of motor actions and in recognizing where objects are in space. The ventral stream on the other hand is responsible for object recognition and form representation. (Huff T, 2024)

2.4 Probabilistic brain

When observing the activity of neurons during a task it varies from trial to trial. This suggests that no static neural code exists. It rather seems that the averaged activity of a neural network matters. Because of that neuronal responses are typically treated statistically or probabilistically (Gerstner and Kistler, 2002). Pouget et al. (2013) state that there is strong behavioural and physiological evidence that the brain both represents probability distributions and performs probabilistic inference. Because of that there are several experiments that support a mathematical framework based on Bayesian inference (see Chapter 3.2) to model brain computations. (Funamizu Akihiro, 2016; Lee TS, 2003; Parr and Friston, 2018; Darlington, Beck, and Lisberger, 2018). An advantage of these probabilistic models is their generality, meaning that they can be applied to various tasks that the brain performs (Pouget et al., 2013). Because of this evidence the neural network model of this thesis can be thought to perform Bayesian computation. The mathematical link between the spiking Winner-Take-All network model and Bayesian inference will be shown in Section 3.4.

2.5 Feedback in the visual cortex

Lee TS (2003) showed that feedback in the visual cortex is not only used for attentional selection or biased competition, but also to modulate the processing of the early visual cortex. Furthermore they showed how this modulated processing can be explained via hierarchical Bayesian inference. One hypothetical example for the modulation of lower hierarchical areas that they give is a human looking at a picture of a face which is partially in shadow. First V1 receives the information of the image and performs edge and line detection. In this initial processing the detected edges are in the well lit portion of the image, while in the shadowy part almost no edges are found. As this information is passed on upwards along the visual cortex hierarchy it is processed further, until it reaches the inferior temporal cortex where the conclusion is made that there is a face in the picture. After that, feedback is sent from the inferior temporal cortex back

to V₁. This feedback tells V₁ that there should be edges hidden in the shadow. With this additional prior information V₁ then is able to detect the faint edge in the shadow and complete the contour of the face. In one of their experiments they made a monkey look at a fixation spot on a screen while Kanizsa squares were presented one at a time in different locations. Kanizsa squares are optical illusions where an illusory square can be seen between four partial disks. One such image can be seen in Figure 2.1. The illusory square is seen because the brain chooses the simplest interpretation that a white square is lying on top of the black circles, occluding them. During the presentation neuronal activity in V₁ and V₂ was measured. When looking at a single neuron in V₁, which has a receptive field size of less than 0.8 degrees, they reported that it responds with increased spiking activity within 45 ms after a real line appears in its receptive field. When that neuron has an illusory line of the Kanizsa square in its receptive field it is not active within 45 ms. However after 100 ms it begins to respond, indicating that it starts seeing the illusory line. In contrast, the measured population of 39 V₂ neurons responded to the illusory contour after 65 ms, 35 ms before V₁. They explained this behaviour as V₂ detecting the illusory contour with information from a spatially more global context and then feeding back that information (prior) to V₁. V₁ then, convinced by V₂, starts hallucinating the illusory line, agreeing with the most likely interpretation of the image. This experiment provides strong evidence that feedback is happening in the hierarchical structure of the visual cortex.

2.6 Plasticity

Electrophysiological experiments showed that the response amplitude of neurons changes over time. Depending on the stimulation a neuron receives the postsynaptic response change systematically. These changes can persist short term for hours or days. However, the stimulation paradigm can also induce persistent changes in the synapses of a neuron. These changes are supposed to reflect 'learning' and 'memory'. If the change increases the postsynaptic response it is called long-term potentiation (LTP). If it decreases the response it is called long-term depression (LTD). In neural network models a weight parameter between two neurons is used to indicate

2 Biological background



Figure 2.1: Kanizsa square shown to the monkey by Lee TS (2003)

the strength of the postsynaptic response. When training a neural network model these weights can be adjusted to improve the performance of the network for a given task. By iteratively optimizing the weights the network learns to solve its task. Via a defined learning rule the network is able to determine the way in which to adapt the weights. There are different ways to define such a learning rule. One simple learning rule is to increase the weights of neurons that are more active than others. This is called Rate-Based Hebbian Learning. An improved version of this learning rule is Spike Timing Dependent Plasticity (STDP). It also factors in the exact timing of presynaptic neuronal activity, compared to the timing of postsynaptic neuronal activity (Gerstner and Kistler, 2002). This is the learning rule used for the neural network model of this thesis and will be explained further in the next section.

2.7 Spiking neural networks

Spiking neural networks (SNNs) are artificial neural networks that resemble biological neural networks more closely. Neurons in typical neural networks used in machine learning transmit information at every propagation cycle. This, however, is not how biological neurons operate. They generate action potentials (neuron spikes) to convey information between each other. These action potentials are only generated when their membrane potential exceeds a threshold. SNN models take this behaviour into account by keeping track of each neurons membrane potential and then determining when they should produce an action potential (Gerstner and Kistler, 2002).

Previously it was believed that biological neural networks encode information within the spike rates of neurons. However neurobiological research shows evidence that at high speed processing this alone can not be sufficient. For example image recognition tasks can be performed at a speed at which each neuron in the involved layers has only less than 10 ms to process the information. Such a time frame is too short for rate coding to occur. Instead it has been shown that high speed processing tasks can be performed using the precise timing of spikes. Furthermore it requires more energy for a neuron to spike many times to express a spike rate, rather

2 Biological background

than spiking just once and having the timing of the spike considered. As the brain evolutionarily aims to minimize its energy consumption this is a strong argument for spike timing encoded information. Also the information encoding capacity is higher in a small set of spiking neurons, compared to rate encoding. (Taherkhani et al., 2020) Because of that STDP is often used as learning rule in SNNs. STDP models the synaptic weight changes of neurons depending on the relative timing of pre- and postsynaptic spikes. If a presynaptic spike arrives shortly before the postsynaptic spike the synaptic weight is increased. The size of that increase depends exponentially on the time between both spikes, according to a time constant. However, when the presynaptic spike occurs after the postsynaptic spike the synaptic weight is decreased. These two mechanisms are called long-term potentiation and long-term depression respectively. Although STDP is often modelled like this, biological experiments show that the standard pair-based approach does not fully explain it in biological neurons. However, there is experimental evidence that multiple-spike protocols like triplets STDP are biologically more plausible. (Taherkhani et al., 2020)

3 Theoretical background

3.1 Bayes' theorem

Bayes' theorem describes the probability of an event to occur, depending on the prior knowledge of conditions related to the event (Joyce, 2019).

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (3.1)$$

$P(A|B)$ is the posterior probability of A given that B is true. $P(B|A)$ is called conditional probability of B given A being true. $P(A)$ is the prior probability, the probability of A occurring without any additional information. Finally $P(B)$ is the marginal probability of B without any given condition. This theorem is often used for Bayesian inference where it expresses how a belief, which is represented as probability changes due to related evidence.

3.2 Bayesian inference

Bayesian inference is a process of data analysis to calculate the probability of a hypothesis depending on the available related evidence. As over time more and more evidence becomes available the probabilities can be updated yielding a more sophisticated view on the hypothesis. It is given, according to Bayes' theorem, by

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}, \quad (3.2)$$

where H represents a hypothesis and E some related evidence. $P(H|E)$ is the posterior probability which signifies the probability of the hypothesis

3 Theoretical background

after the evidence was observed. $P(E|H)$ is called likelihood and it is the probability of observing the evidence given the hypothesis. It is a measure of the compatibility of the observed evidence with the given hypothesis. $P(H)$ is the prior probability which is the probability of the hypothesis before any evidence is considered or obtained. $P(E)$ is called marginal likelihood that represents the probability of the evidence being observed. However it is the same independent of the chosen hypothesis. Thus, it is not factored in when comparing different hypotheses. Bayesian inference can be applied to the "face in shadow" example of Section 2.5. There a neuron in V1 sees a small part of the visual field and signals if it sees an edge or not. At first it has the evidence of the observed pixels available and from it can calculate the likelihood that an edge is present. At first it has no prior knowledge of how probable an edge being present is, meaning that the prior probabilities for there being an edge or no edge are equal. With that likelihood and the prior probabilities it can conclude the posterior probability for the hypothesis "there is an edge" and decides that there is no edge. Later the inferior temporal cortex determines that there is a face in the picture and feeds this information back to V1. This influences the prior probabilities and changes the posterior probability in a way that the V1 neuron starts to see the hypothesis that there is an edge as more likely, in order to complete the contour of the face. We now define X as a binary random vector of the visual input, Y as a multinomial variable of the output of V1 and Z as a multinomial variable of the feedback of the inferior temporal cortex. When plugging these variables into Equation 3.2 with the posterior given by $P(Y|X, Z)$, the likelihood given by $P(X|Y)$ and the prior given by $P(Y|Z)$ we get

$$P(Y = i|X = x, Z = j) = \frac{P(X = x|Y = i)P(Y = i|Z = j)}{\sum_k P(X = x|Y = k)P(Y = k|Z = j)}. \quad (3.3)$$

3.3 Network model

The network model used for the experiments in this thesis was taken from Nessler et al. (2013) and expanded by an additional layer to include some hierarchical feedback information.

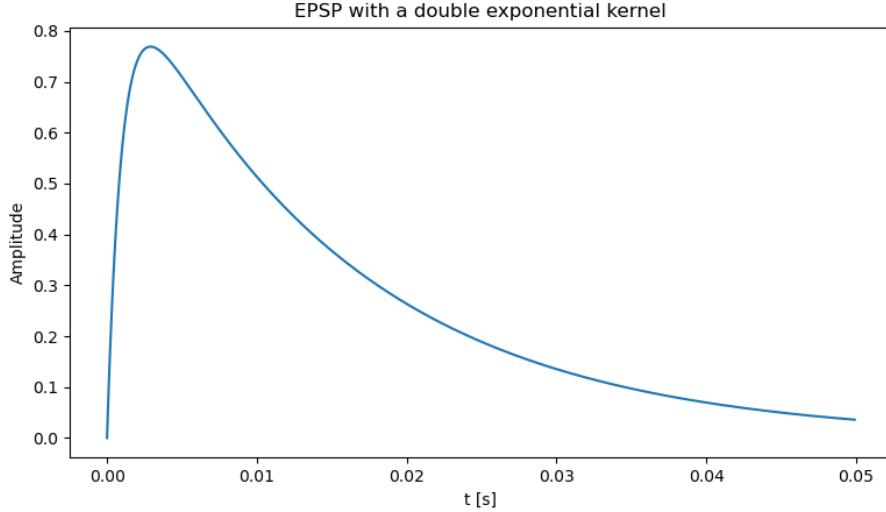


Figure 3.1: Form of an excitatory post synaptic potential generated by an input neuron over time. A double exponential kernel was used to generate this signal. These signals are fed to the next layer of the network.

Neuron model As in Nessler et al. (2013) the input neurons x_1, \dots, x_n are firing according to a poisson process with an average firing rate f_{input} when active and with 0 Hz when in an inactive state. The input neurons receive a binary input, for example a black or white pixel of an image. The excitatory post synaptic potentials (EPSPs) $x_i(t)$ that these neurons produce can be seen in Figure 4.42. A double exponential kernel was used to generate the EPSP. The kernel has a time constant for the rise of the signal τ_{rise} and a time constant for the decay of the signal τ_{decay} . The addition of the time step size δt was necessary to get the time t at the end of the current simulation step. t_f is the time at which the spike of x_i occurred

$$x_i(t) = e^{-(t+\delta t-t_f)/\tau_{decay}} - e^{-(t+\delta t-t_f)/\tau_{rise}}. \quad (3.4)$$

Analogously the EPSP of the prior neurons $z_j(t)$ are given by

$$z_j(t) = e^{-(t+\delta t-t_f)/\tau_{decay}} - e^{-(t+\delta t-t_f)/\tau_{rise}}. \quad (3.5)$$

The membrane potential u_k of each output neuron is calculated by multiplying the EPSP of each input neuron times the weight of the connection

3 Theoretical background

between them, plus the same for the prior neurons

$$u_k(t) = \sum_{i=1}^N w_{ki} \cdot x_i(t) + \sum_{j=1}^J w_{kj} \cdot z_j(t). \quad (3.6)$$

In Nessler et al. (2013) each output neuron y_k also had an intrinsic excitability w_{k0} which was learned for each neuron. For the experiments of this thesis however it was omitted, as the different classes of input images were equally likely, thus the intrinsic excitability of each output neuron would end up being the same.

The output neurons are modelled in a winner-takes-all (WTA) circuit. The WTA behaviour was implemented via an adaptive inhibition signal. The adaptive inhibition is used to regulate the membrane potentials of the output neurons so that all of them together fire with a total firing rate $R(t) = 200\text{Hz}$ on average. Due to that there never is a time window in which no output neuron may fire. However, it is unlikely for an output neuron to fire right after another output neuron has fired.

The inhibition signal was chosen to depend on the current membrane potential of the output neurons. Solving Equation 4.20 for $I(t)$ yields

$$R(t) = \frac{\sum_{k=1}^K e^{u_k(t)}}{e^{I(t)}} \quad (3.7)$$

$$e^{I(t)} = \frac{\sum_{k=1}^K e^{u_k(t)}}{R(t)} \quad (3.8)$$

$$I(t) = \ln \frac{\sum_{k=1}^K e^{u_k(t)}}{R(t)} \quad (3.9)$$

$$I(t) = -\ln R(t) + \ln \sum_{k=1}^K e^{u_k(t)}. \quad (3.10)$$

Nessler et al. (2013) defined that the firing probability of an output neuron y_k is exponentially proportional to its membrane potential u_k minus the received inhibition $I(t)$

$$p(y_k \text{ fires at time } t) \propto e^{u_k(t) - I(t)}. \quad (3.11)$$

This stochastic firing model is supported by experimental biological evidence (Jolivet et al., 2006). Through this definition the firing rate $r_k(t)$ of an output neuron is then modelled by an inhomogeneous Poisson process as

$$r_k(t) = e^{u_k(t)-I(t)}. \quad (3.12)$$

At every timestep of the simulation the inhibition signal $I(t)$ is subtracted from the membrane potential $u_k(t)$ of every output neuron. By that the membrane potentials are altered to always yield a spiking frequency of 200 Hz, regardless if it would be lower or higher without it. This means that the adaptive inhibition signal can also function as an excitatory signal.

The total firing rate of the output neurons is obtained when summing up the firing rates of all output neurons, yielding

$$R(t) = \sum_{k=1}^K e^{u_k(t)-I(t)}. \quad (3.13)$$

The probability of an individual output neuron to fire within a time step δt is given by

$$r_k(t)\delta t. \quad (3.14)$$

The conditional probability $q_k(t)$ that a spike originated from the output neuron y_k is given by

$$q_k(t) = \frac{r_k(t)\delta t}{R(t)\delta t} = \frac{e^{u_k(t)-I(t)}}{\sum_{k'=1}^K e^{u_{k'}(t)-I(t)}}. \quad (3.15)$$

Spike timing dependent plasticity The weights w_{ki} between neurons x_i and y_k are updated whenever an output neuron fires. σ indicates the time window, after which spikes are no longer considered. If y_k produces a spike all its weights to the input neurons are updated as

$$\Delta w_{ki} = \begin{cases} \lambda \cdot (ce^{-w_{ki}} - 1) & \text{if } x_i \text{ fired in } [t^f - \sigma, t^f] \\ \lambda \cdot (-1) & \text{if } x_i \text{ did not fire in } [t^f - \sigma, t^f], \end{cases} \quad (3.16)$$

3 Theoretical background

where λ is the learning rate, the parameter c shifts the weight values, t^f is the time when y_k spiked and σ is the time window in which input spikes are considered as "before" an output spike. As the membrane potentials u_k of the output neurons result from the addition of the EPSPs of the input neurons times the corresponding weight, a way to control the average size of u is needed. If u is too small the output neurons will fire too sparsely and if u is too big it will impair the learning process. So to limit u , the size of the weights is controlled via the parameter c . The learning rate λ is needed to control the size of each weight update. If it is too big few output neurons will respond to multiple input classes and others will only respond too little or not at all. On the other hand if λ is too small the network will learn very slowly and may never converge. The prior weights w_{kj} are also updated whenever an output neuron fires, in the same way as w_{ki}

$$\Delta w_{kj} = \begin{cases} \lambda \cdot (ce^{-w_{kj}} - 1) & \text{if } z_j \text{ fired in } [t^f - \sigma, t^f] \\ \lambda \cdot (-1) & \text{if } z_j \text{ did not fire in } [t^f - \sigma, t^f], \end{cases} \quad (3.17)$$

Network architecture Each pixel of an input image was connected to two neurons. The first of these neurons is in an active state when the pixel is black and in an inactive state otherwise. The second neuron expresses the opposite behaviour. As a consequence the network needs $(width \cdot height \cdot 2)$ excitatory input neurons x_1, \dots, x_N . These input neurons are fully connected to the excitatory output neurons y_1, \dots, y_K . This means that every input neuron x_i is connected to each output neuron y_k . To simulate the feedback from the inferior temporal cortex prior neurons z_1, \dots, z_J were added to the network. Each prior neuron was connected only to the one output neuron that represents the same class as the prior. As there are fewer prior neurons than input neurons influencing each output neuron, they have higher firing frequencies. Other than that the prior neurons behave the same as the input neurons and their EPSPs are handled the same way.

3.4 Mathematical link between the spiking Winner-Take-All network model and Bayesian inference

Nessler et al. (2013) hypothesized that the ensemble of weights of a neuron can be understood as a generative model. They further claimed that every synaptic weight, due to STDP-induced changes, converges stochastically to the log of the conditional probability that the presynaptic neuron has fired just before the postsynaptic neuron, given that the postsynaptic neuron fires. This connection is given by

$$w_{ki} = \log(p(x_i = 1 | y_k = 1)) \quad (3.18)$$

an will be analysed in Section 4.1. Furthermore they claimed that in a Bayesian inference context every input spike provides evidence for an observed variable and every output spike represents one stochastic sample from the posterior distribution over hidden causes, which are encoded in the circuit.

To show the connection between the spiking Winner-Take-All network model and Bayesian inference it will be shown that the relative firing probability $q_k(t)$, given in Equation 3.15, is equal to the posterior probability of Bayesian inference given in Equation 4.1.

TODO:

1. im network model angeben wie y (output) berechnet wird!!!

4 Experiments

The time constant for the rise of the signal τ_{rise} was 1 ms and the time constant for the decay of the signal τ_{decay} was 15 ms.

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

4.1.1 Introduction

The purpose of this experiment was to mathematically analyse a simpler network, thus providing a definitive way to verify the performance of the simulation. The network was scaled down to be one-dimensional with 18 input neurons, four prior neurons and four output neurons, making it easier to analyse. The conditional probabilities of the input and output neurons of the network were calculated by hand and then used to determine the theoretical posterior probability of the network. These probabilities were then used to calculate the weights for the simulation of the network. The distribution of the output spikes was then used to calculate the posterior probability of the simulation. The posterior probabilities of both the mathematical analysis and the simulation are compared, and by varying three parameters of the simulation it is tuned to approximate the analytical solution as closely as possible.

4.1.2 Methods

Network structure The architecture of this network remained the same as in previous experiments (with adaptive inhibition and a noise level of 10%), only the amount of neurons was changed. The input image consists of nine pixels in a horizontal line. Within those nine pixels four output classes can be represented. Each output class consists of three pixels next to each other. This results in each output class overlapping its neighbour classes by one pixel. Thus the centres of the output classes are at position 1, 3, 5 and 7. As there have to be 2 input neurons for each pixel, one neuron being active if the pixel is white and one if the pixel is black, the network has 18 input neurons. Furthermore four prior neurons were implemented, of which only one is being active for one of the output classes at a time. Lastly the network has four output neurons.

Mathematical Analysis The posterior probability of the network was calculated by using

$$P(Y = i|X = x, Z = j) = \frac{P(X = x|Y = i)P(Y = i|Z = j)}{\sum_k P(X = x, Y = k)P(Y = k|Z = j)}. \quad (4.1)$$

$P(X = x|Y = i)$ and $P(Y = i|Z = j)$ were derived by hand corresponding to the paradigm of the experiment. The calculation of $P(X = x|Y = i)$ was split into two parts. First the contribution of the active input neurons was calculated by determining the matrix $P^{X|Y}$. This matrix is of size 4×9 and contains the conditional probabilities of each input neuron y_i being active, given that an output neuron x_l is active. These probabilities were calculated by determining which input neurons are active depending on the output class and which input neurons are inactive, as dictated by the network structure. Furthermore the noise that was applied to the input neurons had to be taken into account. For input neurons belonging to the output class the conditional probability was determined as 1 and lowered by the noise level, while for the input neurons outside of the 3-wide pixel block belonging to the output class it was determined as 0 and raised by the noise level.

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

According to this algorithm $P^{X|Y}$ is given by

$$P_{i,l}^{X|Y} = \begin{cases} 1 - \text{noise level} & \text{if } x_l \text{ is in the active area of } y_i, \\ 0 + \text{noise level} & \text{if } x_l \text{ is not in the active area of } y_i \end{cases}. \quad (4.2)$$

After calculating all entries of $P^{X|Y}$ its rows were multiplied with the input image vector

$$P_1(X = x|Y = i) = P_{i,*}^{X|Y} \cdot x \quad (4.3)$$

resulting in a conditional probability for each output class depending on the input vector. The input vector was given with entries of 1 for active pixels and with entries of 0 for inactive pixels. Next to include the contribution of the input neurons that are spiking when a pixel is inactive the conditional probability of the input neurons that are active for the entries where $x_i = 0$ had to be calculated. To achieve this complementary conditional probability at first $P^{X|Y}$ was subtracted from one. To include the correct conditional probabilities for the complementary case the input image vector x was then also subtracted from one. $1 - P^{X|Y}$ and $1 - x$ were multiplied to yield

$$P_2(X = x|Y = i) = (1 - P_{i,*}^{X|Y}) \cdot (1 - x). \quad (4.4)$$

The results of both calculations were then multiplied element-wise to yield

$$P(X = x|Y = i) = P_1(X = x|Y = i) \odot P_2(X = x|Y = i). \quad (4.5)$$

$P(Y = i|Z = j)$ was determined by first calculating the matrix $P^{Y|Z}$. It has a dimension of 4×4 and contains the conditional probabilities for the output neuron y_i , given the prior neuron z_j being active. For each output class there exists one corresponding prior neuron. As there can never be more than one active prior neuron at a time $P^{Y|Z}$ is given by

$$P_{j,i}^{Y|Z} = \begin{cases} 1 - \text{noise level} & \text{if } i = j, \\ 0 + \frac{1}{3} \text{noise level} & \text{if } i \neq j. \end{cases} \quad (4.6)$$

$P(Y = i|Z = j)$ was then obtained by

$$P(Y = i|Z = j) = P^{Y|Z} \cdot z \quad (4.7)$$

where z is given by a 4×1 one-hot encoded vector of the prior.

4 Experiments

Simulation The input weights for the simulation were calculated as two separate sets. First weights w_1 for the input neurons that are active for active input pixels were determined by

$$w_1 = \ln(P^{X|Y}). \quad (4.8)$$

Second complementary input weights w_2 were calculated for input neurons representing non active input pixels with

$$w_2 = \ln(1 - P^{X|Y}). \quad (4.9)$$

The prior weights w_p were derived by

$$w_p = \ln(P^{Y|Z}). \quad (4.10)$$

The network was simulated for six different input images for each parameter set. After the image presentation period of each input image the amount of the output spikes of different classes were counted and their proportions were calculated to yield the "simulation output probabilities" in later plots. The Kullback-Leibler divergence was chosen to compare the divergence of the analytic and the simulated results of the network. This metric indicates how much two probability distributions diverge from each other. The goal of the parameter search of the simulation was to minimize the Kullback-Leibler divergence. Upon completion of the simulation of an output image the Kullback-Leibler divergence was calculated for that image

$$\begin{aligned} D_{KL}(P_{analysis}(Y = i|X, Z) || P_{simulation}(Y = i|X, Z)) = \\ \sum_{i=1}^K P_{analysis}(Y = i|X, Z) \cdot \ln\left(\frac{P_{analysis}(Y = i|X, Z)}{P_{simulation}(Y = i|X, Z)}\right) \end{aligned} \quad (4.11)$$

where K is the number of output neurons and $P_{analysis}(Y = i|X, Z)$ and $P_{simulation}(Y = i|X, Z)$ are the "analysis output probabilities" and "simulation output probabilities". The six resulting Kullback-Leibler divergences for each parameter set were then averaged to create a single metric by which the performance of the different parameter sets was compared. The three parameters input firing rate f_{input} , prior firing rate f_{prior} and the membrane constant τ_{decay} were varied to inspect their influence on the result, as well

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

as to approximate the analytical solution as closely as possible. Each input image is presented to the network for 20 seconds to reduce the variance between runs. Furthermore each simulation was repeated 20 times with the same parameter set to obtain the mean and standard deviation of the simulation output probabilities and of the Kullback-Leibler divergence.

4.1.3 Results

Analytic Results First the matrix $P^{X|Y}$ was analytically calculated by using Equation 4.2

$$P^{X|Y} = \begin{bmatrix} 0.9 & 0.9 & 0.9 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.9 & 0.9 & 0.9 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.9 & 0.9 & 0.9 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.9 & 0.9 & 0.9 \end{bmatrix}. \quad (4.12)$$

Next the matrix $P^{Y|Z}$ was analytically calculated by using Equation 4.6

$$P^{Y|Z} = \begin{bmatrix} 0.9 & 0.0333 & 0.0333 & 0.0333 \\ 0.0333 & 0.9 & 0.0333 & 0.0333 \\ 0.0333 & 0.0333 & 0.9 & 0.0333 \\ 0.0333 & 0.0333 & 0.0333 & 0.9 \end{bmatrix}. \quad (4.13)$$

For each input image these two matrices were then multiplied with the input vector and prior vector as described Equations 4.3, 4.4, 4.5 and 4.7 to yield $P(X = x|Y = i)$ and $P(Y = i|Z = j)$. Using 4.1 then yielded $P(Y = i|X = x, Z = j)$ also called "Analysis output probabilities" in later plots.

Simulation results without prior To simplify the parameter fitting process the network was at first simulated with inactive prior neurons. Only after determining the best values for f_{input} and τ_{decay} were the prior neurons reactivated and f_{prior} was fitted. The values 0.015 seconds and 0.004 seconds for τ_{decay} were used for the simulation and compared. For each of these values an optimal value for f_{input} was found by simulating different values and looking for the value that yields the smallest Kullback-Leibler divergence.

4 Experiments

$\tau_{decay} = 0.015\text{seconds}$, $f_{prior} = 0\text{Hz}$ Values for f_{input} between 10 and 110 Hz in steps of 10 Hz were simulated. After identifying the input firing rate with the smallest Kullback-Leibler divergence the network was simulated again for f_{input} values ± 10 Hz in steps of 2 Hz of the best value. The result of this search can be seen in Figure 4.1. This process yielded the best input firing rate of 42 Hz. The results of this parameter combination can be seen in Figure 4.2. In this figure six different input images can be seen marked with the letter "A". The active pixels are shown in black. The posterior probability of the mathematical analysis can be seen next to the letter "B". And finally the proportion of output spikes of the different output neurons can be seen next to the letter "C". Each probability of "C" has a standard deviation next to it, which was calculated over 20 runs of the network. At the bottom the value of the Kullback-Leibler divergence is given, with its standard deviation next to it.

When f_{input} was 70 Hz the results of the analysis and the simulation differed more as can be seen in Figure 4.3.

$\tau_{decay} = 0.004\text{seconds}$, $f_{prior} = 0\text{Hz}$ For this parameter combination values between 50 and 150 Hz for f_{input} in steps of 10 Hz were simulated. Analogously to the previous parameter set after finding the best input firing rate the search was performed in finer steps until the best value was determined as 88 Hz. The result of this search can be seen in Figure 4.4. The result of the simulation with those parameters can be seen in Figure 4.5.

Simulation results with prior After determining the best input firing rates for two different values of τ_{decay} , the prior neurons were activated and the best prior firing rate f_{prior} was fitted.

$\tau_{decay} = 0.015\text{seconds}$, $f_{input} = 42\text{Hz}$ The search for the best value of f_{prior} was performed in the same manner as for f_{input} . Values between 140 and 240 Hz were simulated and a prior firing rate of 222 Hz performed the best. The result of this search can be seen in Figure 4.6. The simulation results can be seen in Figure 4.7. The value of the prior is indicated by the red

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

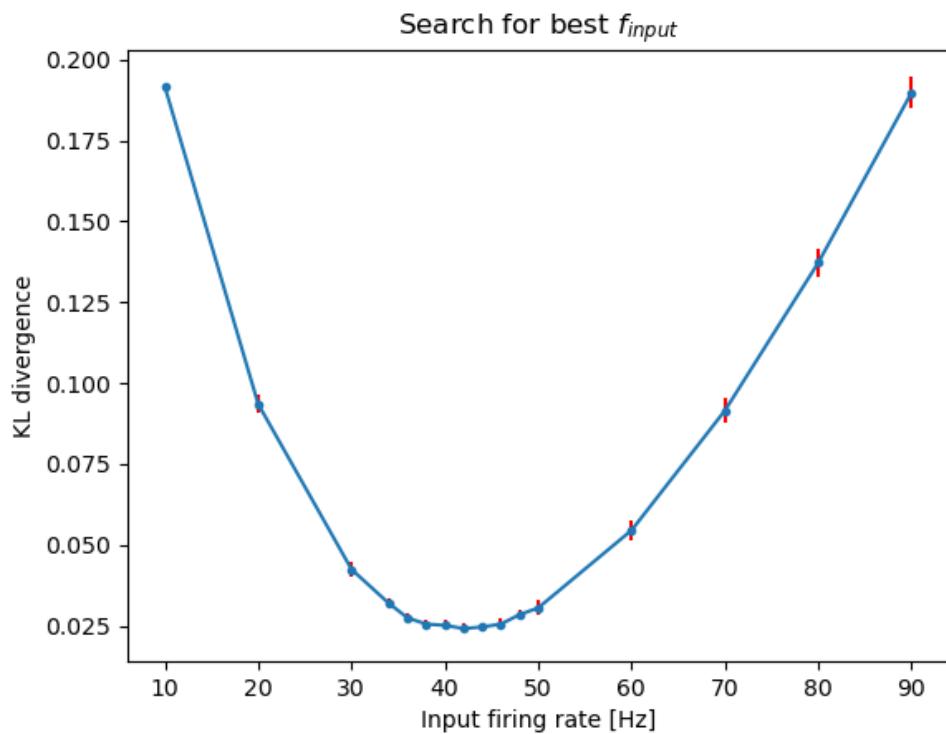


Figure 4.1: KL divergence for different f_{input} values $f_{prior} = 0\text{Hz}$, $\tau_{decay} = 15\text{ms}$

4 Experiments

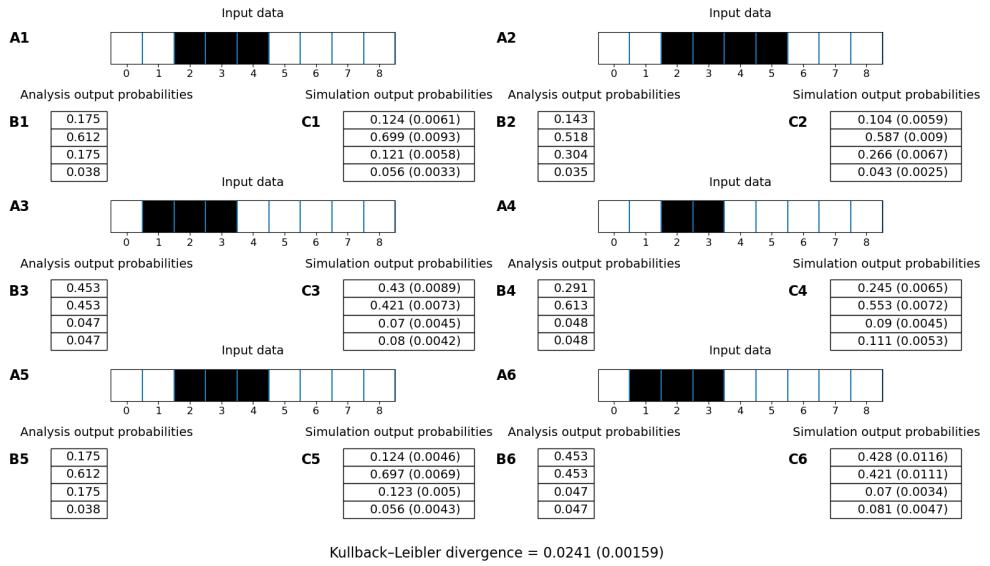


Figure 4.2: Analysis and simulation result. Parameters: $f_{input} = 42\text{Hz}$, $f_{prior} = 0\text{Hz}$, $\tau_{decay} = 15\text{ms}$ **A** Input images with 9×1 pixels. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

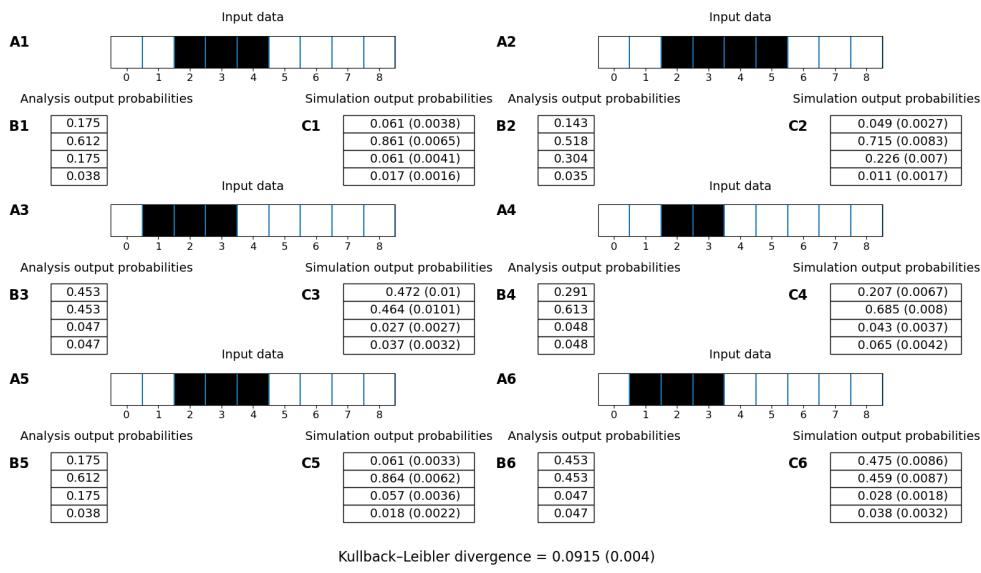


Figure 4.3: **Analysis and simulation result.** Parameters: $f_{input} = 70\text{Hz}$, $f_{prior} = 0\text{Hz}$, $\tau_{decay} = 15\text{ms}$ **A** Input images with 9×1 pixels. The red borders indicate the class of the prior. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4 Experiments

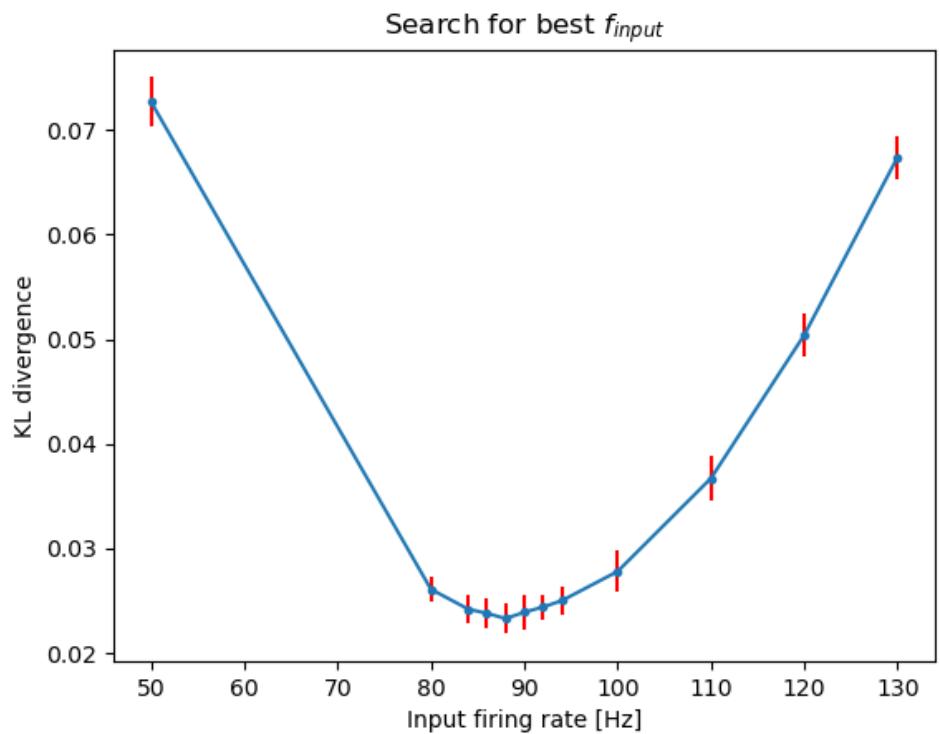


Figure 4.4: KL divergence for different f_{input} values $f_{prior} = 0\text{Hz}$, $\tau_{decay} = 4\text{ms}$

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

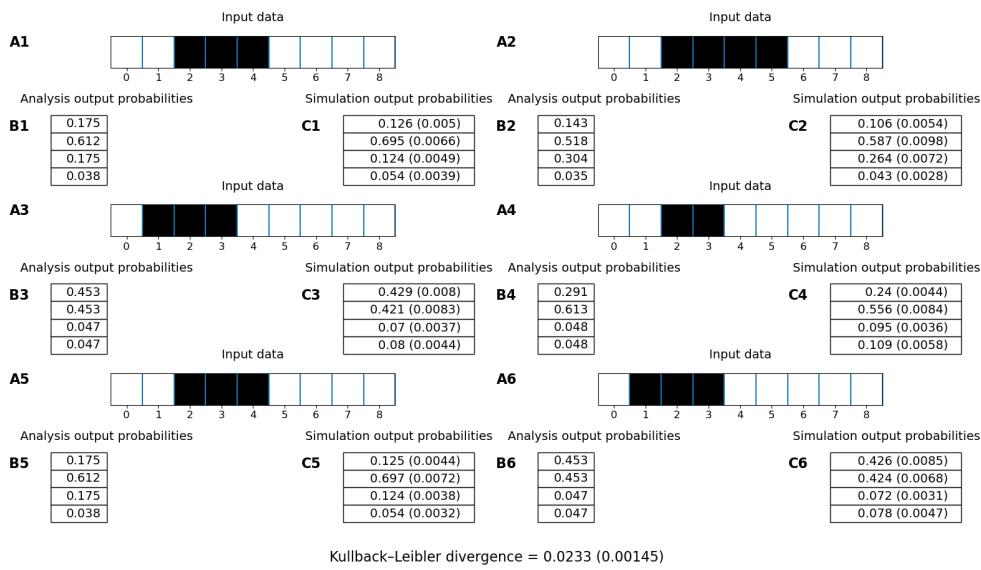


Figure 4.5: **Analysis and simulation result.** Parameters: $f_{input} = 88\text{Hz}$, $f_{prior} = 0\text{Hz}$, $\tau_{decay} = 4\text{ms}$ **A** Input images with 9×1 pixels. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4 Experiments

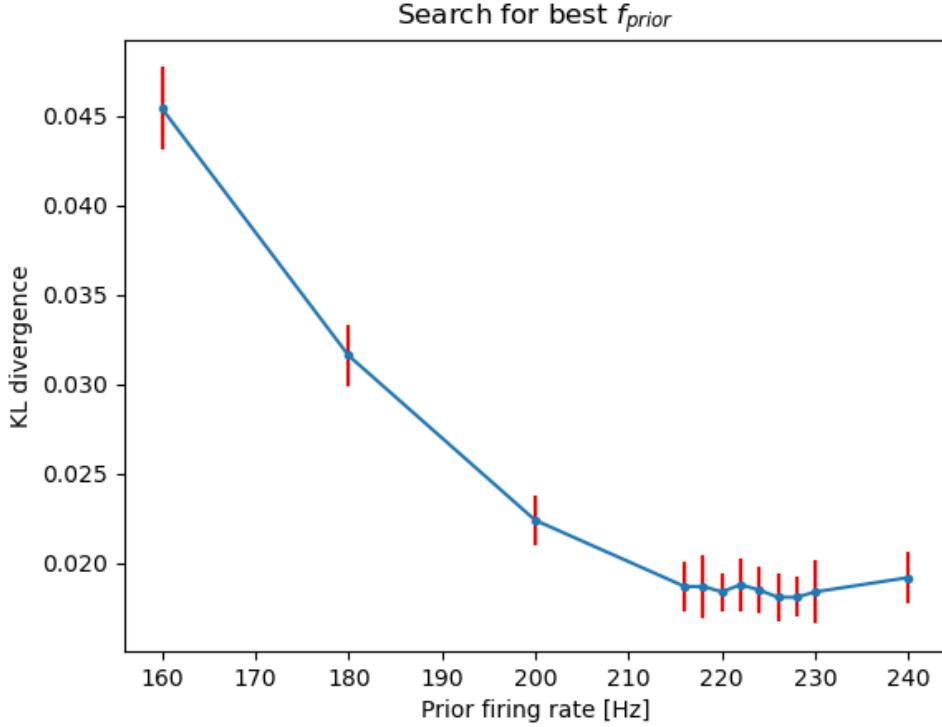


Figure 4.6: KL divergence for different f_{prior} values $f_{input} = 42\text{Hz}$, $\tau_{decay} = 15\text{ms}$

border which is three pixels wide and centered at the center position of the corresponding output class.

$\tau_{decay} = 0.004\text{seconds}$, $f_{input} = 88\text{Hz}$ The search for the best value of f_{prior} was performed in the same manner as for f_{input} . Values between 360 and 460 Hz were simulated and a prior firing rate of 440 Hz performed the best. The result of this search can be seen in Figure 4.8. The results of this parameter combination are given in Figure 4.9.

To demonstrate the impact of rising f_{prior} it was set to 600 Hz and the results can be seen in Figure 4.10.

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

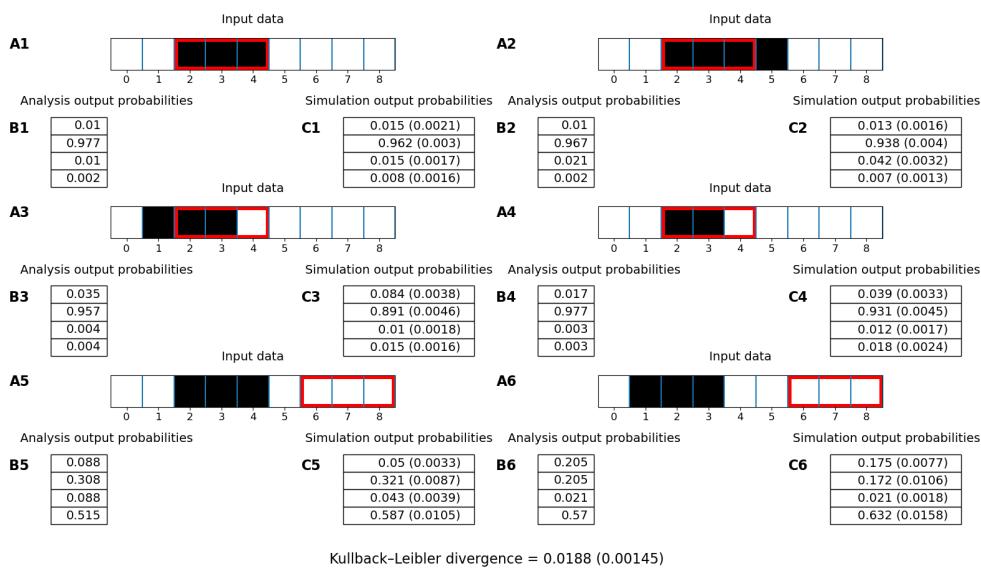


Figure 4.7: **Analysis and simulation result.** Parameters: $f_{input} = 42\text{Hz}$, $f_{prior} = 222\text{Hz}$, $\tau_{decay} = 15\text{ms}$. **A** Input images with 9×1 pixels. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4 Experiments

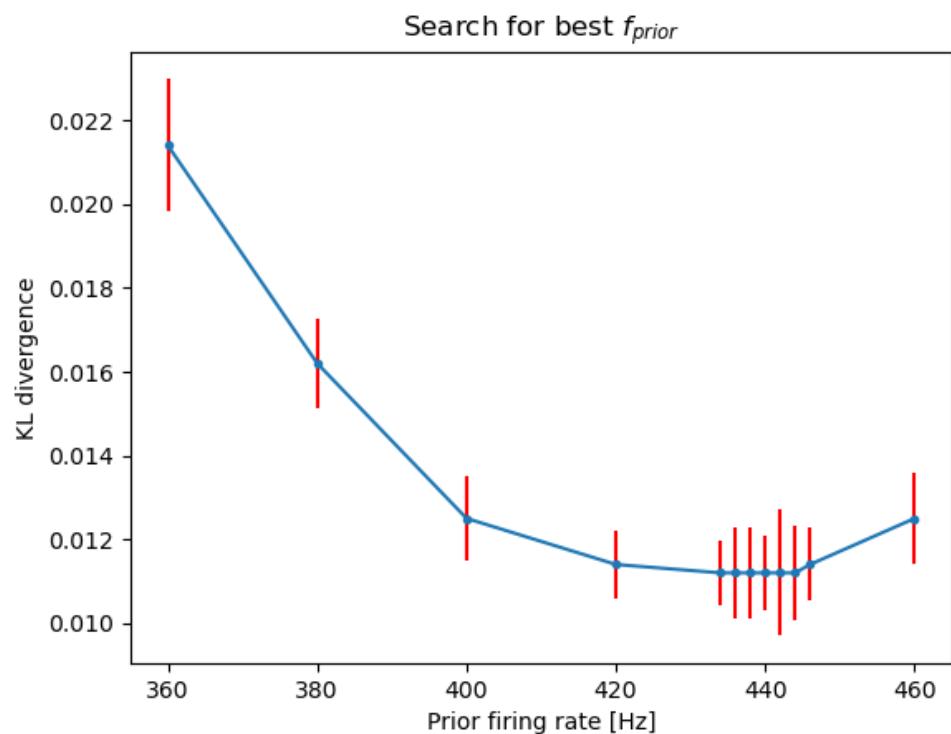


Figure 4.8: KL divergence for different f_{prior} values $f_{input} = 88\text{Hz}$, $\tau_{decay} = 4\text{ms}$

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

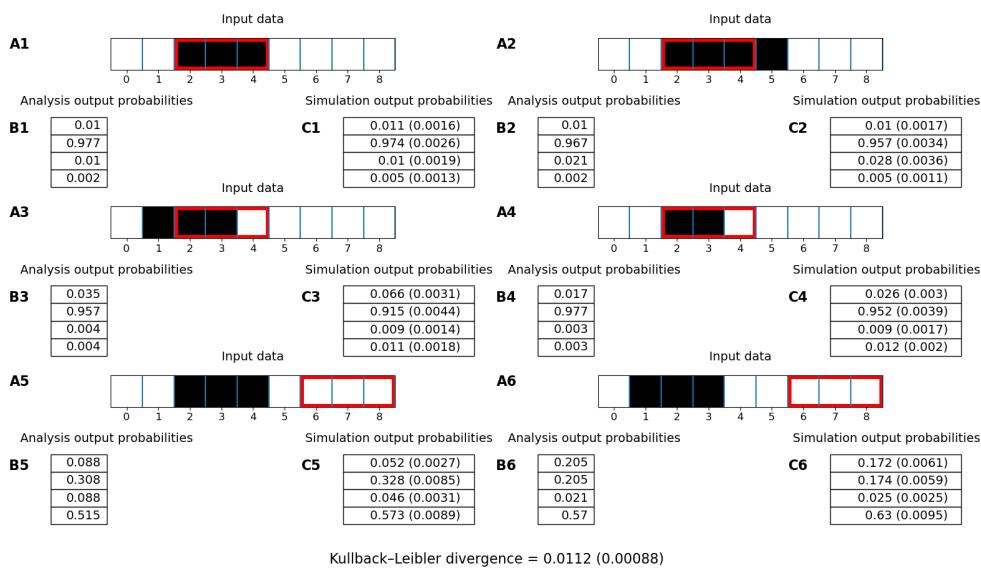


Figure 4.9: **Analysis and simulation result.** Parameters: $f_{input} = 88\text{Hz}$, $f_{prior} = 440\text{Hz}$, $\tau_{decay} = 4\text{ms}$ **A** Input images with 9×1 pixels. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4 Experiments

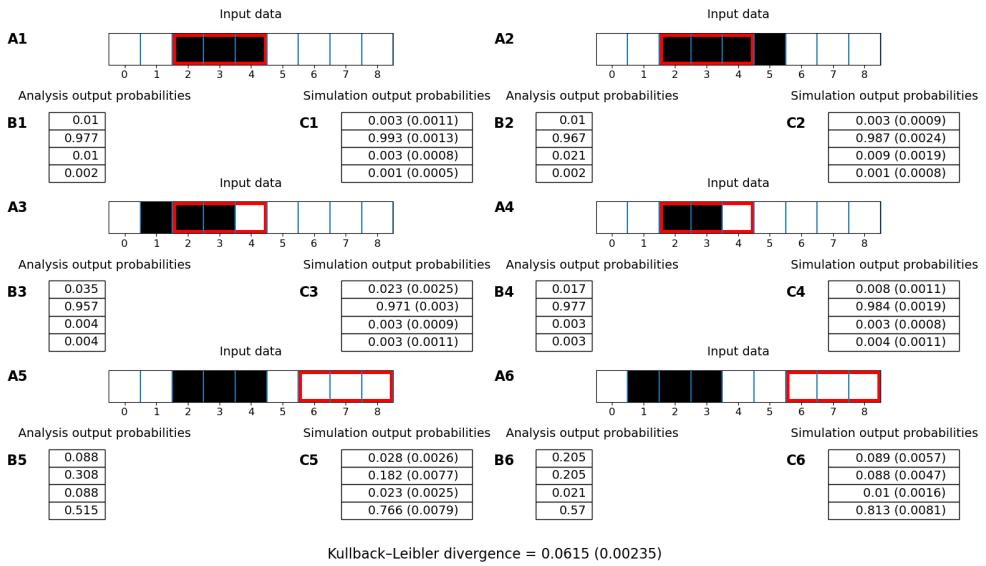


Figure 4.10: Analysis and simulation result. The Kullback-Leibler divergence could not be calculated for this case, as it is not defined for probabilities of 0.

Parameters: $f_{input} = 88\text{Hz}$, $f_{prior} = 600\text{Hz}$, $\tau_{decay} = 4\text{ms}$

A Input images with 9×1 pixels. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

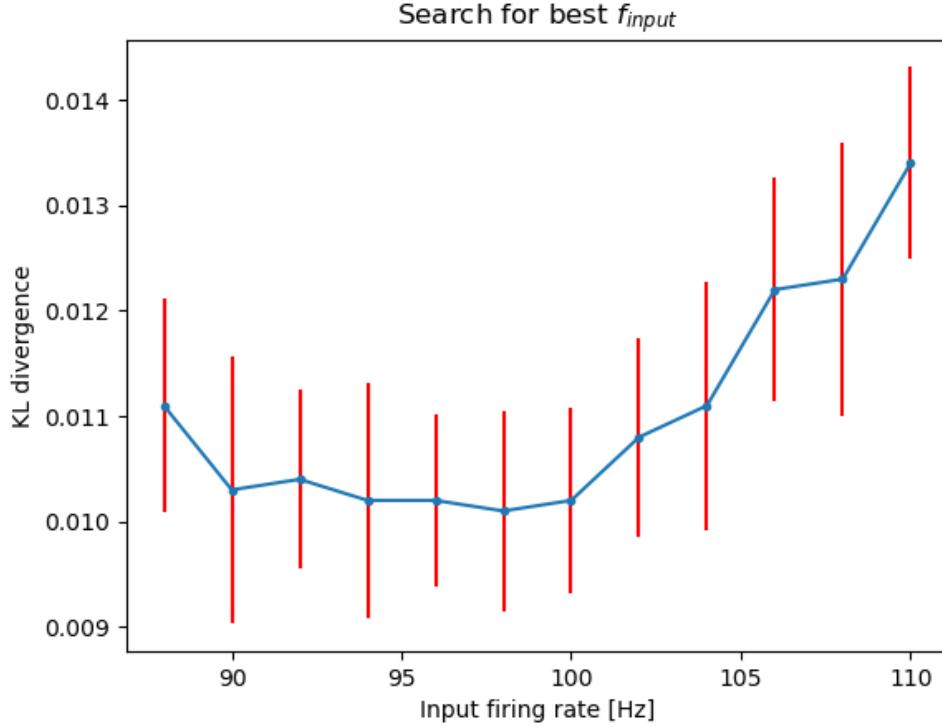


Figure 4.11: KL divergence for different f_{input} values $f_{prior} = 440\text{Hz}$, $\tau_{decay} = 4\text{ms}$

$\tau_{decay} = 0.004\text{seconds}$, $f_{input} = 98$, $f_{prior} = 440\text{Hz}$ Finally to it was tested if a increase of f_{input} might decrease the Kullback-Leibler divergence even further. The network was simulated with increasing values of f_{input} in steps of 2 Hz, beginning from 90 Hz. The result of this search can be seen in Figure 4.11. The best result was obtained with an input firing rate of 98 Hz. The results of that simulation can be seen in Figure 4.12.

4.1.4 Discussion

In this experiment the impact of the three network parameters f_{input} , f_{prior} and τ_{decay} was analysed.

4 Experiments

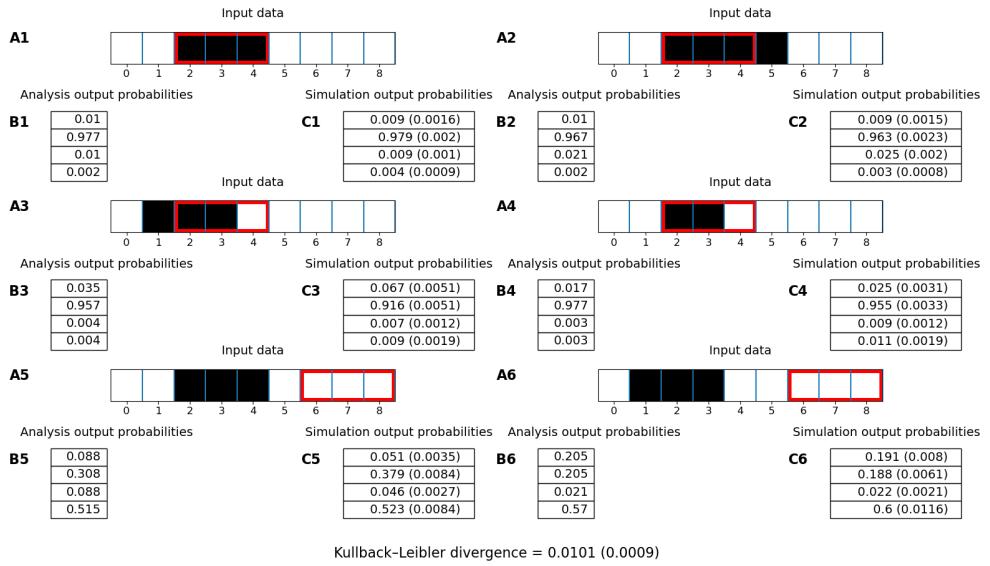


Figure 4.12: Analysis and simulation result. Parameters: $f_{input} = 98\text{Hz}$, $f_{prior} = 440\text{Hz}$, $\tau_{decay} = 4\text{ms}$ **A** Input images with 9×1 pixels. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4.1 Experiment 5: Mathematical analysis and simulation of a 1D network

f_{input} controls how strongly the information of the pixels of the input image is weighed. This means that by raising f_{input} the impact of the active pixels increases, while the impact of the prior neuron decreases comparatively. This effect will be shown in the discussion about f_{prior} . However this is not the only impact this parameter has. When raising f_{input} it was also observed that the probabilities for output classes adjacent to the active pixels decreased. This can be observed when comparing Figures 4.2 and 4.3 where f_{input} was raised from 42 Hz to 70 Hz. For example when looking at "C4" in the mentioned figures, it can be seen that for $f_{input} = 42\text{Hz}$ the simulation output probability for class 1 is 0.245. This high probability is mostly due to the active pixel number 2, which belongs to class 1 and 2 at the same time. When now increasing f_{input} to 70 Hz one might expect the probabilities for class 1 and 2 to rise. This however does not happen, instead the simulation output probability for class 1 fell to 0.207, while for class 2 it rose. It is assumed that this happens because the membrane potentials of the output neurons are never normalized. As the input neurons spike more quickly the membrane potential of output neuron 1 rises slower than the membrane potential of output neuron 2. This leads to shifted firing rates in favour of output neuron 2. However this behaviour might be expected, as faster spiking input neurons correspond statistically to taking more samples of a probability distribution. This means that the more samples the network takes, the more certain it becomes of the more likely option, which is output class 2.

f_{prior} When increasing the prior firing rate the impact on the result of the prior neurons increases, while the impact of the input neurons decreases comparatively. When comparing the Figures 4.9 and 4.10 which show the results for $f_{input} = 88\text{Hz}$, $\tau_{decay} = 4\text{ms}$ and $f_{prior} = 440$ and 600Hz respectively it can be seen under "C6" how the probability for output class 4 rose, while all other probabilities fell. The opposite behaviour can be observed when raising f_{input} instead, but no separate figure will be provided for this case.

τ_{decay} determines for how long and how strongly an input or prior spike contributes to the membrane potentials of the output neurons. The bigger

τ_{decay} is, the smaller f_{input} and f_{prior} need to be, to minimize the Kullback-Leibler divergence. This is represented by the final parameters for τ_{decay} of 4 and 15 ms. While for $\tau_{decay} = 15\text{ms}$ the best input firing rate was 42 Hz and the prior firing rate was 222 Hz, for $\tau_{decay} = 4\text{ms}$ they had to be raised to 98 and 440 Hz to minimize the Kullback-Leibler divergence.

Approximating the analytic solution After first finding the optimal f_{input} for disabled prior activity and then finding the optimal f_{prior} with enabled prior activity for τ_{decay} the simulation was not yet approximating the analytical solution perfectly. Because of that it was tried to raise f_{input} to further decrease the Kullback-Leibler divergence. The further search for a better f_{input} was performed for $\tau_{decay} = 0.004\text{ms}$ and $f_{prior} = 440\text{Hz}$, as this set of parameters performed the best overall. When comparing the former optimal result in Figure 4.9 with $f_{input} = 88\text{Hz}$, to the result with $f_{input} = 98\text{Hz}$ in Figure 4.12, it can be seen that the Kullback-Leibler divergence decreased from 0.0112 to 0.0101. The simulation output probabilities with the higher f_{input} approximated the analysis output probabilities for example for "B₃" and "C₃" in the mentioned figures on one hand better, while on the other hand for "B₅" and "C₅" the output probabilities of the analysis and the simulation of class 2 diverged more. A further increase of f_{input} started to increase the Kullback-Leibler divergence again, thus the final optimum of the parameter set was reached. It is assumed that increasing f_{input} after f_{prior} was fitted was necessary because the proportion of the impact of f_{input} on the simulation decreased due to the introduction of the prior activity.

4.2 Experiment 6: Simulation of the 1D network with double size

4.2.1 Introduction

In this experiment the optimal parameters determined in Experiment 5 were used for a network double in size of the network in Experiment 5. By doing

4.2 Experiment 6: Simulation of the 1D network with double size

this the applicability of parameters determined for one network to another network was to be examined.

4.2.2 Methods

The network of Experiment 5 was doubled in size by increasing the number of input neurons from 18 to 36 neurons. This resulted in input images with 18 pixels. Furthermore the matrix $P^{X|Y}$ was doubled in size by inserting each value of the matrix to the right of itself. The amount of the prior neurons was kept the same with 4 prior neurons. However this network was simulated once for a prior firing rate of 440 Hz, and once for the double rate of 880 Hz. The rest of the methods were performed analogously to Experiment 5.

4.2.3 Results

The expanded matrix $P^{X|Y}$ was doubled in size, now being $[18 \times 4]$ yielding

$$P^{X|Y} = \begin{bmatrix} 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.1 & 0.1 & 0.1 & \dots \\ \dots & 0.1 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.1 & \dots \\ \dots & 0.1 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.1 & \dots \\ \dots & 0.1 & 0.1 & 0.1 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 \end{bmatrix}. \quad (4.14)$$

The simulation result for $f_{input} = 98\text{Hz}$, $f_{prior} = 440\text{Hz}$ and $\tau_{decay} = 4\text{ms}$ can be seen in Figure 4.13. The Kullback-Leibler divergence is infinity in this result, because some simulation output probabilities were zero and the Kullback-Leibler divergence is not defined for probabilities of zero. To circumvent this an approximated Kullback-Leibler divergence was calculated by setting the simulation output probabilities that were 0 to 0.0000001. This yielded a Kullback-Leibler divergence of 0.1411.

Finally the simulation result for $f_{input} = 98\text{Hz}$, $f_{prior} = 880\text{Hz}$ and $\tau_{decay} = 4\text{ms}$ can be seen in Figure 4.14. As in the previous result the Kullback-Leibler

4 Experiments

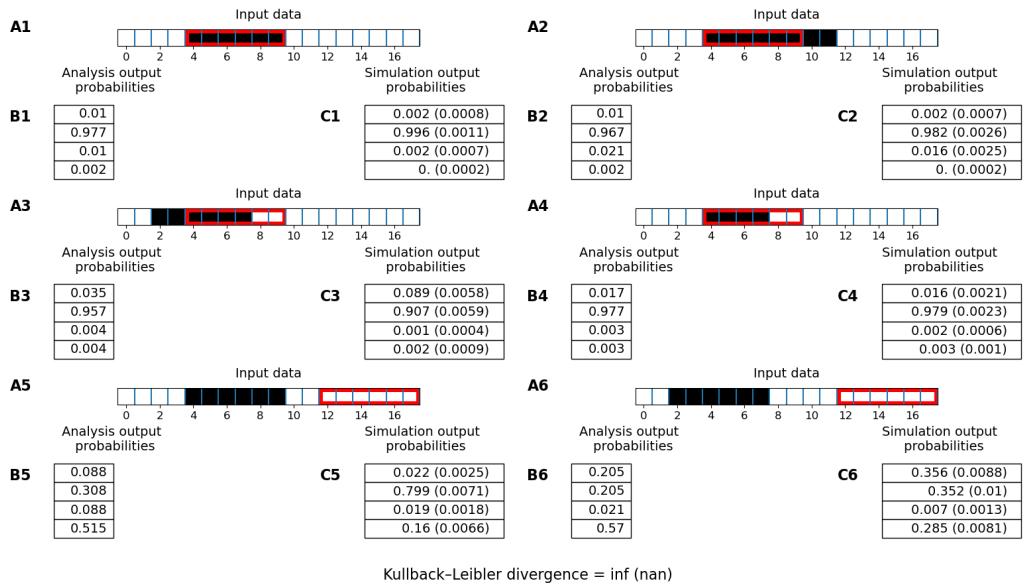


Figure 4.13: **Analysis and simulation result.** Parameters: $f_{input} = 98Hz$, $f_{prior} = 440Hz$, $\tau_{decay} = 4ms$ **A** Input images with 18×1 pixels. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4.2 Experiment 6: Simulation of the 1D network with double size

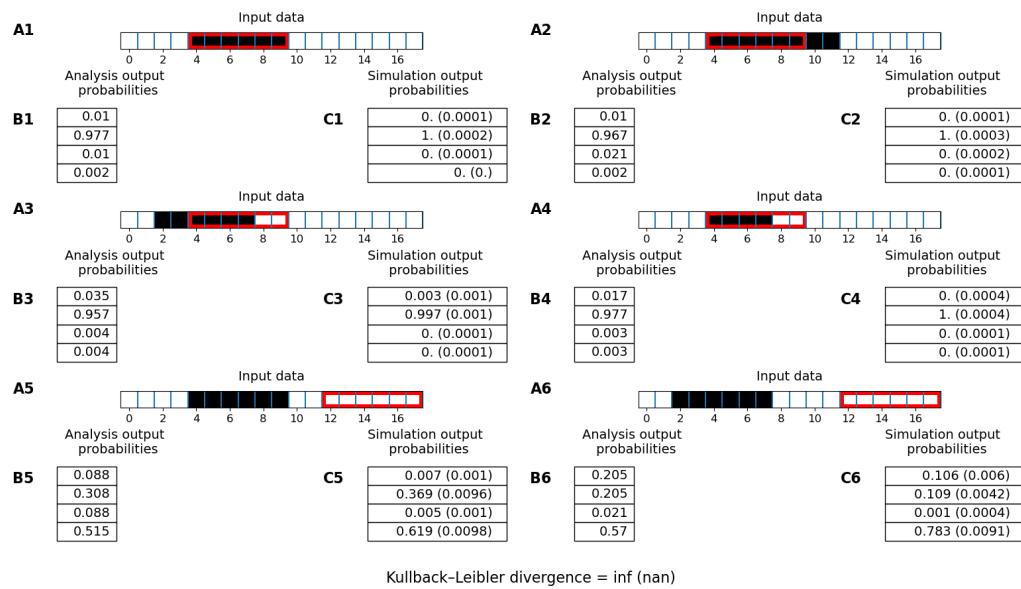


Figure 4.14: **Analysis and simulation result.** Parameters: $f_{input} = 98Hz$, $f_{prior} = 880Hz$, $\tau_{decay} = 4ms$ A Input images with 18×1 pixels. B Analytically calculated posterior probabilities. C Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4 Experiments

divergence was approximated and calculated as 0.2391 for this parameter set.

Finally the Kullback-Leibler divergence for the results of the network in Experiment 5 with the parameters $f_{input} = 98\text{Hz}$, $f_{prior} = 440\text{Hz}$ and $\tau_{decay} = 4\text{ms}$ was also approximated in the above mentioned way and resulted in a value of 0.0094.

4.2.4 Discussion

When comparing Figures 4.12 and 4.13 it can be seen that the analysis output probabilities stayed the same. This is expected, because by doubling the pixels of the input images the information within it does not change as the areas corresponding to an output class double in size, as do the active pixels shown in black. However when comparing the Kullback-Leibler divergences of the network in Experiment 5 and the network of this experiment with parameters of $f_{input} = 98\text{Hz}$, $f_{prior} = 440\text{Hz}$ and $\tau_{decay} = 4\text{ms}$ it can be seen that the network with double size performed worse. The Kullback-Leibler divergence of the network with doubled size was 0.1411 compared to 0.0094 of the network the parameters were fitted to. This clearly indicates that the fitted parameters are not universal to every network size. This might be due to the fact that the membrane potentials of the output neurons are never normalized, which was already discussed in Experiment 5. However if such a normalization was implemented the fitted parameters of one network size could be universal to all other sizes.

4.3 Experiment 7: Training of the 1D network with predetermined parameters

4.3.1 Introduction

In this Experiment the network of Experiment 5 was used, but the weights of the network were not derived from the matrices $P^{X|Y}$ and $P^{Y|Z}$, but they

[4.3 Experiment 7: Training of the 1D network with predetermined parameters](#)

were learned from the input images.

4.3.2 Methods

The network architecture was the same as in Experiment 5 and the training paradigm was the same as in Experiment 4. The used network parameters were $f_{input} = 98Hz$, $f_{prior} = 440Hz$ and $\tau_{decay} = 4ms$. First the weight shifting parameter c was set to 1. Additionally to improve the results the values 2, 3 and 4 were also tried for c .

4.3.3 Results

The training results can be seen in Figure [4.15](#).

The evaluation results can be seen in Figure [4.16](#).

The Kullback-Leibler divergence for different values of c can be seen in Figure [4.17](#).

The training and evaluation results for the best value of $c = 3$ can be seen in Figures [4.18](#) and [4.19](#).

4.3.4 Discussion

Using the previously determined optimal network parameters for the weight training process was successful. When comparing A with B and C with D in Figure [4.15](#) it can be seen that the network learned the correct discrimination function. However the size the weights and the prior weights is too small. The Kullback-Leibler divergence also was worse with 0.0625 compared to 0.0101 from Experiment 5 Figure [4.12](#). To try to improve the similarity of the learned and the calculated probability matrices, and also to minimize the Kullback-Leibler divergence, different values for the weight shifting parameter c were tried. The lowest Kullback-Leibler divergence was achieved with $c = 3$. For this c the divergence was 0.0342. When comparing the probability matrices in Figure [4.18](#) it is apparent that the values of the

4 Experiments

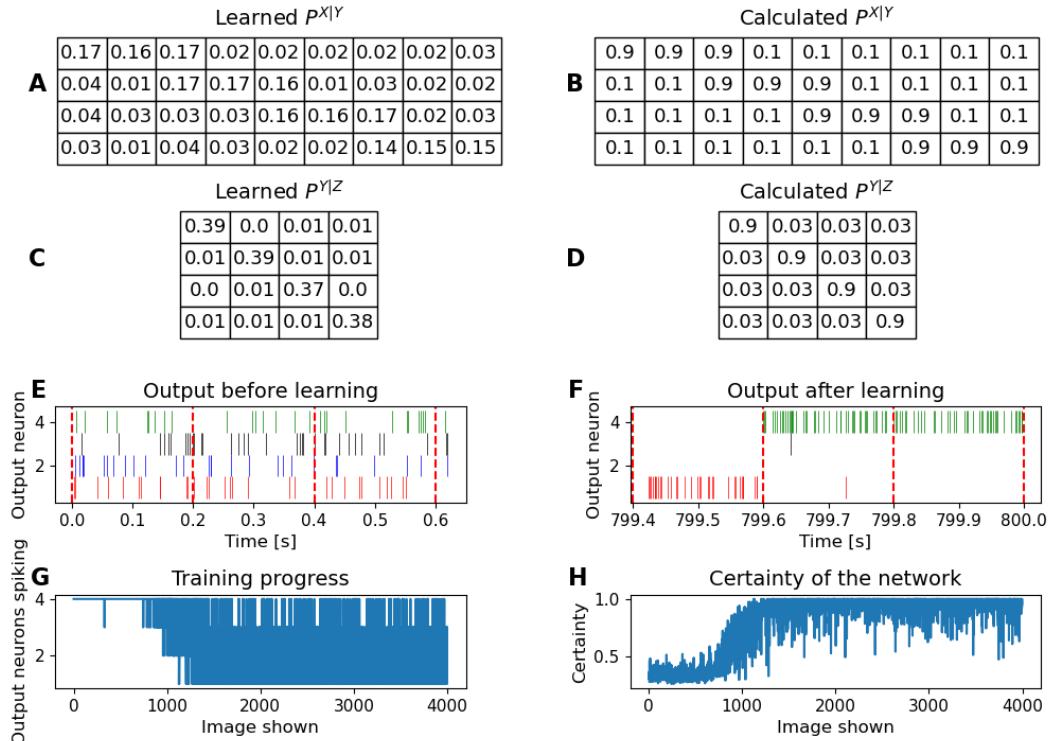


Figure 4.15: Training result $f_{input} = 98\text{Hz}$, $f_{prior} = 440\text{Hz}$, $\tau_{decay} = 4\text{ms}$, $c = 1$ **A** The learned probability matrix $P^{X|Y}$. It was determined by taking the weights to the power of e. **B** The calculated probability matrix $P^{X|Y}$. **C** The learned prior probability matrix $P^{Y|Z}$. **D** The calculated prior probability matrix $P^{Y|Z}$. **E, F** Spike activity expressed by the output neurons before and after the training of the network. **G** Number of distinct output neurons active during the presentation duration of each training image. **H** Proportion of most active output neuron to activity of all other output neurons during the presentation duration of each training image.

4.3 Experiment 7: Training of the 1D network with predetermined parameters

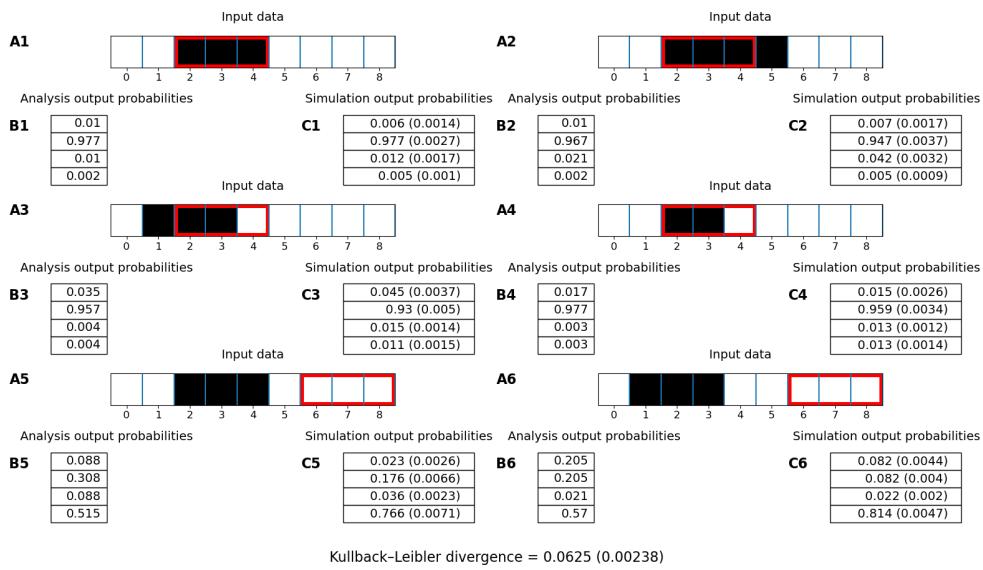


Figure 4.16: **Analysis and simulation result.** Parameters: $f_{input} = 98Hz$, $f_{prior} = 440Hz$, $\tau_{decay} = 4ms$ **A** Input images with 9×1 pixels. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

4 Experiments

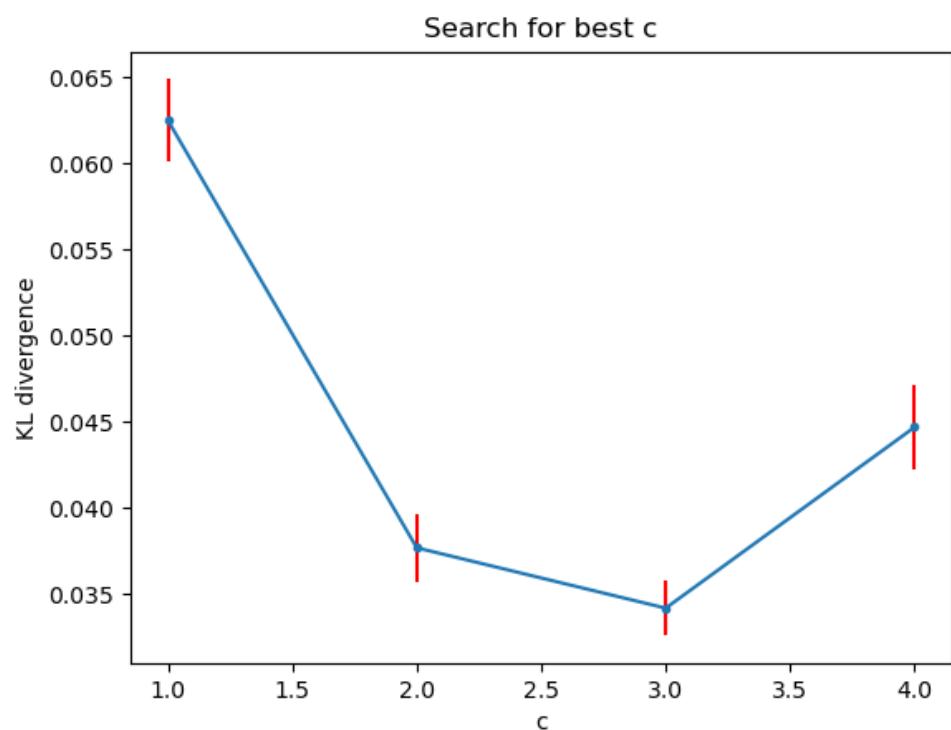


Figure 4.17: **KL divergence for different c values** $f_{input} = 98Hz$, $f_{prior} = 440Hz$, $\tau_{decay} = 4ms$

4.3 Experiment 7: Training of the 1D network with predetermined parameters

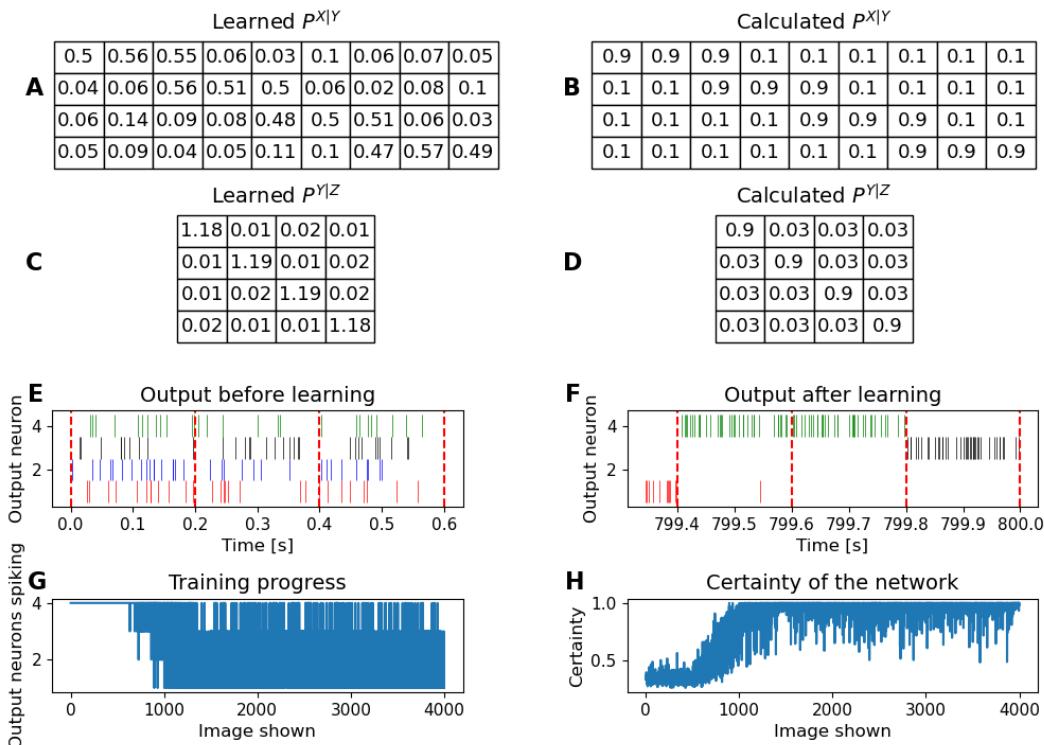


Figure 4.18: Training result $f_{input} = 98\text{Hz}$, $f_{prior} = 440\text{Hz}$, $\tau_{decay} = 4\text{ms}$, $c = 3$ **A** The learned probability matrix $P^{X|Y}$. It was determined by taking the weights to the power of e. **B** The calculated probability matrix $P^{X|Y}$. **C** The learned prior probability matrix $P^{Y|Z}$. **D** The calculated prior probability matrix $P^{Y|Z}$. **E, F** Spike activity expressed by the output neurons before and after the training of the network. **G** Number of distinct output neurons active during the presentation duration of each training image. **H** Proportion of most active output neuron to activity of all other output neurons during the presentation duration of each training image.

4 Experiments

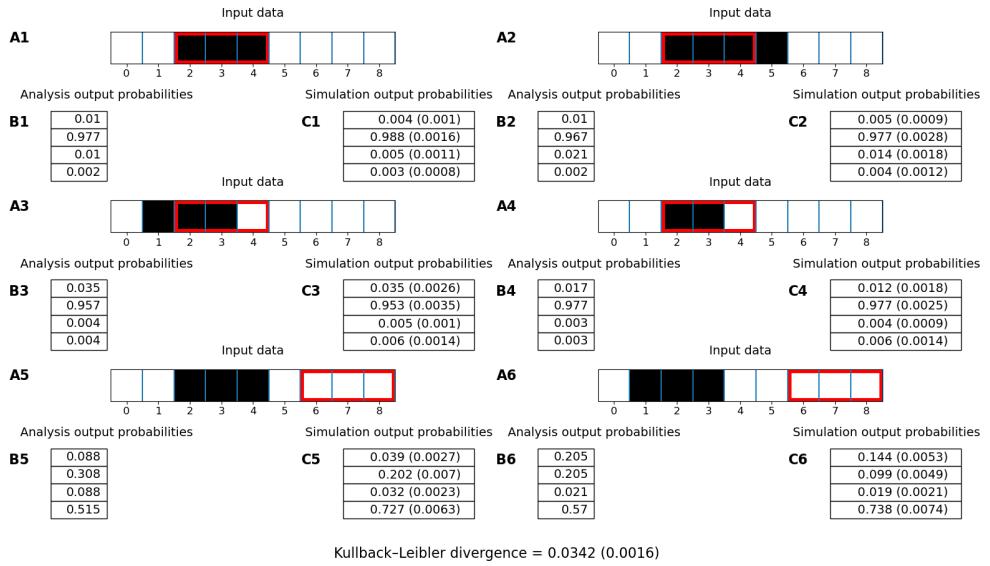


Figure 4.19: Analysis and simulation result. Parameters: $f_{input} = 98\text{Hz}$, $f_{prior} = 440\text{Hz}$, $\tau_{decay} = 4\text{ms}$, $c = 3$. **A** Input images with 9×1 pixels. **B** Analytically calculated posterior probabilities. **C** Proportions of the spikes of the output neurons during the simulation and their standard deviations in brackets.

learned $P^{X|Y}$ are still too small and they could profit from further increasing c . On the other hand the values in the learned $P^{Y|Z}$ are already too big and would profit from a smaller c . If one would want to further minimize the Kullback-Leibler divergence two separate weights shifting parameters for the input weights and the prior weights could be implemented.

4.4 Experiment 2: Horizontal and vertical bars

4.4.1 Introduction

For this experiment the impact of a neuron layer that encodes a-priori information should be analysed.

4.4.2 Methods

Input data 29×29 black and white images with either horizontal or vertical oriented bars on them were used, as it was more straightforward to express a-priori information. The orientation of the training images was chosen randomly via a uniform distribution. Also the positions of the bars in the images were uniformly distributed. The rest of the image generation process is analogous to experiment 1, except no circular mask was used. Examples of the input data can be seen in Figure 4.20. To show the value of the a-priori information validation images with two bars forming a cross were also generated, seen in Figure 4.21. When shown to the network in the validation process the prior neurons were given the information that a cross is either in horizontal or vertical orientation.

Network architecture This experiment used a expanded version of the network used in experiment 1. An additional layer of prior neurons z_1, z_2 was added. Whenever an image was oriented vertically z_1 was active and z_2 was inactive. For horizontal orientations z_2 was active and z_1 was inactive. Prior neurons in an active state had a firing frequency of 50 Hz and fired with 0 Hz when inactive.

4 Experiments

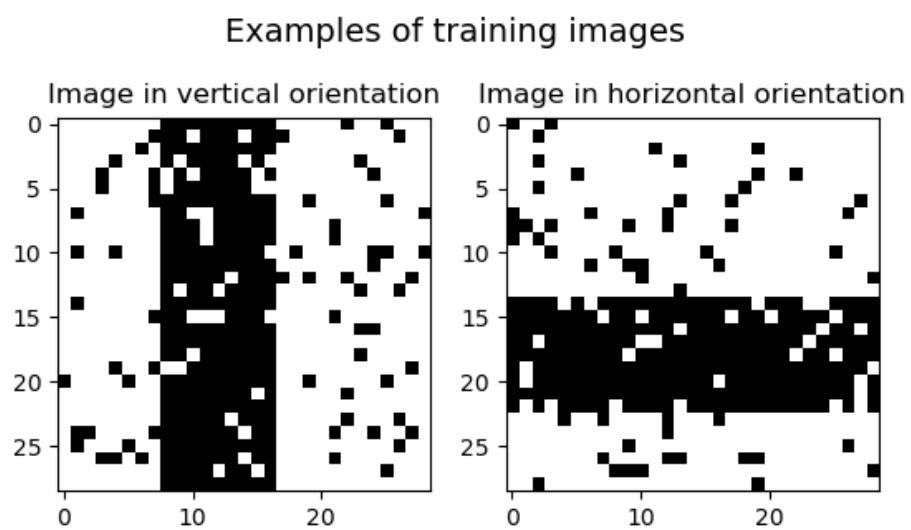


Figure 4.20: Training images generated for experiment 2. One image of each possible orientation at a random position.

4.4 Experiment 2: Horizontal and vertical bars

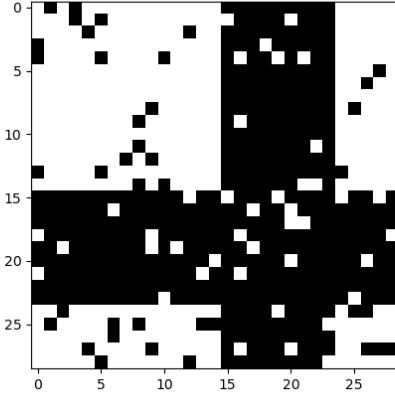


Figure 4.21: Generated cross image which can represent either horizontal or vertical orientation.

Neuron model The input and output neurons functioned the same way as in the previous experiment. The prior neurons fire according to a poisson process with their firing rate. Each prior neuron Z_l is connected to every output neuron Y_k and thus has weights w_{kl} that were learned by the network.

Parameters As the amount of input neurons and the average number of black pixels in an input image stayed roughly the same in this experiment, the same parameters $c = 20$ and $\lambda = 10^{-3}$ could be used.

However as there are only two prior neurons, a way to amplify their produced signals was needed, otherwise their impact on the membrane potential of the output neurons would not be distinctive enough. So the EPSPs $z_l(t)$ were multiplied by the factor Z_{factor} . This factor was determined via grid search. The additional prior layer resulted in an expanded version of the membrane potential $u_k(t)$

$$u_k(t) = \sum_{i=1}^n w_{ki} \cdot x_i(t) + \sum_{j=1}^n w_{kj} \cdot Z_{factor} \cdot z_j(t). \quad (4.15)$$

4 Experiments

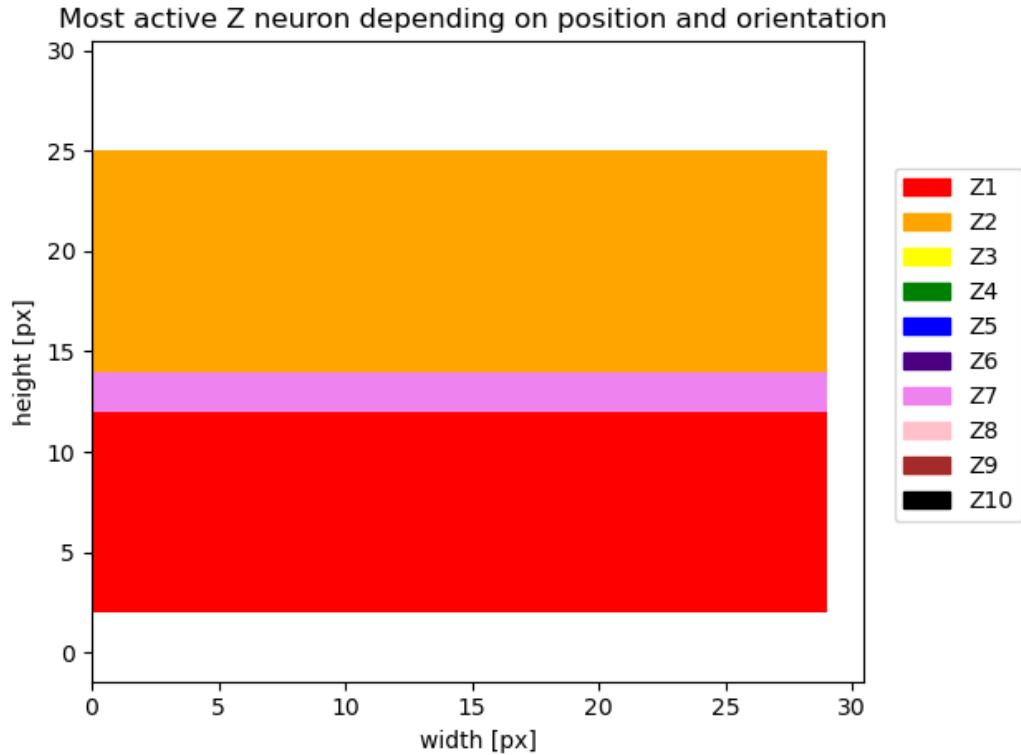


Figure 4.22: Most active output neuron for horizontal orientation and position on the y-axis of the training image during the training process. $c = 20$, $\lambda = 10^{-3}$, $Z_{factor} = 20$

4.4.3 Results

The following values for Z_{factor} were tried with $c = 20$ and $\lambda = 10^{-3}$:

- $Z_{factor} = 3$
- $Z_{factor} = 5$
- $Z_{factor} = 7$
- $Z_{factor} = 10$
- $Z_{factor} = 20$

For $Z_{factor} = 10$ and 20 the prior neurons impacted the learning progress negatively and let single prior neurons respond to too much area. An example of this can be seen in Figure 4.22

4.4 Experiment 2: Horizontal and vertical bars

The best results were achieved with $Z_{factor} = 5$. When looking at the training progress in Figure 4.23 it can be seen that the training accuracy is higher compared to experiment 1. This is due to the added a-priori information and the fact that less of each bar in an image is overlapping with multiple areas of output neurons. The network activity at the end of the training process can be seen in Figure 4.24. In Figures 4.25 and 4.26 the most active output neuron of the trained network is plotted for horizontal bars in every position and in the second plot for vertical bars in every position. Every one of the ten output neurons responds primarily to one coherent area in one orientation. The values of the learned prior weights w_{kl} were plotted in Figures 4.27 and 4.28. In these figures, in combination with Figures 4.25 and 4.26, can be seen that each prior neuron specialized on one orientation.

During the validation of all possible horizontal bar images the output spike activity was recorded and how many distinct output neurons were spiking during each image presentation period. For the parts where the 7 pixels high bar in the image did not overlap the areas of two output neurons only one output neuron was active. Only in the border areas there were two output neurons active. This can be seen in Figures 4.29 and 4.30. Compared to experiment 1 the activity is more homogeneous as each bar can only be in at most the area of four output neurons, when two of these areas are reinforced by the prior neurons.

To show the impact of the prior neurons an image with two bars on it forming a cross (Figure 4.31) was generated and z_2 was set active indicating that the orientation is supposed to be horizontal. This resulted in the spiking pattern seen in Figure 4.32. In it Y_1 is more active than Y_9 due to the influence of the prior neuron Z_2 .

4 Experiments

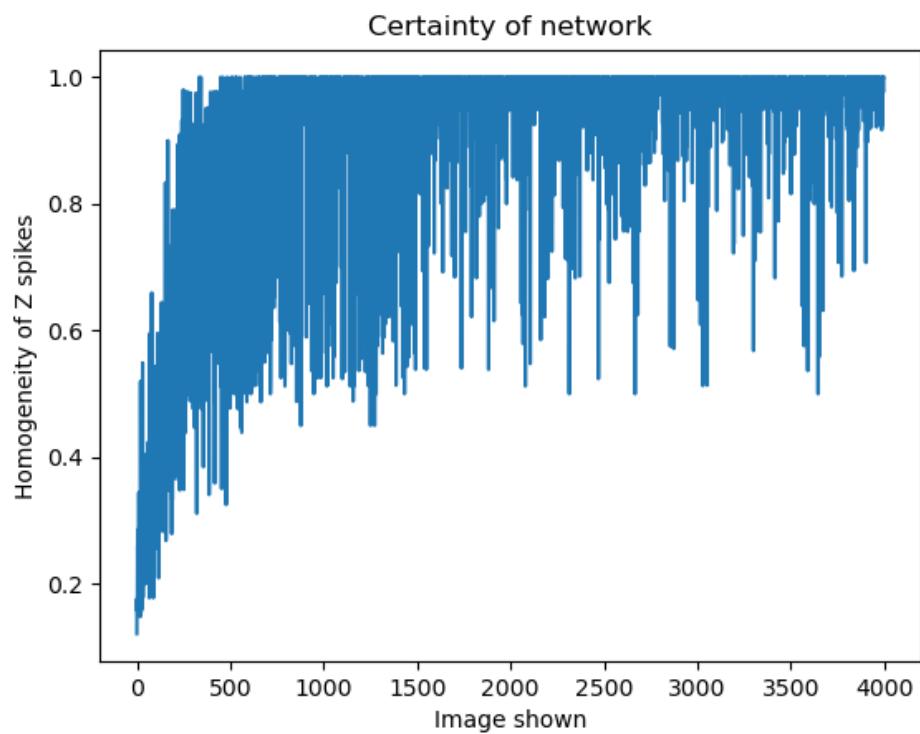


Figure 4.23: Proportion (accuracy) of most active output neuron to activity of all other output neurons during the presentation duration of each training image.
 $c = 20, \lambda = 10^{-3}, Z_{factor} = 5$

4.4 Experiment 2: Horizontal and vertical bars

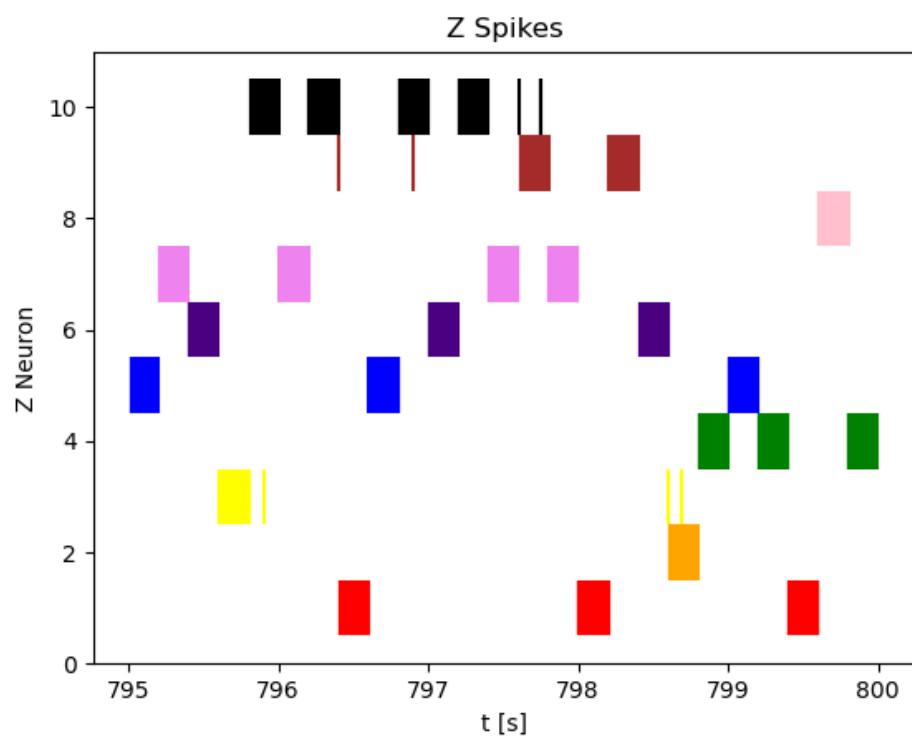


Figure 4.24: Last 1000 output neuron spikes, $c = 20$, $\lambda = 10^{-3}$, $Z_{factor} = 5$

4 Experiments

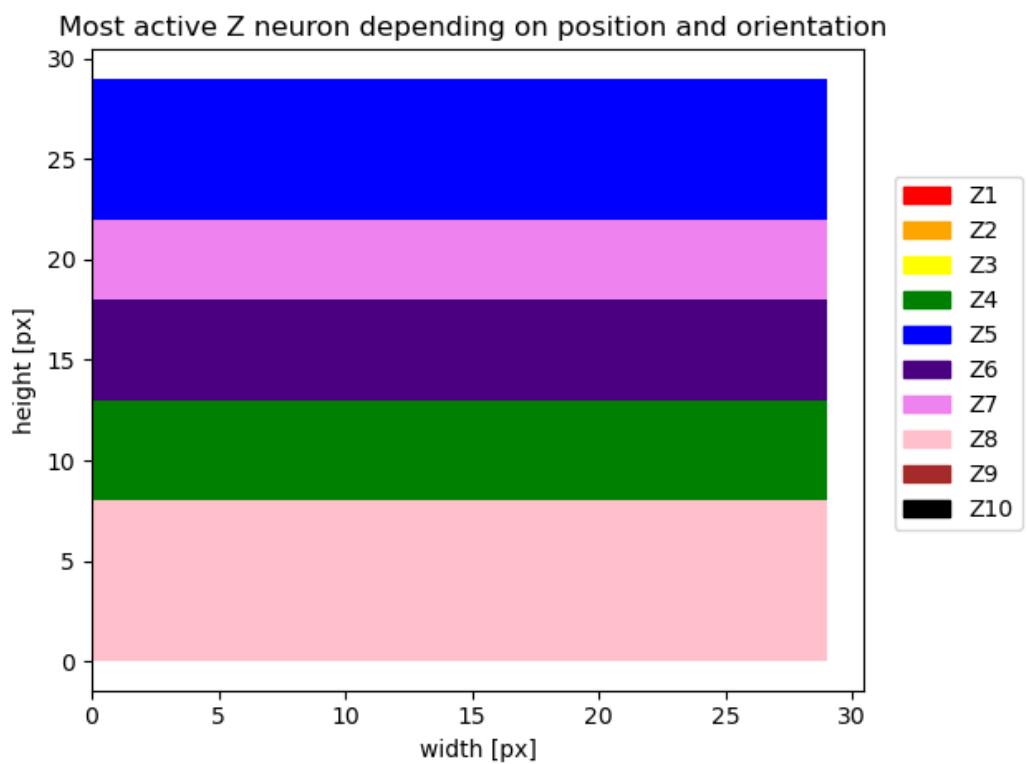


Figure 4.25: Most active output neuron for images of horizontal orientation and position on the x-axis of during the validation process. $c = 20$, $\lambda = 10^{-3}$, $Z_{factor} = 5$

4.4 Experiment 2: Horizontal and vertical bars

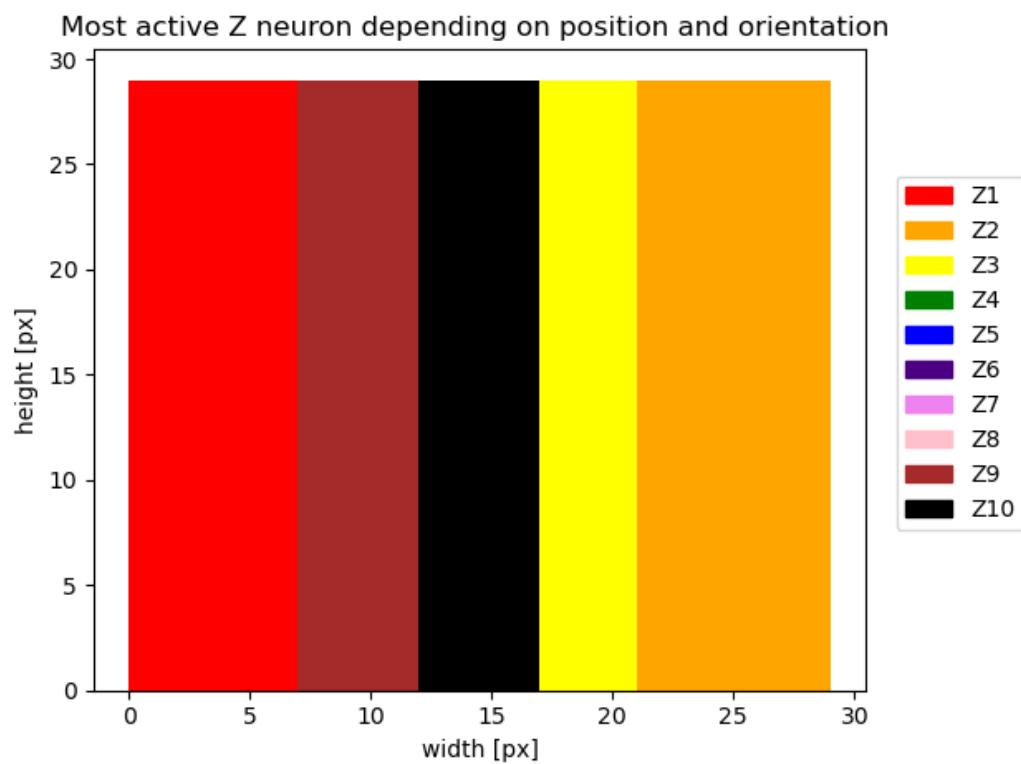


Figure 4.26: Most active output neuron for images of vertical orientation and position on the x-axis of during the validation process. $c = 20, \lambda = 10^{-3}, Z_{factor} = 5$

4 Experiments

Weights of A1 to all Z

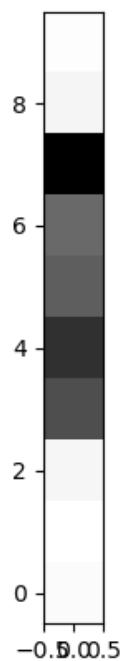


Figure 4.27: Values of w_k , darker color means higher value. $c = 20, \lambda = 10^{-3}, Z_{factor} = 5$

4.4 Experiment 2: Horizontal and vertical bars

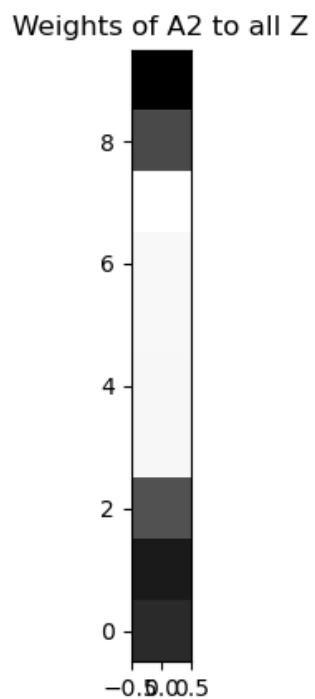


Figure 4.28: Values of w_{k2} , darker color means higher value. $c = 20, \lambda = 10^{-3}, Z_{factor} = 5$

4 Experiments

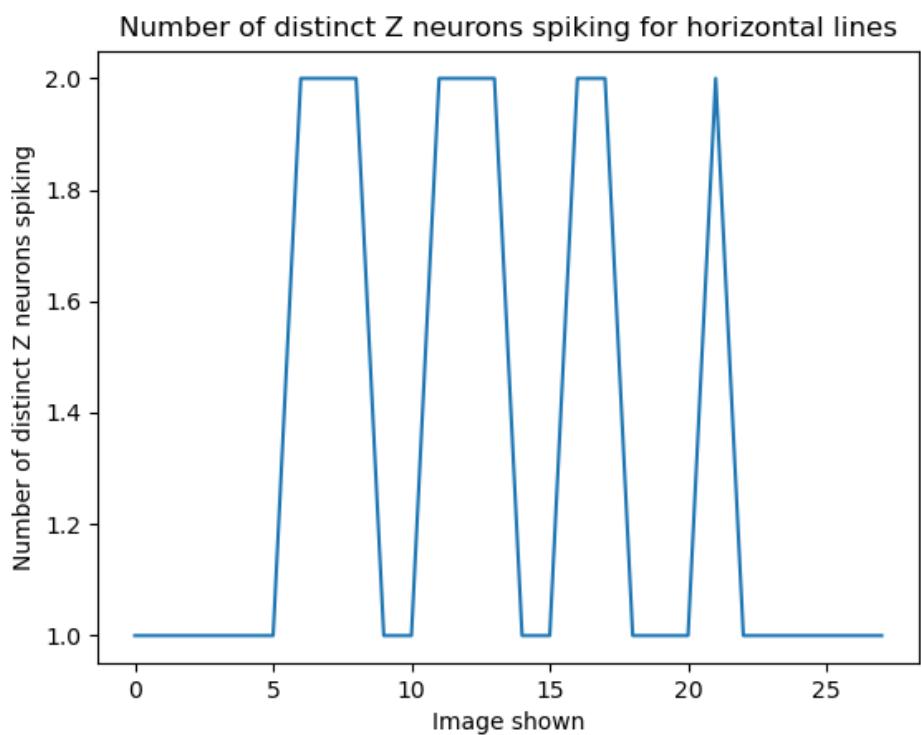


Figure 4.29: Distinct output neurons spiking during the presentation of all possible horizontal oriented validation images, $c = 20, \lambda = 10^{-3}, Z_{factor} = 5$

4.4 Experiment 2: Horizontal and vertical bars

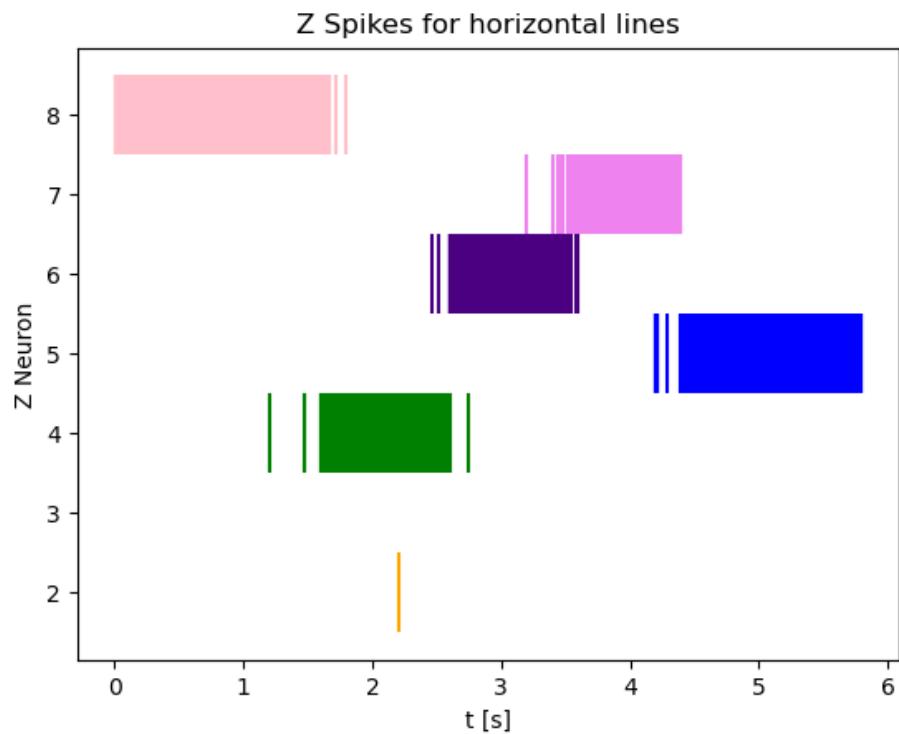


Figure 4.30: Output neuron spikes during the presentation of all possible horizontal oriented validation images, $c = 20, \lambda = 10^{-3}, Z_{factor} = 5$

4 Experiments

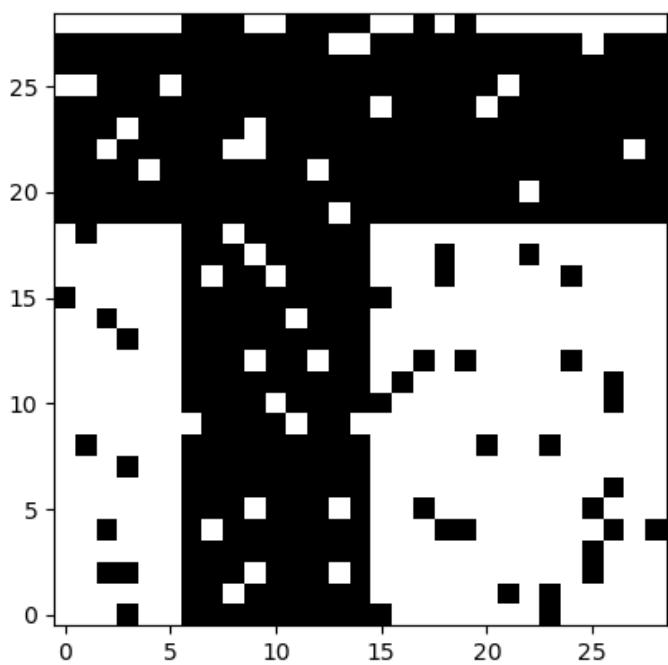


Figure 4.31: Generated validation cross image, with orientation defined as horizontal.
 $c = 20, \lambda = 10^{-3}, Z_{factor} = 5$

4.4 Experiment 2: Horizontal and vertical bars

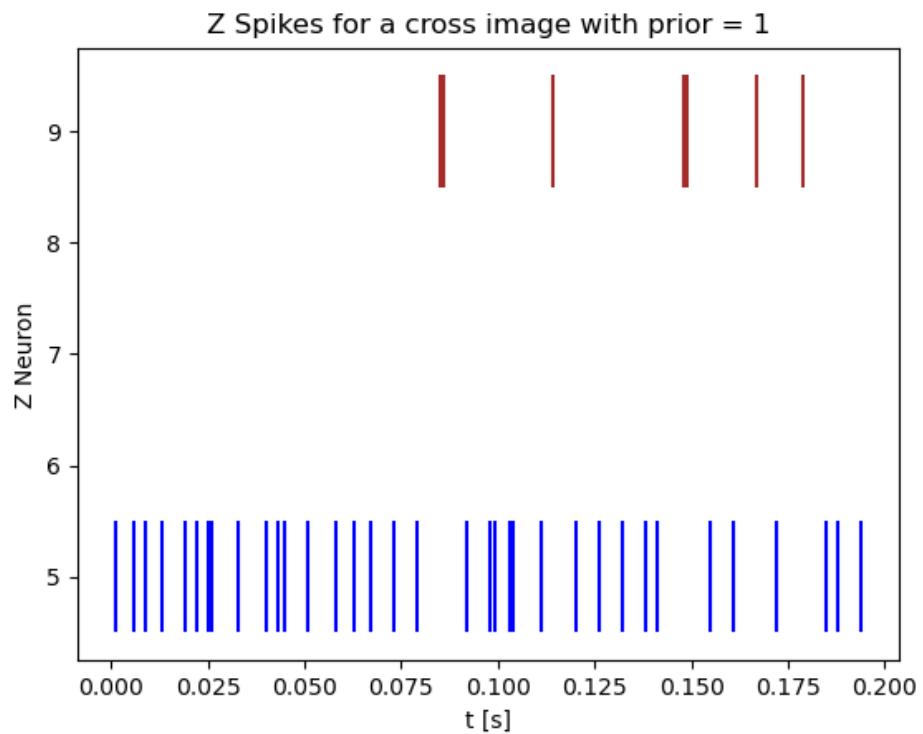


Figure 4.32: Output spikes during the presentation of the validation cross image. $c = 20, \lambda = 10^{-3}, Z_{factor} = 5$

4.5 Experiment 4: Horizontal and vertical bars with adaptive inhibition

4.5.1 Introduction

This experiment applies the adaptive inhibition already analysed in Experiment 4.7 to Experiment 4.4. In Experiment 4.4 the impact of the prior neurons was too small to influence the spiking activity of the output neurons. The newly implemented adaptive inhibition simplifies the process of changing the prior neurons, as it keeps the output firing frequency constant, thus preventing the need to perform a new parameter search upon each change to the prior neurons.

4.5.2 Methods

As in Experiment 4.7 the firing rate of the output neurons was set to 200 Hz. To increase the influence of the prior neurons the firing rate of the prior neurons was increased from 50 Hz to 200 Hz. This change alone did not increase the impact of the prior neurons enough to help the network correctly detect a "cross-bar image" as either horizontal or vertical. It was also tried to increase the parameter c_{prior} for the learning of the prior weights to shift the weight values towards bigger values. This did not increase the values of the weights, they stayed at a maximum of four. Thus the number of prior neurons had to be increased. Via grid search different numbers of prior neurons were tried.

4.5.3 Results

The tried parameters were as follows:

- $c = c_{prior} = 20$
- $\lambda = 10^{-3}$
- number of prior neurons = 10, 20, 50, 100, 200

4.5 Experiment 4: Horizontal and vertical bars with adaptive inhibition

For 50, 100 and 200 prior neurons the training process was impaired by the activity of the prior neurons. This arose as some output neurons responding to too large areas, while other prior neurons not responding to any specific areas. For 50 prior neurons four output neurons responded to horizontal bars, while six output neurons responded to vertical bars. This was unexpected and might be due to the stochastic nature of the training image generation. This was not yet analysed further and the validation of the network was performed with 20 prior neurons, as it was the largest amount of prior neurons that resulted in a properly trained network. The results of the training process can be seen in Figures 4.33 and 4.34.

Next the impact of the prior was checked. To do this an image with one horizontal (at pixel 12) and one vertical bar (at pixel 5) on it was generated. Then the prior was set once to 0 (vertical) and once to 1 (horizontal). The image and the prior was then fed to the network. The cross image can be seen in Figure 4.35. The spiking activity depending on the prior can be seen in Figures 4.36 and 4.37.

To better show the impact of the prior it was gradually changed from horizontal to vertical. The starting firing frequency of the horizontal prior neurons was 200 Hz and 0 Hz for the vertical prior neurons. A cross image was shown to the network for 200 ms. After each image presentation duration the firing frequency of the horizontal prior neurons was decreased by 1 Hz and increased by 1 Hz for the vertical prior neurons. The used cross image can be seen in Figure 4.38. The firing frequency of the 2 most active output neurons depending on the firing frequency of the vertical prior neurons can be seen in Figure 4.39. As expected the correct horizontal output neuron is active in the beginning with about 200 Hz and the correct vertical output neuron is almost inactive. With rising firing frequency of the vertical prior neurons the activity of the horizontal output neurons decreases and the activity of the vertical output neurons increases. The membrane potentials of all output neurons for vertical prior neurons spiking with 200 Hz and horizontal prior neurons spiking with 0 Hz can be seen in Figure 4.40.

4 Experiments

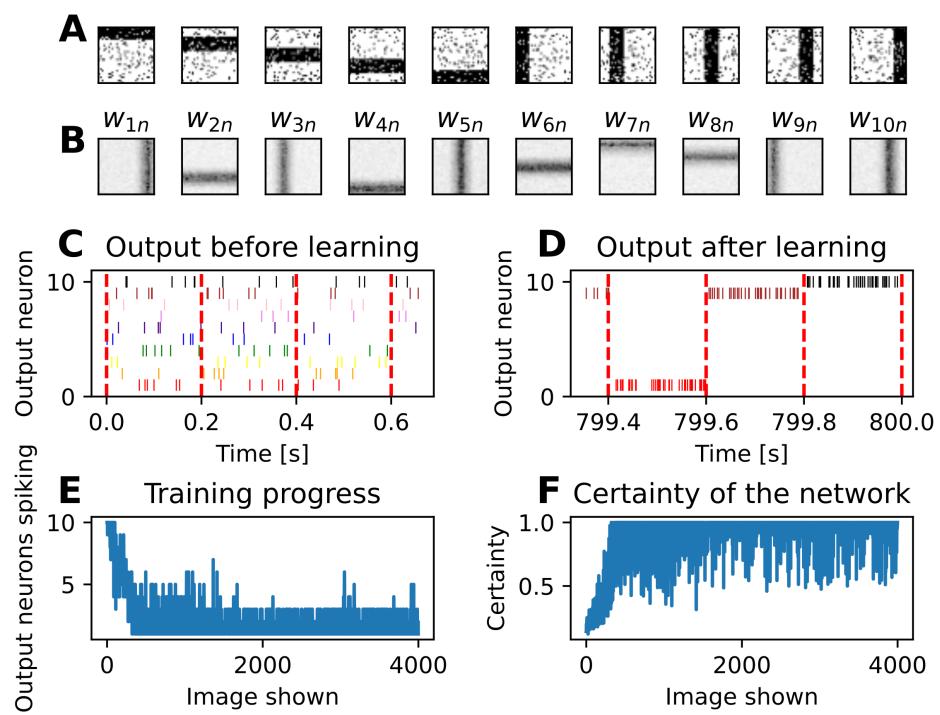


Figure 4.33: **Training with 20 prior neurons.** **A** Examples of 35×35 -pixel input images of horizontal and vertical bars with background noise. **B** Learned weights of the connections between input and output neurons. **C, D** Spike activity expressed by the output neurons before and after the training of the network. **E** Number of distinct output neurons active during the presentation duration of each training image. **F** Proportion of most active output neuron to activity of all other output neurons during the presentation duration of each training image.

4.6 Experiment 1: Rotated Bars

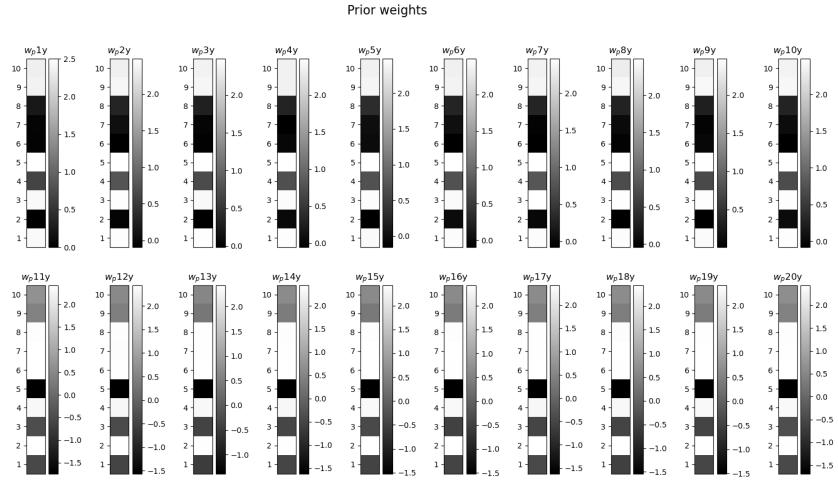


Figure 4.34: Learned weights of the connections between prior and output neurons.

4.6 Experiment 1: Rotated Bars

4.6.1 Introduction

The goal of this task was to recreate example 2 from Nessler et al. (2013). In this example they fed a winner-take-all spiking neural network images with bars in different orientations on them. The network then clustered the images into ten groups depending on their orientation.

4.6.2 Methods

Input data The images used in this task were generated with a size of 29×29 pixels. Black bars with a width of 7 pixels going through the center of the image were drawn onto a white background. To simulate noise each pixel had a chance of ten percent to have its color flipped. To ensure that all bars in the images have the same length regardless of their orientation a circular mask with a radius of 15 pixels was applied to the images. This recolored all pixels outside of the mask to white. During the training of the network one

4 Experiments

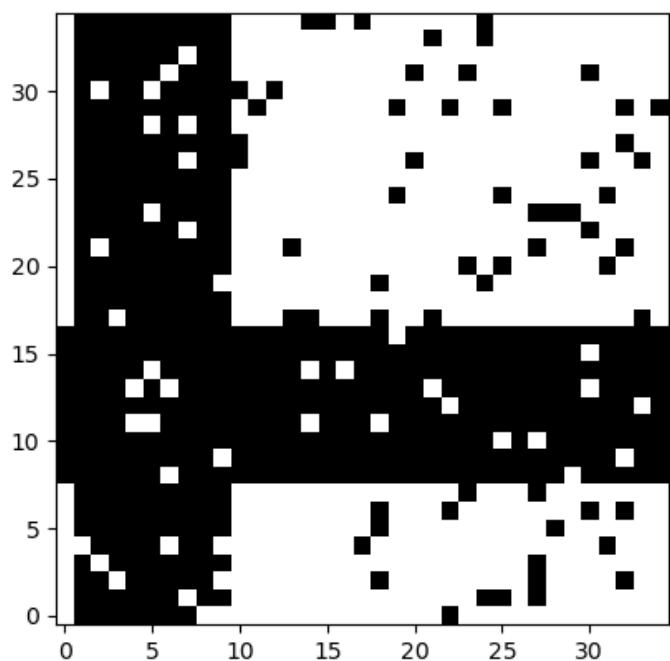


Figure 4.35: Cross image fed to the network.

4.6 Experiment 1: Rotated Bars

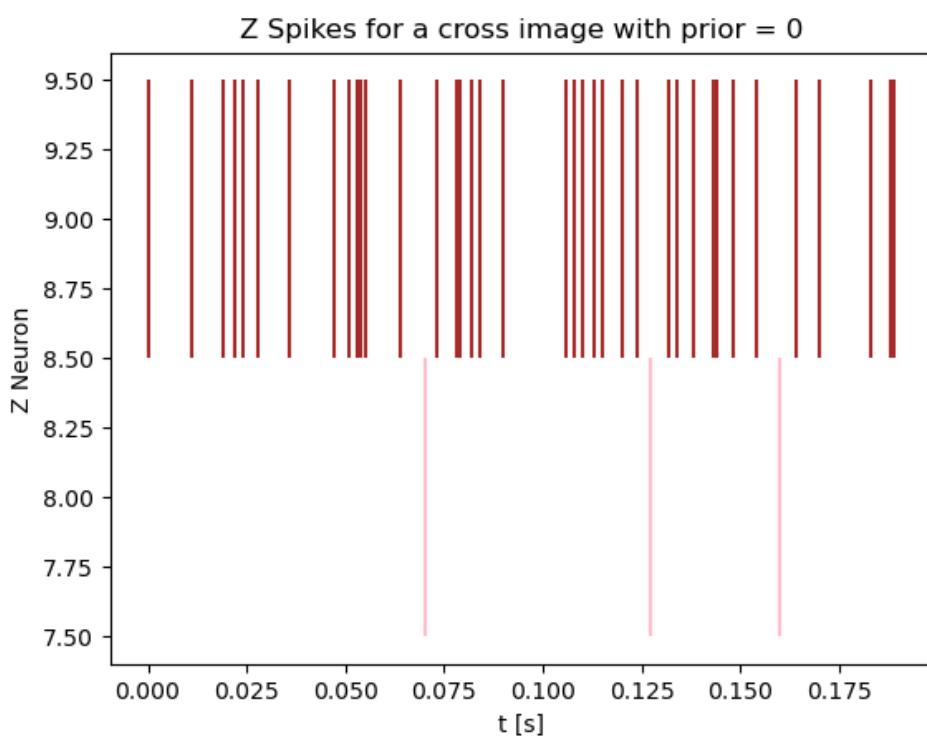


Figure 4.36: Spiking activity of the output neurons with prior = 0.

4 Experiments

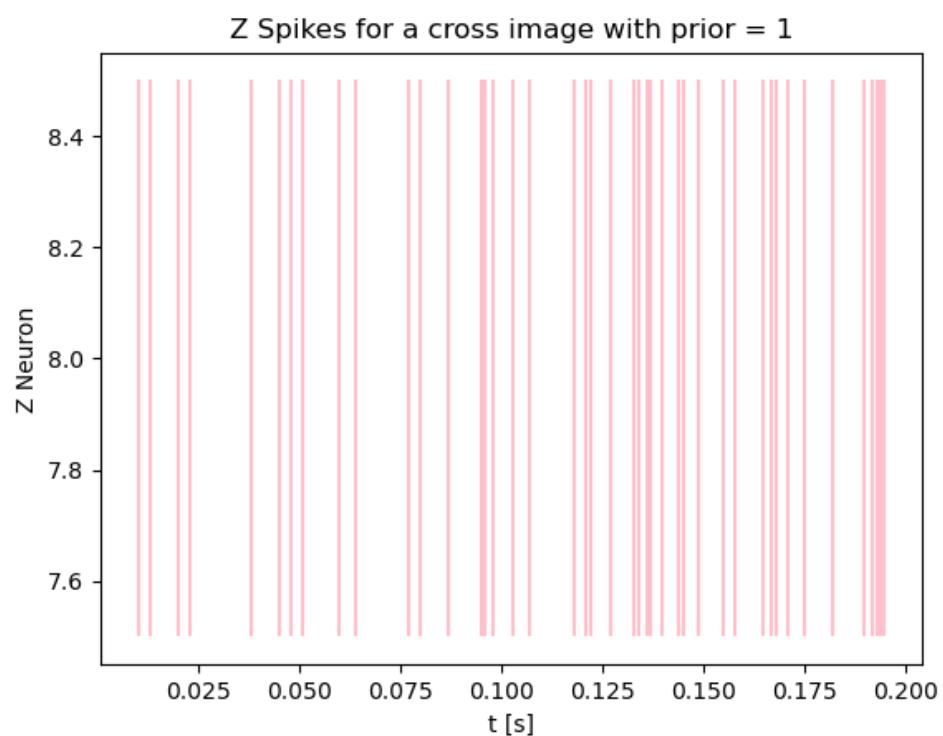


Figure 4.37: Spiking activity of the output neurons with prior = 1.

4.6 Experiment 1: Rotated Bars

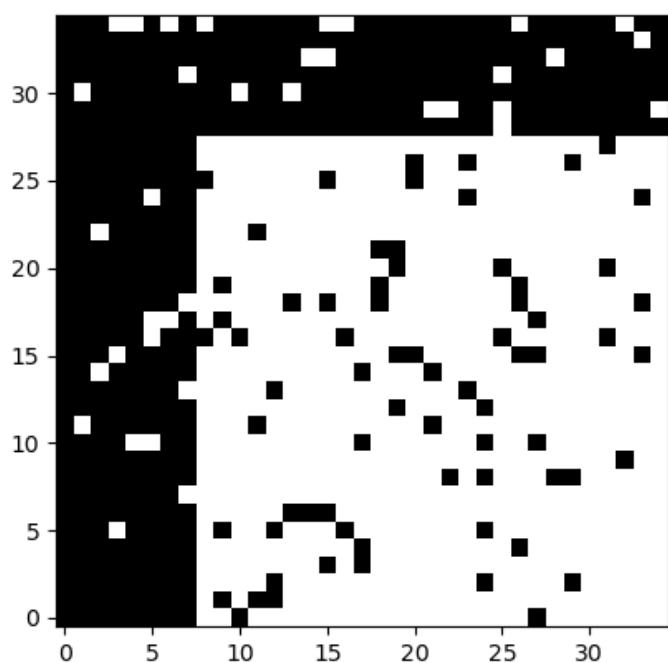


Figure 4.38: Cross image fed to the network.

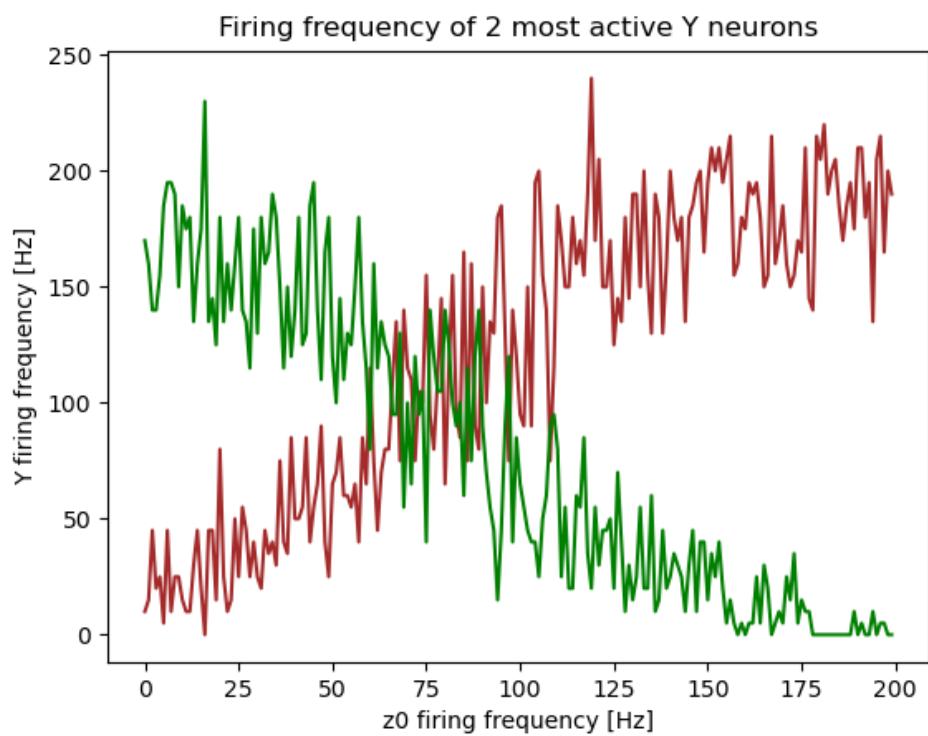


Figure 4.39: Firing frequency of the 2 most active output neurons depending on prior firing frequency.

4.6 Experiment 1: Rotated Bars

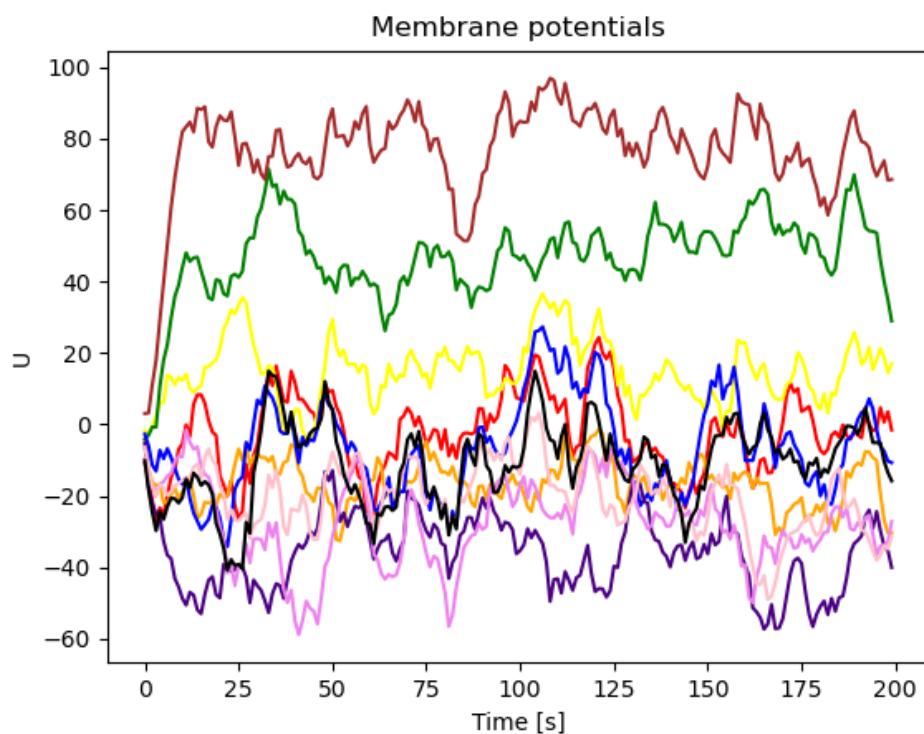


Figure 4.40: Membrane potentials of output neurons for vertical prior neurons spiking with 200 Hz and horizontal prior neurons spiking with 0 Hz.

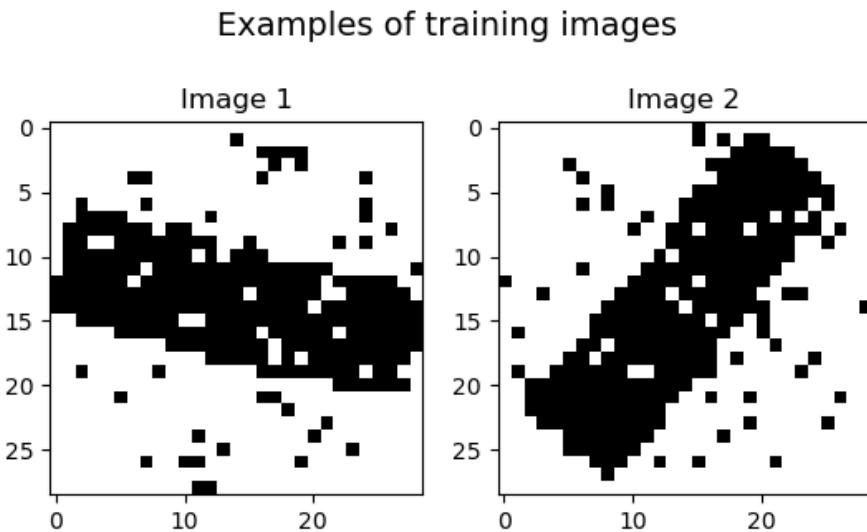


Figure 4.41: Two examples of the generated training data in this experiment.

image per iteration was generated in a uniformly distributed orientation and each image was shown for 200 ms. For the training of the network 4000 of these images were shown. The randomly chosen orientation could lie between 0 and 359 degrees. Two examples of such images can be seen in Figure 4.41.

Neuron model !!!!!!!! A lot of the following was copied to "Theoretical Background" and rewritten (without specific numbers)

As in Nessler et al. (2013) the input neurons X are firing according to a poisson process with an average firing rate of 20 Hz when active and with 0 Hz when in an inactive state. The excitatory post synaptic potentials (EPSPs) $x_i(t)$ that these neurons produce can be seen in Figure 4.42. A double exponential kernel was used to generate the EPSP. The time constant for the rise of the signal τ_{rise} was 1 ms and the time constant for the decay of the signal τ_{decay} was 15 ms. The addition of the time step size δt was necessary to get the time t at the end of the current simulation step. t_f is

4.6 Experiment 1: Rotated Bars

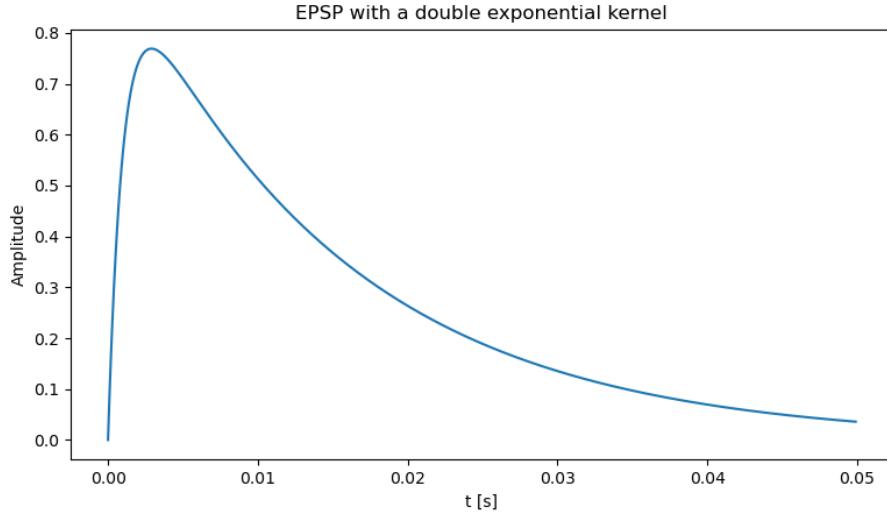


Figure 4.42: Form of an excitatory post synaptic potential generated by an input neuron over time. A double exponential kernel was used to generate this signal. These signals are fed to the next layer of the network.

the time at which the spike of x_i occurred

$$x_i(t) = e^{-(t+\delta t-t_f)/\tau_{decay}} - e^{-(t+\delta t-t_f)/\tau_{rise}}. \quad (4.16)$$

The firing rate of an output neuron y_k is given by

$$r_k(t) = e^{u_k(t)-I(t)}. \quad (4.17)$$

The probability of an individual output neuron to fire within a time step δt is given by

$$r_k(t) \cdot \delta t. \quad (4.18)$$

Network architecture Each pixel of an input image was connected to two neurons. The first of these neurons is in an active state when the pixel is black and in an inactive state otherwise. The second neuron expresses the opposite behaviour. As a consequence the network needs 1682 ($29 \cdot 29 \cdot 2$) excitatory input neurons x_1, \dots, x_n . These input neurons are fully connected

4 Experiments

to ten excitatory output neurons y_1, \dots, y_k . This means that every input neuron x_i is connected to each output neuron y_k . The membrane potential u_k of each output neuron is calculated by multiplying the EPSP of each input neuron times the weight of the connection between them

$$u_k(t) = \sum_{i=1}^n w_{ki} \cdot x_i(t). \quad (4.19)$$

In Nessler et al. (2013) each output neuron y_k also had an intrinsic excitability w_{k0} which was learned for each neuron. For this experiment however it was omitted, as each orientation of input images was equally likely, thus the intrinsic excitability of each output neuron would end up being the same.

The output neurons are modelled in a winner-takes-all (WTA) circuit. This means that whenever one output neuron spikes, a lateral inhibitory signal is fed to all output neurons, thus preventing the further activation of them for $\sigma_{inh} = 5ms$. After completion of the training of the network each output neuron should be active for bars in a coherent area. Each of those areas should ideally be of equal size. As the bars can be oriented between 0 and 359 degrees, but each 180 degree rotation of an image results in an equivalent image, the desired size of the active area of an output neuron should be 18 degrees.

Inhibition The inhibition signal was chosen to depend on the current membrane potential of the output neurons. According to Nessler et al. (2013) the total firing rate of the output neurons is

$$R(t) = \sum_{k=1}^K e^{u_k(t) - I(t)}. \quad (4.20)$$

Solving this equation for $I(t)$ yields

$$R(t) = \frac{\sum_{k=1}^K e^{u_k(t)}}{e^{I(t)}} \quad (4.21)$$

$$e^{I(t)} = \frac{\sum_{k=1}^K e^{u_k(t)}}{R(t)} \quad (4.22)$$

4.6 Experiment 1: Rotated Bars

$$I(t) = \ln \frac{\sum_{k=1}^K e^{u_k(t)}}{R(t)} \quad (4.23)$$

$$I(t) = -\ln R(t) + \ln \sum_{k=1}^K e^{u_k(t)}. \quad (4.24)$$

When implementing the inhibition the $-\ln R(t)$ term of Equation 4.24 was overlooked, that means it was assumed to be zero. Because of that $R(t)$ equals 1 when the inhibition is active. This error was not detected at first, as the chance that a Y neuron fires within a time step of 1 ms with active inhibition is 1/1000 due to that oversight.

Whenever an output neuron produces a spike the inhibition signal $I(t)$ is subtracted from the membrane potential $u_k(t)$ of every output neuron. This happens for the duration of $\sigma_{inh} = 5ms$. Thus follows

$$I(t) = \begin{cases} \ln(\sum_{i=1}^k e^{u_i}) & \text{if any } y_k \text{ fired in } [t^f, t^f + \sigma_{inh}] \\ 0 & \text{if any } y_k \text{ did not fire in } [t^f, t^f + \sigma_{inh}]. \end{cases} \quad (4.25)$$

Spike timing dependent plasticity !!!!! STDP WAS ALREADY EXPLAINED IN THEORETICAL BACKGROUND. THUS ENSURE THAT NOTHING HERE IS EXPLAINED AGAIN!!!! The weights w_{ki} between neurons x_i and y_k are updated whenever an output neuron fires. The time window σ was set to 10 ms according to Nessler et al. (2013). If y_k produces a spike all its weights are updated as

$$\Delta w_{ki} = \begin{cases} \lambda \cdot (ce^{-w_{ki}} - 1) & \text{if } x_i \text{ fired in } [t^f - \sigma, t^f] \\ \lambda \cdot (-1) & \text{if } x_i \text{ did not fire in } [t^f - \sigma, t^f], \end{cases} \quad (4.26)$$

where λ is the learning rate, the parameter c shifts the weight values, t^f is the time when y_k spiked and σ is the time window in which input spikes are considered as "before" an output spike. As the membrane potentials u_k of the output neurons result from the addition of the EPSPs of the 1682 input neurons times the corresponding weight, a way to control the average size of u is needed. If u is too small the output neurons will fire too sparsely

and if u is too big it will impair the learning process. So to limit u , the size of the weights is controlled via the parameter c . The learning rate λ is needed to control the size of each weight update. If it is too big few output neurons will take over more than the expected 18° areas and others will only respond for smaller areas or not at all. On the other hand if λ is too small the network will learn very slowly and may never converge. Due to these two parameters being unwieldy to determine analytically they were chosen via grid search.

4.6.3 Results

Parameter search The two parameters c , which controls the size of the weights, and the learning rate λ were fitted to the network via grid search. The tested parameters were as follows:

- $c = 1 (\lambda = 10^{-2}, 10^{-3}, 10^{-4})$
- $c = 10 (\lambda = 10^{-2}, 10^{-3}, 10^{-4})$
- $c = 20 (\lambda = 10^{-2}, 10^{-2.5}, 10^{-3}, 10^{-3.5}, 10^{-4}, 10^{-4.5}, 10^{-5})$
- $c = 30 (\lambda = 10^{-2}, 10^{-2.5}, 10^{-3}, 10^{-3.5})$

The simulation was conducted by simulating small discrete time steps and calculating the changes of the network in each step. The step size δt was chosen as 1 ms.

$c = 1$ The best results for $c = 1$ were achieved with $\lambda = 10^{-2}$. But overall this value for c did not work, even though the network did learn to cluster images into eight groups depending on their orientation. In Figure 4.43 one can see which output neuron was the most active during the training process for each angle in 1° steps.

In Figure 4.44 the training progress of the network can be seen. The figure shows the number of distinct output neurons active during the presentation of each training image shown. Due to the large learning rate, compared to other values of c , the network learned quickly. However after iteration 70 there were images shown for which not a single output neuron spiked. This

4.6 Experiment 1: Rotated Bars

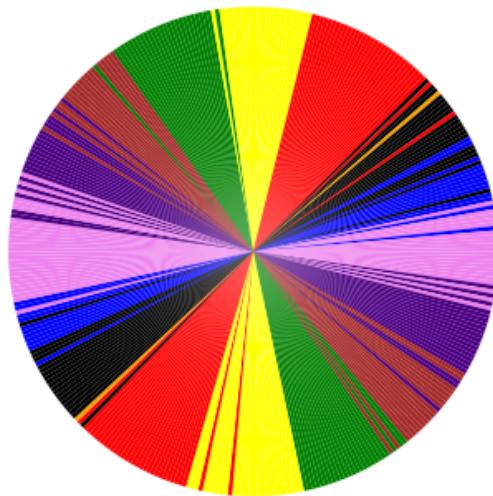


Figure 4.43: Most active output neuron depending on orientation of the training image during the training process. $c = 1, \lambda = 10^{-2}$

4 Experiments

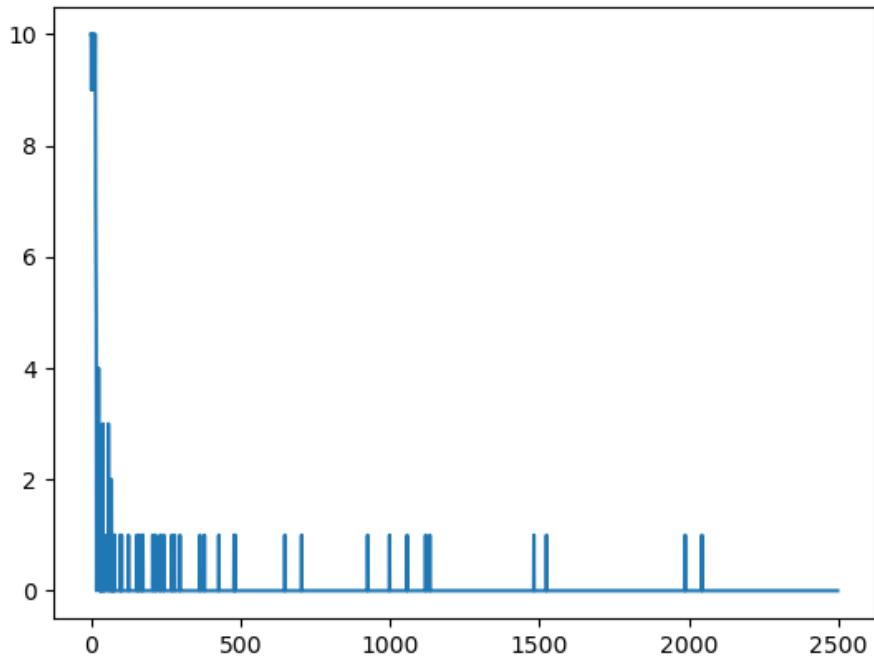


Figure 4.44: Number of distinct output neurons active during the presentation duration of each training image. x-axis: image shown, y-axis: distinct output neurons active, $c = 1, \lambda = 10^{-2}$

dying out of the networks activity is due to the parameter c being too small, which leads to too low membrane potentials.

$c = 10$ This value of $c = 10$ had the same problem of the network activity dying out as $c = 1$ although at a later point in time, as can be seen in Figure 4.45. Also in Figure 4.46 the last 50 output spikes of the training process can be seen. As indicated by Figure 4.45 the activity is sparse.

$c = 20$ $c = 20$ was the first value for which the network activity did not die out after some time. For $\lambda = 10^{-2}$ one output neuron learned to spike

4.6 Experiment 1: Rotated Bars

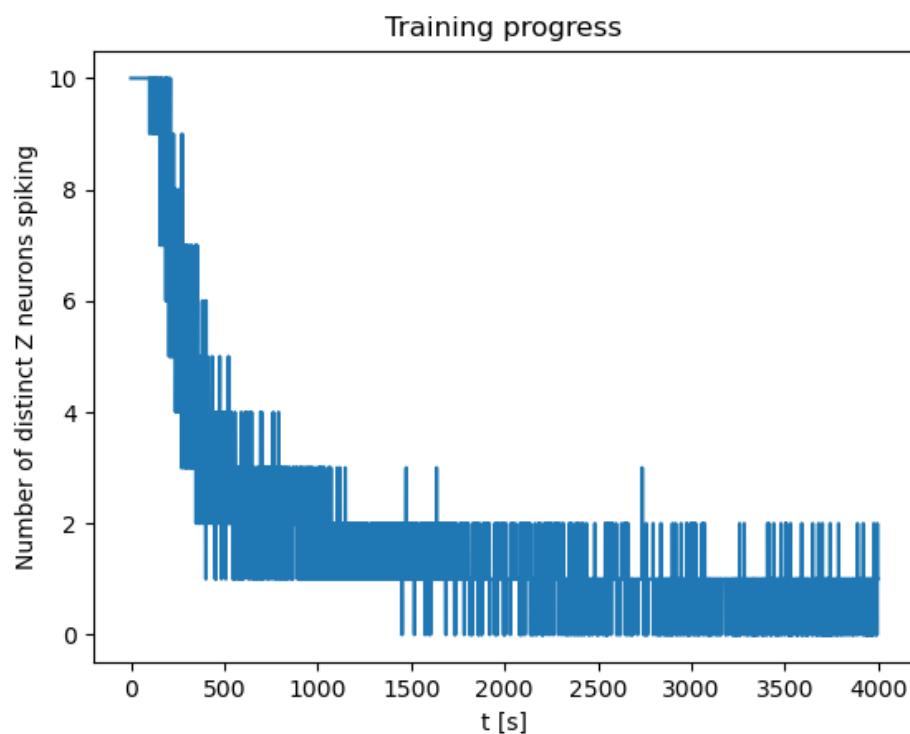


Figure 4.45: Number of distinct output neurons active during the presentation duration of each training image. x-axis: image shown, y-axis: distinct output neurons active, $c = 10, \lambda = 10^{-3}$

4 Experiments

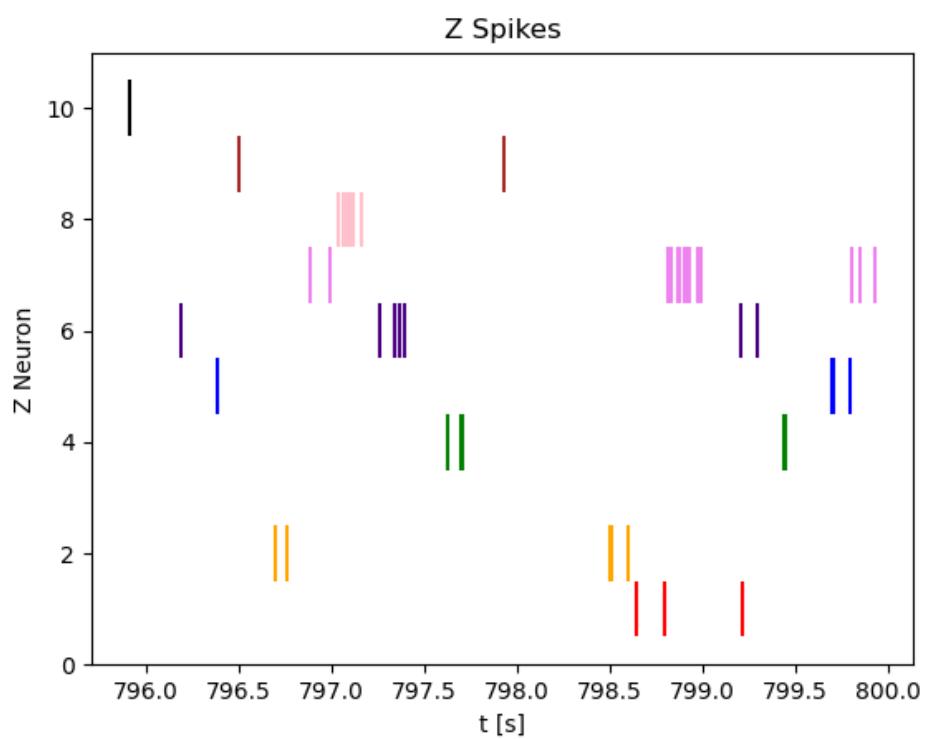


Figure 4.46: Last 50 output neuron spikes, $c = 10, \lambda = 10^{-3}$

4.6 Experiment 1: Rotated Bars

Most active Z neuron depending on angle

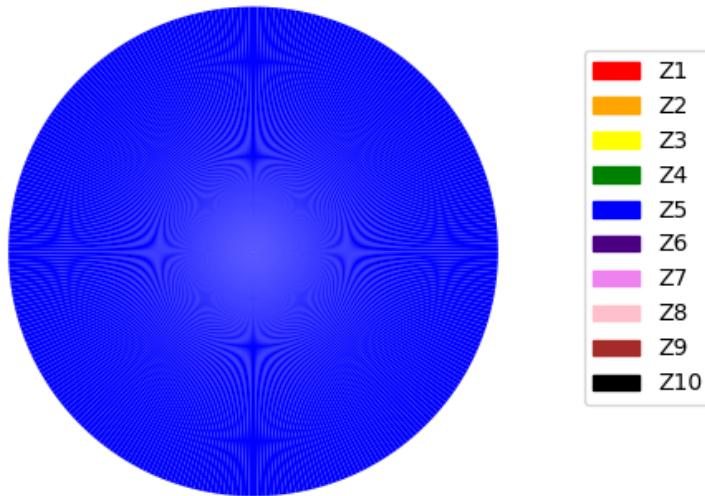


Figure 4.47: Most active output neuron depending on orientation of the training image during the training process. $c = 20, \lambda = 10^{-2}$

first for every possible input image orientation, see Figure 4.47.

With $c = 20$ and $\lambda = 10^{-3}$ the first combination that yielded a stable network was found. In Figure 4.48 the amount of distinct output neurons firing during the presentation of each training image can be seen. Figure 4.49 shows the proportion of the most active output neuron to all other output neurons active for each training image. Both of these figures can be used to measure the training progress of the network. Figure 4.49 however shows the additional information how certain the network is that a training image belongs to a specific group.

As this network did not die out after some time the trained network was analysed further. 180 images were generated in 1° steps and each was shown

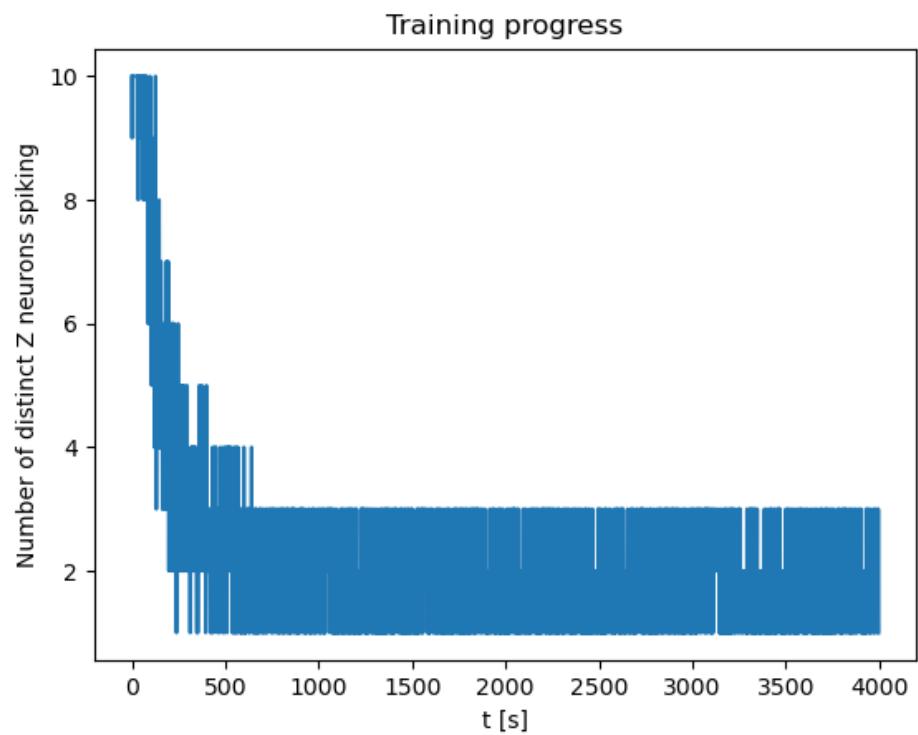


Figure 4.48: Number of distinct output neurons active during the presentation duration of each training image. x-axis: image shown, y-axis: distinct output neuron active, $c = 20, \lambda = 10^{-3}$

4.6 Experiment 1: Rotated Bars

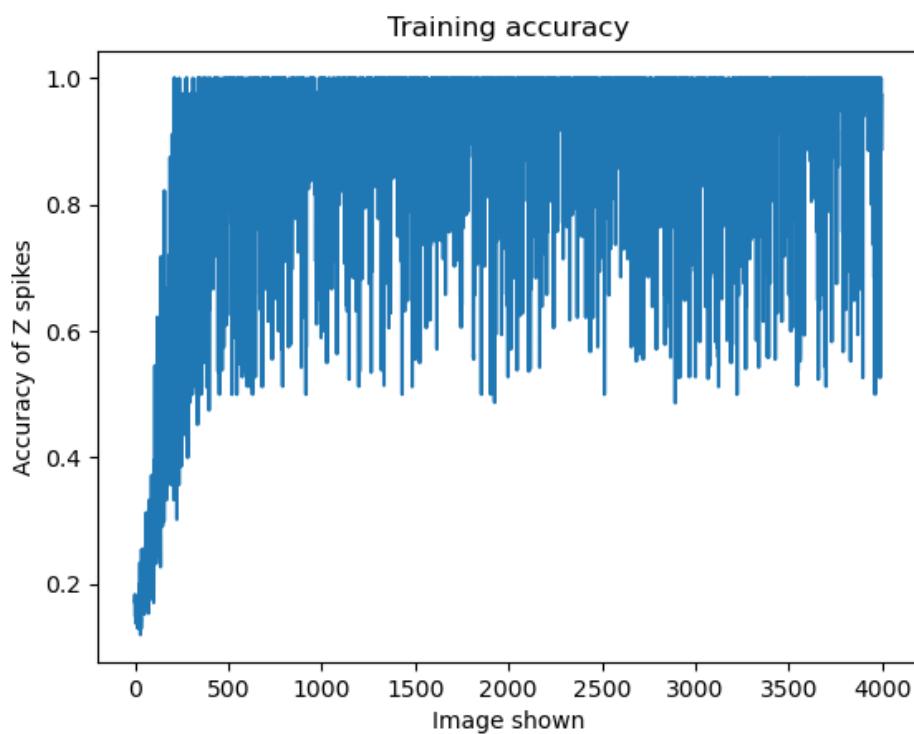


Figure 4.49: Proportion (accuracy) of most active output neuron to activity of all other output neurons during the presentation duration of each training image.
 $c = 20, \lambda = 10^{-3}$

to the network for 200 ms. During each image presentation duration the most active output neuron was recorded. This yielded Figure 4.50.

Also the number of distinct output neurons firing during each image presentation was recorded and is shown in 4.51. In this figure it seems that the points in which there are three distinct output neurons active are periodic in nature. This can be explained by Figure 4.52. There it can be seen that whenever the image orientation nears the border between two competing output neurons both start to be active. This explains why for large parts of Figure 4.51 2 neurons are active, as large portions of the bar in the image is overlapping the areas of the two nearest output neurons, thus producing high membrane potentials for both. The third distinct output neuron that is occasionally active seems to be of stochastic nature as much of the bars in the images overlaps areas of all other output neurons, thus generating a non zero membrane potential for each output neuron. However the occurrence of 3 distinct active output neurons seems to mostly occur at or close to the border between two competing output neurons, as there are already 2 distinct neurons firing by design.

Also it was possible to project the learned weights w_{ki} into the 2-D space to observe what they represent. This projection can be seen in Figure 4.53 for the weights of y_{10} .

The results for smaller λ were also valid, but they did not seem to yield superior results to the parameters $c = 20$ and $\lambda = 10^{-3}$. As with smaller learning rates the training simply took longer and the performance of the network did not improve they were discarded and the learning rate $\lambda = 10^{-3}$ was declared winner for $c = 20$. To save space the figures of smaller learning rates will not be shown here.

$c = 30$ also yielded a functioning network, but it did perform analogous to $c = 20$. It did perform in the same way, as with $c = 20$ the output neurons already fire every 5 ms, slowed down by the inhibition. By increasing c further the output neurons did not increase their activity.

4.6 Experiment 1: Rotated Bars

Most active Z neuron depending on angle

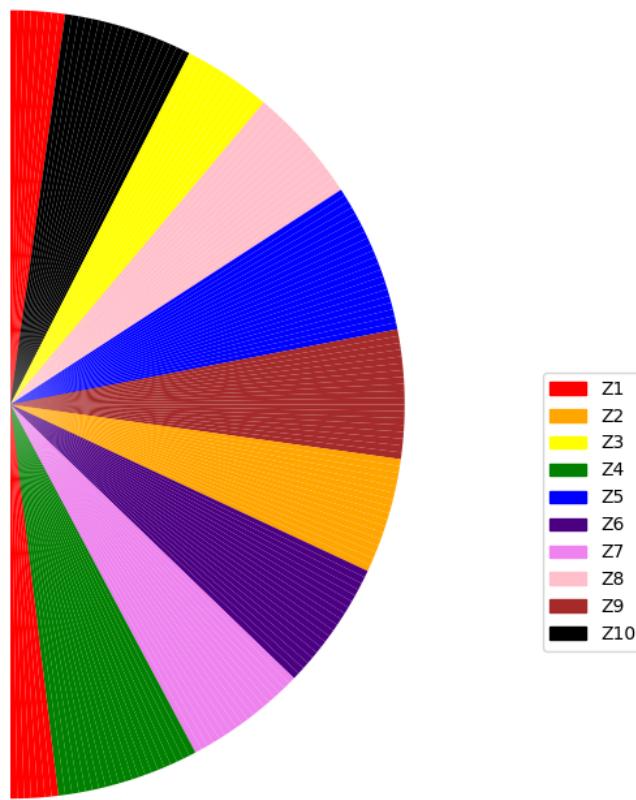


Figure 4.50: Most active output neuron depending on orientation of the training image during the training process. $c = 20, \lambda = 10^{-3}$

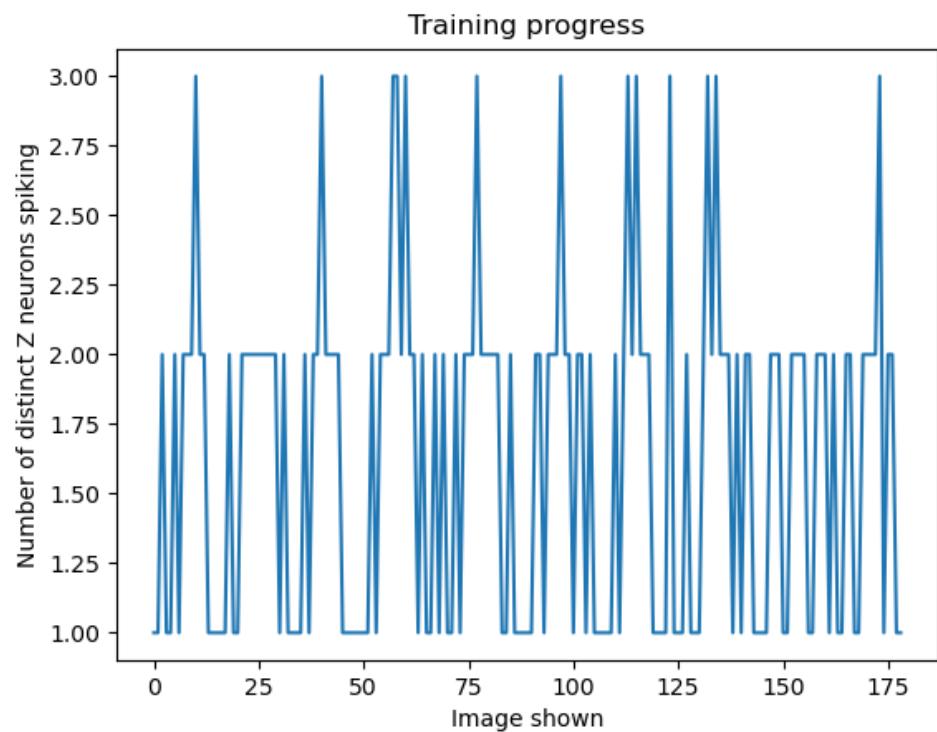


Figure 4.51: Number of distinct output neurons active during the presentation duration of each training image. x-axis: image shown, y-axis: distinct output neuron active, $c = 20, \lambda = 10^{-3}$

4.7 Experiment 3: Rotated bars with adaptive inhibition

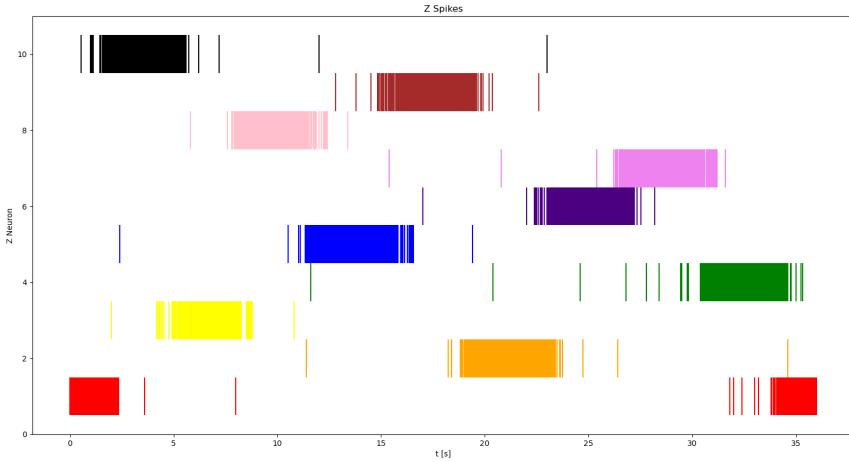


Figure 4.52: Output spikes over the presentation of input images from 0 to 179° , $c = 20$, $\lambda = 10^{-3}$

4.6.4 Conclusion

The parameters $c = 20$ and $\lambda = 10^{-3}$ were finally chosen as the network performed the best and trained the quickest with these parameters, without raising the membrane potential needlessly.

4.7 Experiment 3: Rotated bars with adaptive inhibition

4.7.1 Introduction

This experiment analyses a different approach to the implementation of the inhibition of the output neurons. In Experiment 4.6 and 4.4 the output neurons were stopped from firing for a defined time window after any output neuron fired. The time window during which the inhibition was active was 5 ms, which resulted in an average output firing rate of 200 Hz.

4 Experiments

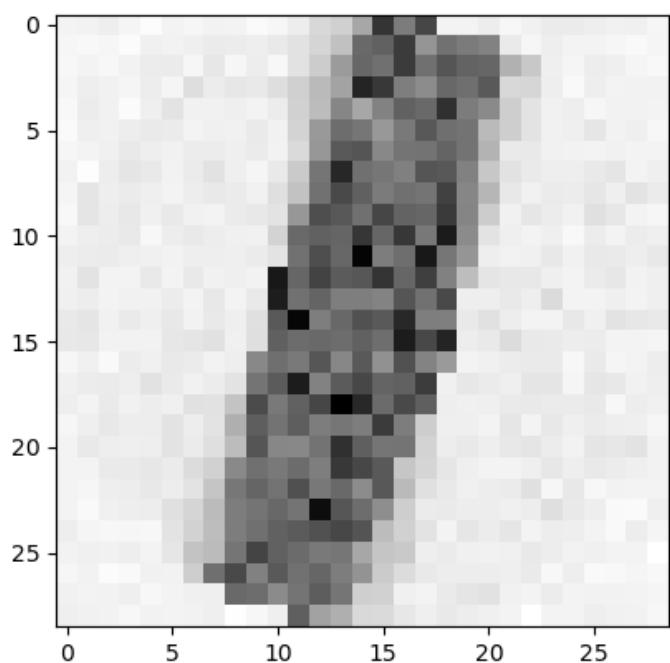


Figure 4.53: Visualization of the learned weights of y_{10} .

4.7 Experiment 3: Rotated bars with adaptive inhibition

In this experiment an adaptive inhibition will be used which will regulate the membrane potentials of the output neurons so that all of them together fire with 200 Hz on average. The distinction to the previous experiments is that there never is a time window in which no output neuron may fire. Also it is assumed that the weight shifting parameter c will not be needed to be fitted to the network, as the adaptive inhibition regulates the output firing rate of the network.

4.7.2 Methods

The adaptive inhibition is given by Equation 4.24 which was already used in Experiments 4.6 and 4.4. However the total output firing rate $R(t)$ was set to 200 Hz in this experiment.

4.7.3 Results

Several different values for the parameter c were tested. As expected the firing rate of the output neurons 200 Hz on average regardless of the value of c . However for values of c bigger than 100 the network did not learn correctly. For those values some output neurons learned to respond to areas of more than 18° , while other neurons did not respond to any areas. For $c = 20$ the network performed equally to Experiment 4.6. The results for that parameter can be seen in Figures 4.54, 4.55 and 4.56.

4 Experiments

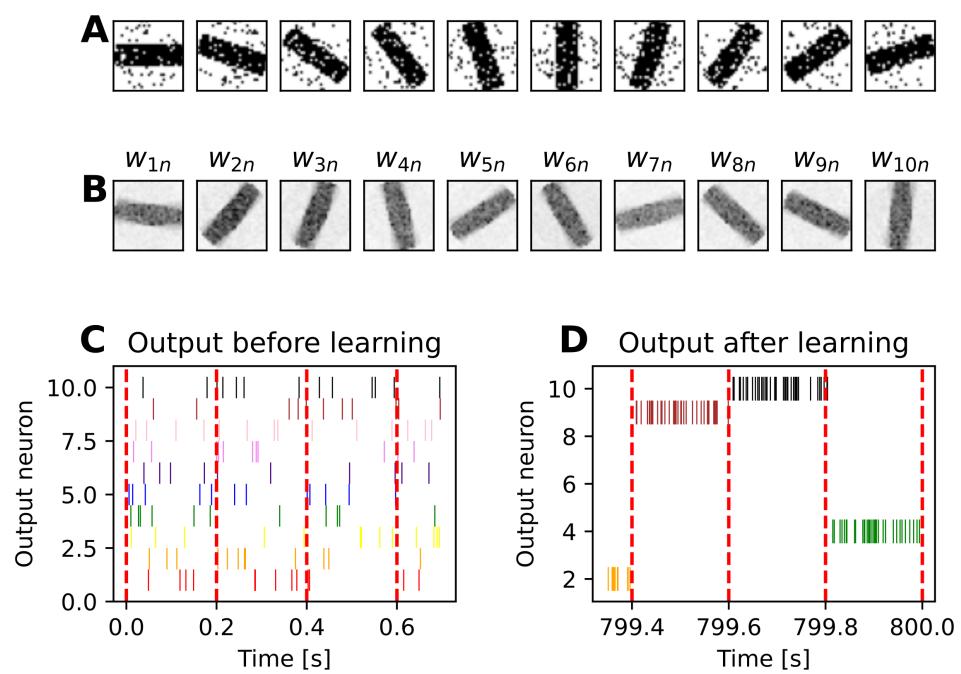


Figure 4.54: **Training.** **A** Examples of 29×29 -pixel input images of rotated bars and background noise. **B** Learned weights of the connections between input and output neurons. **C, D** Spike activity expressed by the output neurons before and after the training of the network.

4.7 Experiment 3: Rotated bars with adaptive inhibition

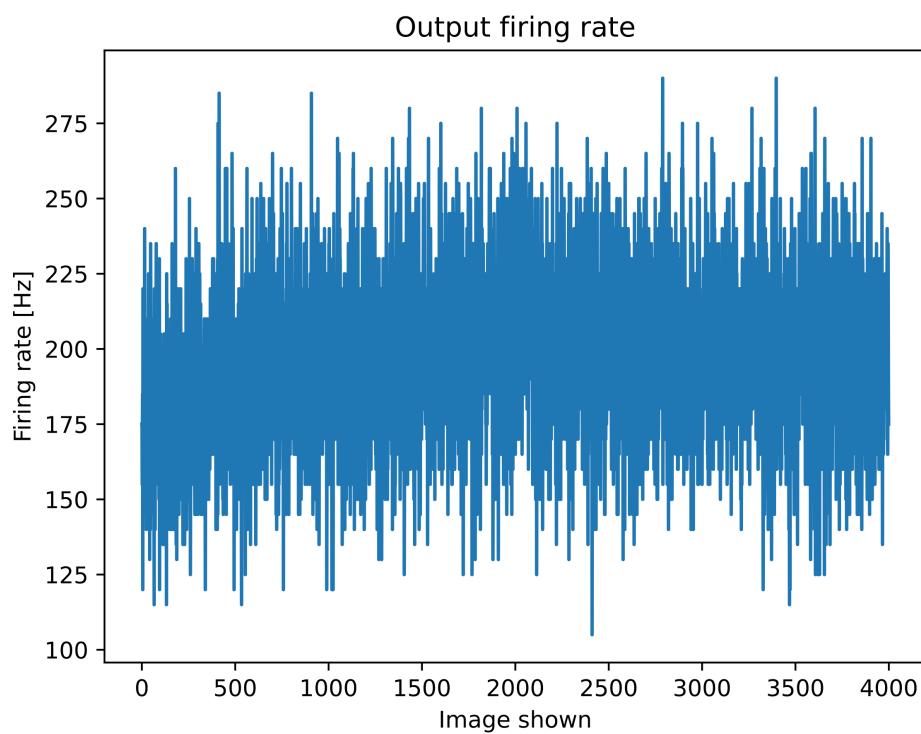


Figure 4.55: Firing rate of all output neurons combined over the training process.

4 Experiments

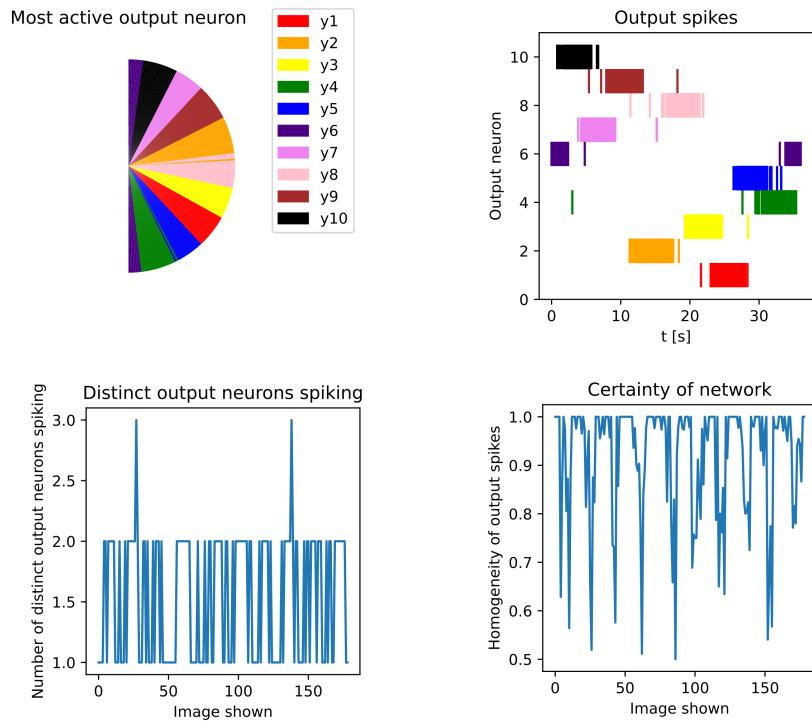


Figure 4.56: Validation. **A** Most active output neuron depending on orientation of the training image. Training images were shown for 200 ms of each possible orientation in 1° steps. **B** Spike activity expressed by the output neurons during the validation process described in (B). **C** Number of distinct output neurons active during the presentation of each validation image. **D** Proportion (accuracy) of most active output neuron to activity of all other output neurons during the presentation duration of each training image.

5 Discussion

Bibliography

- Arbib, Michael A. (2003). *The handbook of brain theory and neural networks 2nd Edition*. Cambridge, MA, USA: MIT Press, pp. 1228–1231. ISBN: 0-262-01197-2 (cit. on p. 3).
- Contreras, Diego (2004). “Electrophysiological classes of neocortical neurons.” In: *Neural Networks 17.5. Vision and Brain*, pp. 633–646. ISSN: 0893-6080. DOI: <https://doi.org/10.1016/j.neunet.2004.04.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0893608004000863> (cit. on p. 3).
- Dan Yang, Poo Mu-Ming (2004). “Spike timing-dependent plasticity of neural circuits.” In: *Neuron 44*, pp. 23–30. DOI: [10.1016/j.neuron.2004.09.007](https://doi.org/10.1016/j.neuron.2004.09.007) (cit. on p. 2).
- Darlington, Timothy R, Jeffrey M Beck, and Stephen G Lisberger (Oct. 2018). “Neural implementation of Bayesian inference in a sensorimotor behavior.” en. In: *Nat. Neurosci. 21.10*, pp. 1442–1451 (cit. on p. 6).
- Feldman, Daniel E. (2012). “The Spike-Timing Dependence of Plasticity.” In: *Neuron 75.4*, pp. 556–571. ISSN: 0896-6273. DOI: <https://doi.org/10.1016/j.neuron.2012.08.001>. URL: <https://www.sciencedirect.com/science/article/pii/S0896627312007039> (cit. on p. 2).
- Funamizu Akihiro Kuhn Bernd, Doya Kenji (Dec. 2016). “Neural substrate of dynamic Bayesian inference in the cerebral cortex.” In: *Nature Neuroscience 19*. DOI: [doi:10.1038/nrn.4390](https://doi.org/10.1038/nrn.4390) (cit. on pp. 1, 6).
- Gerstner, Wulfram and Werner M. Kistler (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press (cit. on pp. 2, 6, 9).
- Guo, Shangqi et al. (2019). “Hierarchical Bayesian Inference and Learning in Spiking Neural Networks.” In: *IEEE Transactions on Cybernetics 49.1*, pp. 133–145. DOI: [10.1109/TCYB.2017.2768554](https://doi.org/10.1109/TCYB.2017.2768554) (cit. on p. 2).
- Huff T Mahabadi N, Tadi P. (2024). *StatPearls [Internet]*. StatPearls Publishing. Chap. Neuroanatomy, Visual Cortex. (Cit. on p. 5).

Bibliography

- Jolivet, Renaud et al. (Aug. 2006). "Predicting spike timing of neocortical pyramidal neurons by simple threshold models." In: *Journal of Computational Neuroscience* 21.1, pp. 35–49. ISSN: 1573-6873. doi: [10.1007/s10827-006-7074-5](https://doi.org/10.1007/s10827-006-7074-5) (cit. on p. 15).
- Jones EG; Hendry, SHC, ed. (1984). *Cerebral cortex: cellular components of the cerebral cortex*. New York: Plenum Press. Chap. Basket cells, pp. 309–334 (cit. on p. 3).
- Joyce, James (2019). "Bayes' Theorem." In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2019. Metaphysics Research Lab, Stanford University (cit. on p. 11).
- Lee TS, Mumford D. (July 2003). "Hierarchical Bayesian inference in the visual cortex." In: *J Opt Soc Am A Opt Image Sci Vis*. doi: [doi:10.1364/josaa.20.001434](https://doi.org/10.1364/josaa.20.001434) (cit. on pp. 1, 6, 8).
- Maass, Wolfgang (Nov. 2000). "On the Computational Power of Winner-Take-All." In: *Neural Computation* 12.11, pp. 2519–2535. ISSN: 0899-7667. doi: [10.1162/089976600300014827](https://doi.org/10.1162/089976600300014827). eprint: <https://direct.mit.edu/neco/article-pdf/12/11/2519/814328/089976600300014827.pdf>. URL: <https://doi.org/10.1162/089976600300014827> (cit. on pp. 2, 3).
- Meunier, David et al. (2009). "Hierarchical modularity in human brain functional networks." In: *Frontiers in Neuroinformatics* 3. ISSN: 1662-5196. doi: [10.3389/neuro.11.037.2009](https://doi.org/10.3389/neuro.11.037.2009). URL: <https://www.frontiersin.org/articles/10.3389/neuro.11.037.2009> (cit. on pp. 1, 4).
- Nessler, Bernhard et al. (Apr. 2013). "Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity." In: *PLOS Computational Biology* 9.4, pp. 1–30. doi: [10.1371/journal.pcbi.1003037](https://doi.org/10.1371/journal.pcbi.1003037). URL: <https://doi.org/10.1371/journal.pcbi.1003037> (cit. on pp. 1, 2, 12–14, 17, 67, 74, 76, 77).
- Palmer, Stephen (Jan. 1999). *Vision Science: From Photons to Phenomenology*. Vol. 1, p. 153 (cit. on p. 5).
- Parr, Thomas and Karl J. Friston (2018). "The Anatomy of Inference: Generative Models and Brain Structure." In: *Frontiers in Computational Neuroscience* 12. ISSN: 1662-5188. doi: [10.3389/fncom.2018.00090](https://doi.org/10.3389/fncom.2018.00090). URL: <https://www.frontiersin.org/articles/10.3389/fncom.2018.00090> (cit. on pp. 1, 6).
- Pouget, Alexandre et al. (Sept. 2013). "Probabilistic brains: knowns and unknowns." In: *Nature Neuroscience* 16.9, pp. 1170–1178. ISSN: 1546-1726.

Bibliography

- DOI: [10.1038/nn.3495](https://doi.org/10.1038/nn.3495). URL: <https://doi.org/10.1038/nn.3495> (cit. on p. 6).
- Riesenhuber, Maximilian and Tomaso Poggio (2002). "Neural mechanisms of object recognition." In: *Current Opinion in Neurobiology* 12.2, pp. 162–168. ISSN: 0959-4388. DOI: [https://doi.org/10.1016/S0959-4388\(02\)00304-5](https://doi.org/10.1016/S0959-4388(02)00304-5). URL: <https://www.sciencedirect.com/science/article/pii/S0959438802003045> (cit. on p. 5).
- Rodney J Douglas, Kevan A C Martin (2004). "Neuronal circuits of the neocortex." In: *Annual review of neuroscience* 27, pp. 419–451. DOI: [10.1146/annurev.neuro.27.070203.144152](https://doi.org/10.1146/annurev.neuro.27.070203.144152) (cit. on pp. 2, 3).
- Taherkhani, Aboozar et al. (2020). "A review of learning in biologically plausible spiking neural networks." In: *Neural Networks* 122, pp. 253–272. ISSN: 0893-6080. DOI: <https://doi.org/10.1016/j.neunet.2019.09.036>. URL: <https://www.sciencedirect.com/science/article/pii/S0893608019303181> (cit. on p. 10).