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, August 2023

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Abstract

This thesis explores hierarchical architectures of spiking Winner-Take-All (WTA) networks, with the focus on simulating feedback mechanisms found in the visual cortex. Such feedback mechanisms are related to attention and consistent beliefs across different of the visual cortex. First, the biological and theoretical concepts of spiking neural networks and their relationship to Bayesian inference are established. It is demonstrated how the hypothesized probabilistic nature of the brain can be linked to the model of a spiking WTA network that performs Bayesian inference. The thesis involves a series of experiments designed to test the network's response to visual stimuli. The response of the network to ambiguous images is tested and it is shown that the added feedback makes a crucial contribution to interpreting such images. An effect of the visual cortex of seeing illusory lines is reproduced by feeding the network feedback that contradicts the visual input. Furthermore, the model's link to Bayesian inference is verified by calculating conditional probabilities of neurons and then deriving their synaptic weights from them. To gain a better understanding of the network model the impact of the different network hyperparameters is analysed. An unexpected property of the firing frequencies of input and prior neurons is discovered, which changes the output's probability distribution of the network. The reusability of hyperparameters to networks of different sizes is tested. It is found that the hyperparameters are not universal to all network sizes. The quality of the training of the network was compared to the analytical optimum and it was found that the training could not quite reach it. These experiments combined reveal that incorporating feedback enhances the network's ability to process visual information, reflecting behaviours observed in biological systems.

Contents

List of Figures

List of Tables

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. The feedback from high-level to low-level cortical areas can be modelled as the prior probability (Nessler et al., 2013).

There are various models for the computational dynamics of biological neurons (Gerstner and Kistler, 2002). One model which is well supported are spiking neural networks. These models aim to resemble biological neural networks more closely than common artificial neural networks. They generate neuron spikes which increase the membrane potentials of other neurons they are connected to. The impact those neuron spikes have is often modelled via Spike Timing Dependent Plasticity (STDP) which takes the exact timing of pre- and postsynaptic spikes into account. This form of plasticity agrees with biological experiments (Feldman, 2012; Dan Yang, 2004). 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). Additionally Winner-Take-All modules perform computationally efficient (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 spiking neural network that receives visual input and also feedback from another network higher up in the hierarchy.

In Chapter 2 the biological background for this thesis and the underlying hierarchical spiking Winner-Take-All network are given. The hierarchical structure of the brain is explained and further looked at via an example of hierarchy in the visual cortex. To better understand the example a brief overview of the visual cortex is given first. An argument, supported by experiments, for the probabilistic brain is given. This hypothesis is essential for this thesis, as the network model is of probabilistic nature. Finally synaptic plasticity and spiking neural networks are explained. The theoretical background, explaining the mathematical model used for the neural networks in this thesis, is explained in Chapter 3. In it Bayesian inference is explained and how it can be applied to the example of hierarchical feedback in the visual cortex. After that the network model, used for the simulation of the

experiments, is explained and all necessary equations are provided. Finally, the link between the spiking Winner-Take-All network model and Bayesian inference is explained and shown. The various performed experiments are provided in Chapter 4. There are four different experiments that helped to analyse the network model and the aforementioned mathematical link between the model and Bayesian inference. The first experiment shows how the network can decide how to interpret ambiguous images with the help of the additional added feedback. This behaviour of the network can be interpreted as it setting attention on some part of an image. Further, it is shown how the output neurons learned to respond to specific areas of images through unsupervised learning. In the second experiment the conditional probabilities of the input and prior neurons are calculated and from those probabilities the weights of the network are determined. This proves the link between the spiking Winner-Take-All network model and Bayesian inference experimentally. Then the optimal network hyperparameters are searched and their influence on the network analysed. Furthermore, this experiment shows output for which there is no visual input, owed to the feedback. This behaviour is similar to the behaviour of seeing illusory lines observed in the visual cortex by Lee TS (2003). In Experiment 3 the transferability of network hyperparameters to networks of different sizes is tested. This proved impossible as the firing frequencies change the distribution of the output probabilities of the network. At last in Experiment 4 the optimal hyperparameters from Experiment 2 are used to train weights and to test how close to the optimal solution one can get by training the weights. Through that, problems in the training process of the network can be identified. Finally, in Chapter 5, the results and implications of the experiments are 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. However, 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 can 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

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

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 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 Synaptic 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 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 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).

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

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

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

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

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 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. 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

$$P(Y=k|X=x,Z=j)=\frac{P(X=x|Y=k)P(Y=k|Z=j)}{\Sigma_{k'}P(X=x|Y=k')P(Y=k'|Z=j)}. \tag{3.3}$$

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.

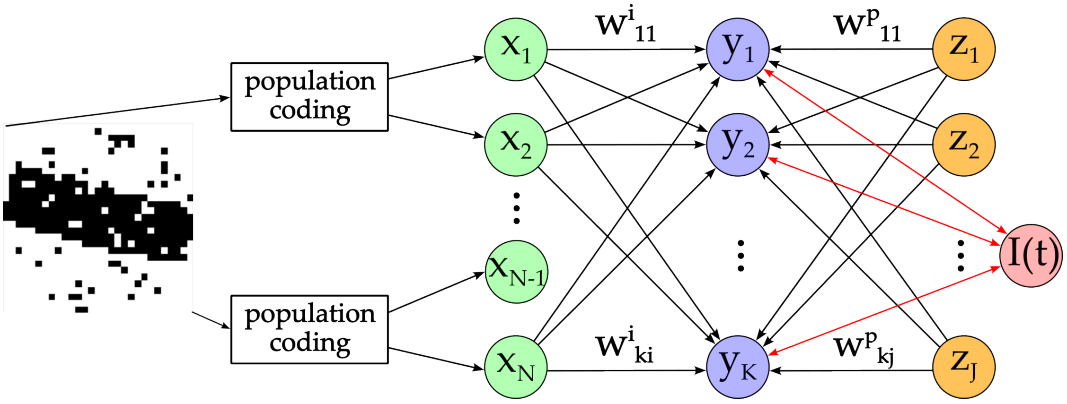

Figure 3.1: Architecture of the network.

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 $x_1, ..., x_N$. These input neurons are fully connected to the excitatory output neurons $y_1, ..., y_K$. 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 $z_1, ..., z_J$ were added to the network. The prior neurons were also fully connected to the output neurons. A visualization of the network architecture can be seen in Figure 3.1.

Neuron model As in Nessler et al. (2013) the input neurons $x_1, ..., x_N$ 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 3.2. 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

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

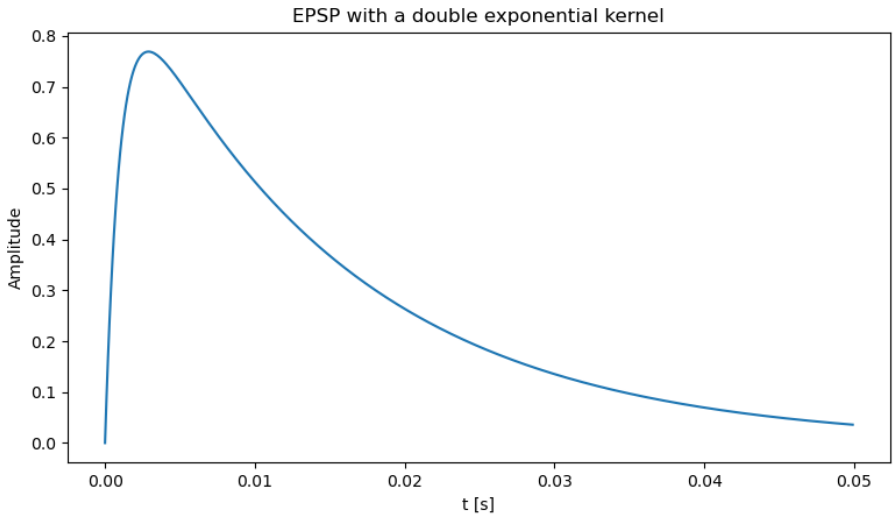

Figure 3.2: Form of an excitatory post synaptic potential generated by an input neuron over time. A double exponential kernel was used to generate this signal. Its value is passed on to the next layer of the network.

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

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

The membrane potential u_k of each output neuron is calculated by multiplying the EPSP of each input neuron times the input weight w^I_{ki} of the connection between them, plus the EPSP of each prior neuron times the prior weight w^P_{kj}

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

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 excitabilities of the output neurons would all end up being equal to each other.

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 $R(t) = 200Hz$ 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 3.13 for I(t) yields

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

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

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

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

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 $I(t)$

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

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

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

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

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

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

$$r_k(t)\delta t. \tag{3.14}$$

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

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

The inhibition term cancels out because all output neurons receive the same inhibition, meaning that it is independent of k.

Spike timing dependent plasticity The input weights w_{ki}^I 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

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

where λ is the learning rate, the hyperparameter 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 hyperparameter 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 too large parts of the input, while others might not respond 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^P_{kj} are also updated whenever an output neuron fires, in the same way as w^I_{ki}

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

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

$$w^I_{ki} = log(p(x_i = 1 | y_k = 1)) \tag{3.18}$$

an will be analysed in Section 4.2. 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 posterior probability of Bayesian inference given in Equation 3.3 is equal to the relative firing probability $q_k(t)$, given in Equation 3.15.

As explained in Section 3.2 the visual input, modelled by the input neurons, can be thought of as the Bayesian likelihood and the feedback, modelled by the prior neurons, as the Bayesian prior. As defined in Equation 3.11 the firing probability of an output neuron is proportional to the exponent of its membrane potential minus the inhibition. To obtain the Bayesian likelihood and prior we first split the membrane potential into the contributions of the input and prior neurons. The first part that signifies the contribution of the input neurons is given by

$$e^{u_x} = e^{\sum_{i=1}^{N} w_{ki}^{I} \cdot x_i(t)}. \tag{3.19}$$

The second part gives the contribution of the prior neurons as

$$e^{u_z} = e^{\sum_{j=1}^{J} w_{kj}^{P} \cdot z_j(t)}. \tag{3.20}$$

These two parts can be seen as two partial membrane potentials. By inserting each of them into Equation 3.11 we get the likelihood

$$P(X|Y) = e^{u_x - I(t)}. \tag{3.21}$$

and the prior

$$P(Y|Z) = e^{u_z - I(t)}. \tag{3.22}$$

When inserting the likelihood and the prior into Equation 3.3 we get

$$
\begin{aligned}
P(Y=k|X=x,Z=j) &= \frac{e^{\sum_{i=1}^{N} wi_{ki}\cdot x_i(t)-I(t)} \cdot e^{\sum_{j=1}^{J} w^P_{kj}\cdot z_j(t)-I(t)}}{\sum_{k'=1}^{K} e^{\sum_{i=1}^{N} w^I_{k'i}\cdot x_i(t)-I(t)} \cdot e^{\sum_{j=1}^{J} w^P_{k'j}\cdot z_j(t)-I(t)}} \\
&= \frac{e^{-I(t)\cdot N+\sum_{i=1}^{N} w^I_{ki}\cdot x_i(t)} \cdot e^{-I(t)\cdot J+\sum_{j=1}^{J} w^P_{kj}\cdot z_j(t)}}{\sum_{k'=1}^{K} e^{-I(t)\cdot N+\sum_{i=1}^{N} w^I_{k'i}\cdot x_i(t)} \cdot e^{-I(t)\cdot J+\sum_{j=1}^{J} w^P_{k'j}\cdot z_j(t)}} \\
&= \frac{e^{-I(t)\cdot N} \cdot e^{\sum_{i=1}^{N} w^I_{ki}\cdot x_i(t)} \cdot e^{-I(t)\cdot J} \cdot e^{\sum_{j=1}^{J} w^P_{kj}\cdot z_j(t)}}{e^{-I(t)\cdot N} \cdot e^{-I(t)\cdot J} \cdot \sum_{k'=1}^{K} e^{\sum_{i=1}^{N} w^I_{k'i}\cdot x_i(t)} \cdot e^{\sum_{j=1}^{J} w^P_{k'j}\cdot z_j(t)}} \\
&= \frac{e^{\sum_{i=1}^{N} w^I_{ki}\cdot x_i(t)+\sum_{j=1}^{J} w^P_{kj}\cdot z_j(t)}}{\sum_{k'=1}^{K} e^{\sum_{i=1}^{N} w^I_{k'i}\cdot x_i(t)+\sum_{j=1}^{J} w^P_{k'j}\cdot z_j(t)}} \\
&= \frac{e^{U_k(t)}}{\sum_{k'=1}^{K} e^{U'_k(t)}} \\
&= q_k(t)
\end{aligned}
\tag{3.23}
$$

4 Experiments

4.1 Experiment 1: Horizontal and vertical bars

4.1.1 Introduction

For this experiment images of either horizontal or vertical bars will be shown to the WTA network. Through unsupervised learning the output neurons will learn to cluster the images together, depending on the orientation and position of the bars. The the learning of the network will be visualized and analysed. Further, the effect of the prior neurons will be demonstrated by generating ambiguous images which show a horizontal and a vertical bar at the same time. The output of the network will be analysed, depending on the activity of the prior neurons.

4.1.2 Methods

Input data Black bars with a width of seven pixels were drawn onto a white background with a size of 35 x 35 pixels. The bars could be oriented either horizontally or vertically. The network was supposed to identify ten different groups within these images, five with a horizontal orientation and five with a vertical orientation. With a given bar width of seven pixels the image height and width were determined as 35 pixels, to yield equally sized groups. Assuming equally sized groups and starting to count from position 0 the centers of the groups should then be at positions 3, 10, 17, 24, 31. The orientation of the training images was chosen randomly via a uniform distribution. The positions of the bars in the training images were also distributed randomly, via a uniform distribution, along the 35 pixels

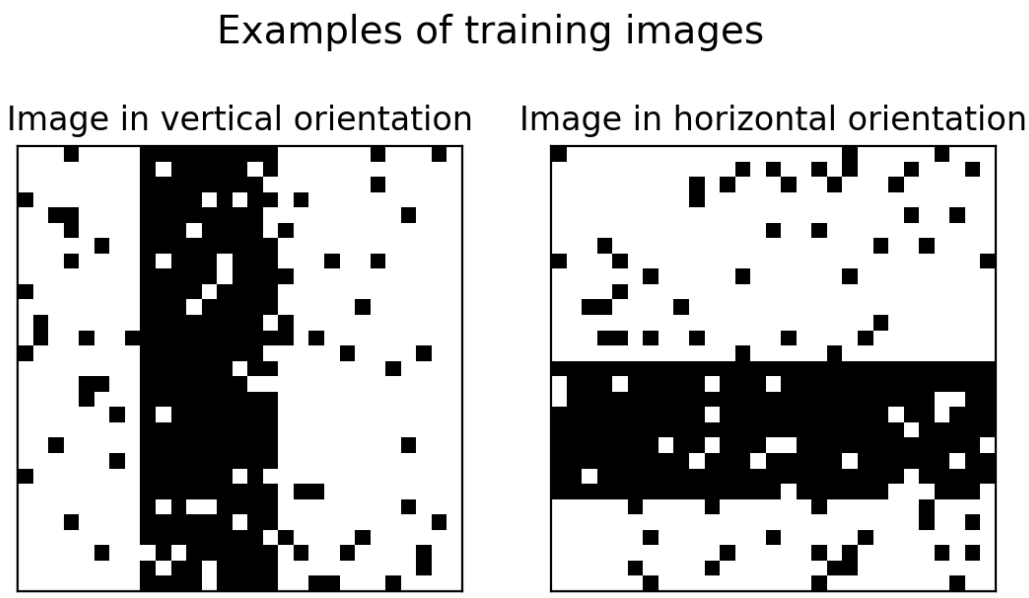

Figure 4.1: Training images generated for experiment 1. One image of each possible orientation at a random position.

of either axis. To simulate noise each pixel of an image had a chance of ten percent to have its color flipped after the generation. During the training of the network random images were generated and presented for 200 ms. As the simulation had a duration of 800 seconds this resulted in 4000 images being shown to the network. Examples of the input data can be seen in Figure 4.1. To show the value of the added a-priori information, validation images with two bars forming a cross were also generated, seen in Figure 4.2. When shown to the network in the validation process the prior neurons were given the information that a cross is either of horizontal or vertical orientation.

Network architecture As every pixel of an image was connected to two input neurons the network had 2450 input neurons. Each of these neurons had a firing frequency f_{input} of 20 Hz, when in the active state. Then there were ten output neurons, one for each possible group. Lastly, there were two equal sized groups of prior neurons with firing frequencies f_{prior} of

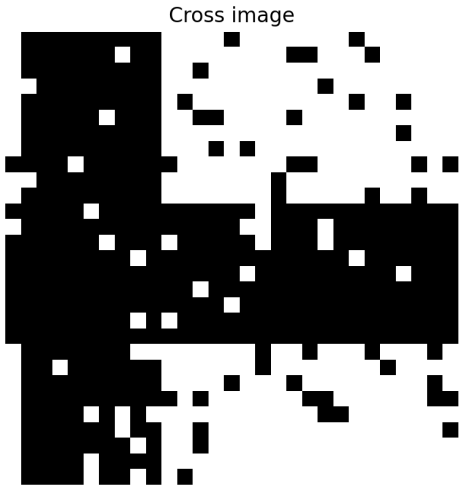

Figure 4.2: Generated cross image which can represent either horizontal or vertical orientation.

200 Hz. This firing frequency was assigned a high plausible value to keep the needed number of prior neurons small compared to the number of input neurons. During training the first group z^h was active when the image was of horizontal orientation and the other group z^v was active for vertical orientation. The number of prior neurons had to be determined via grid search, as they needed to be set at values where the impact of the a-priori information is neither too strong nor too weak.

Network and hyperparameters The simulation of the experiment was performed in time steps of 1 ms. This step size was used for all experiments. As given by Nessler et al. (2013) the time window σ was 10 ms, the time constant for the rise of the EPSPs τ_{rise} was 1 ms and the time constant for the decay of the EPSPs τ_{decay} was 15 ms. Before performing this experiment, Experiment 2 of Nessler et al. (2013) was reproduced to validate that the implementation of the simulation was correct. This proof of concept was omitted in this work, however within it the weight shift hyperparameter $c = 20$ and the learning rate $\lambda = 10^{-}3$ were determined via grid search. These hyperparameter values were reused for this experiment as the input data, as well as the network architecture, were similar.

4.1.3 Results

First different numbers of prior neurons were tested. Simulations with 10, 20, 50, 100 and 200 prior neurons were performed. For 50, 100 and 200 prior neurons the training process was impaired by the activity of the prior neurons. Some output neurons were responding to too large areas, while other prior neurons were not responding to any specific areas at all. This happened for example because the first output neuron that generated a spike for a horizontal bar got reinforced by the prior neurons to respond to all horizontal bars, no matter the position. Thus other output neurons were less likely to respond to horizontal bars, resulting in output neurons that were not responding to any specific orientation or area at all. With a number of 10 and 20 prior neurons each of the output neurons learned to respond to coherent areas of the images. The prior neurons also learned to respond to either horizontal or vertical images. To maximize the impact of the a-priori information the validation of the network was performed with 20 prior neurons, as it was the largest number that resulted in a properly trained network. The results of the training process can be seen in Figure 4.3. In Figure 4.3 A examples of input images are shown. The bars were positioned without overlapping each other, thus showing the optimal areas output neurons should learn to respond to. The areas that output neurons did learn to respond to can be seen in the visualization of the input weights in Figure 4.3 B. It can be seen that the training was successful as five output neurons responded to horizontal bars and the other five to vertical bars. Furthermore, each output neuron responded to different areas of the input image and there was little overlap between each other. The training progress of the network is given in Figure 4.3 E. When the network has completed the training it should mostly have only one or two output neurons being active for any input image. Although, there may be more active output neurons sporadically, due to the stochastic nature of the model. According to this plot the training process could have been stopped earlier after 2000 shown images. However, as there was no danger of overfitting the data more images were shown, to ensure that the network had always finished the training when the simulation stopped. In Figure 4.3 F the relative activity of the most active output neuron can be seen. First, it can also be used to judge the progress of the training process, as it increased over time. Furthermore

it can also be seen as a measure of how clearly an image belonged to the area the most active output neuron responded to. If the position of an image was exactly in the middle of the learned area of the most active output neuron the relative activity should be one. However, when the position was on the border between two learned areas of output neurons the relative activity should only be 0.5, as the image belonged to both areas. The learned prior weights can be seen in Figure 4.4. There it can be seen that the first 10 prior neurons learned the same pattern, as all of them were in an active state whenever vertical bars were shown to the network. When comparing these prior weights to the input weights in Figure 4.3 B it can be seen that the five highest prior weights of each prior neuron are between themselves and output neurons that learned to respond to vertical bars. The last 10 prior weights showed exactly the opposite pattern, having their five highest weight values between themselves and output neurons that responded to horizontal bars.

To validate the learned patterns the network was shown horizontal bars with a height of seven pixels with their centers at every possible position, beginning at position zero and incrementing in steps of one. Each image was shown for 200 ms each, while recording the output neuron activity. To visualize which output neuron is the most active for each position a horizontal bar with a height of one was drawn into a 35 x 35 pixel image, color-coded to represent the most active output neuron for that position. This can be seen in Figure 4.5 A. y_7 and y_{10} claimed areas with heights of eight pixels, while the areas of y_9 and y_4 had only a height of six pixels. Only y_2 had an area height of the expected seven pixels. These varying heights were expected to happen, due to the unsupervised learning method applied in this thesis. Depending on the randomly generated images the network received during the training process there may be more images on one side of the border between two output neurons than on the other side. This would result in the favoured neuron having stronger weights around the border area. Because of that the favoured neuron might end up claiming a larger active area. The output spikes of the network can be seen in Figure 4.5 B. It can be seen that outside of the border areas between the output neurons there was mostly only one neuron active. However, shortly after two seconds y_8 was active. The time frame between 2 seconds and 2.2 seconds this happened in corresponds to position 10. y_8 should mostly be

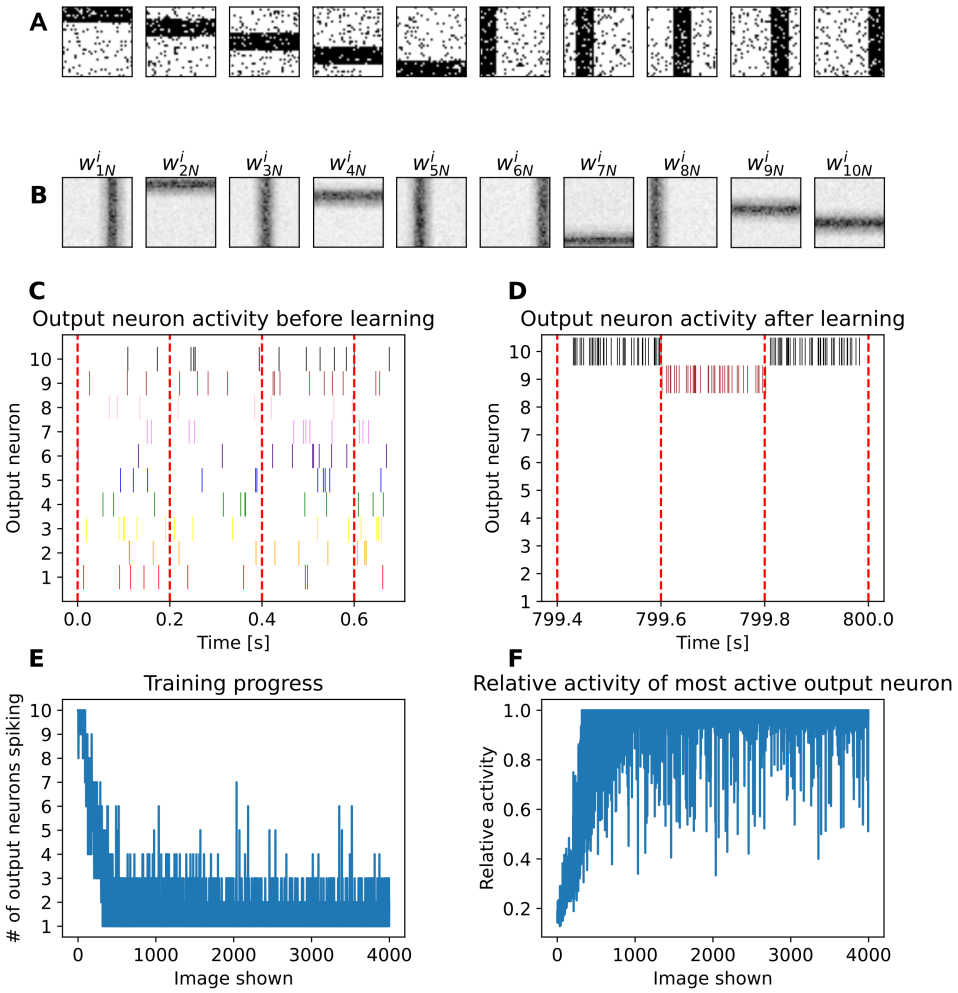

Figure 4.3: **Training with 20 prior neurons. A** Examples of 35 x 35-pixel input images of horizontal and vertical bars with background noise. They were positioned without overlapping each others bars, thus showing the optimal areas output neurons should learn to respond to. **B** Learned weights of the connections between the input neurons, that were active for black pixels of the input image, and output neurons. As there was one such input neuron per pixel of the image the weights could be plotted in the same format as the image to easily interpret them. **C, D** Spike activity expressed by the output neurons before and after the training of the network. **E** Number of active output neurons during the presentation duration of each training image. **F** This shows the number of spikes the most active output neuron generated, relative to the number of spikes all other output neurons generated combined.

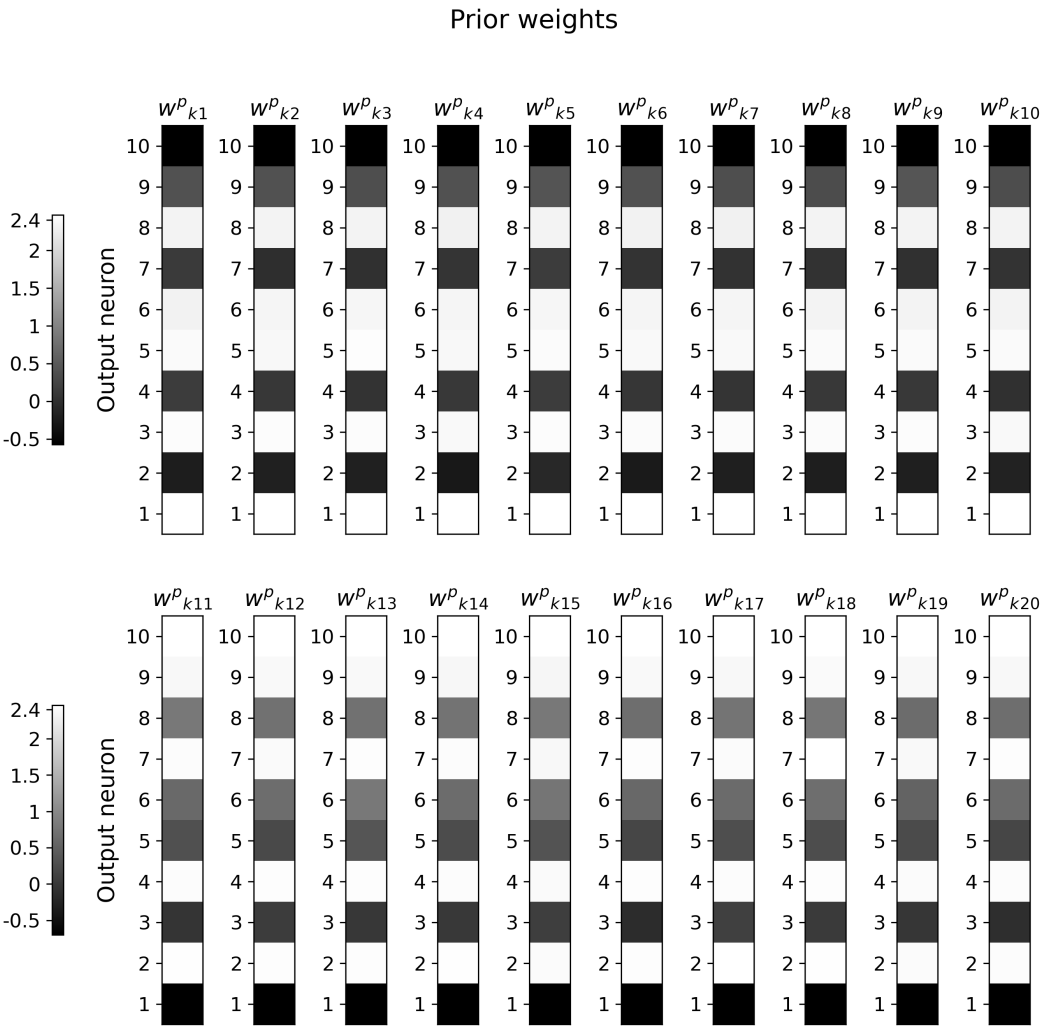

Figure 4.4: Learned weights of the connections between prior and output neurons.

active for vertical bars, however its learned area has some overlap with with a horizontal bar at position ten and thus there is always a small chance for it to generate a spike. In Figure 4.5 C the number of active output neurons for each position is given. The validation process was repeated ten times and the mean and standard deviation of the number were calculated. In this plot four peaks appeared. These peaks are around the borders between the adjacent active areas of output neurons seen in Figure 4.5 A. Around the border areas a number of active output neurons bigger than one was expected, as the horizontal bars reach into the areas of two output neurons at once. When looking at the number of active output neurons at position 10, which is approximately 1.1 with a high standard deviation, it shows that y_8 did not falsely learn to be active for horizontal bars, but rather that it was a stochastic outlier. Figure 4.5 D shows the relative activity of the most active output neuron. Compared to Figure 4.5 C it provides the additional information of how split the activity of neighbouring output neurons is in the border areas. For example it can be seen that at position 22 y_4 had a relative activity of 0.62. It was only barely able to be the most active neuron and might almost have had only an active area with a height of five pixels.

The validation process for the vertical bars was done analogously to the horizontal bars. Its results are given in Figure 4.6.

Next the impact of the prior was analysed. An image with a horizontal bar at position 12 and a vertical bar at position 5 on it was generated. First the prior neurons z^v were activated and z^h were deactivated. Then the image was shown to the network for 200 ms. After that the activity of the prior neurons was reversed and the image was shown again. The results can be seen in Figure 4.7. In that figure it can be seen that the output of the network completely changes depending of the prior activity. The prior neurons determine if the network focuses on the vertical or horizontal bar of the image.

To further illustrate the dependence of the output on the prior the firing frequencies of the prior neurons were gradually changed. The starting firing frequency of z^h was set to 200 Hz and to 0 Hz for z^v. For those firing frequencies a cross image was shown to the network for 200 ms. It had a horizontal bar at position 31 and a vertical bar at position 3. These position are as far at the edge of the image as possible, while still displaying the

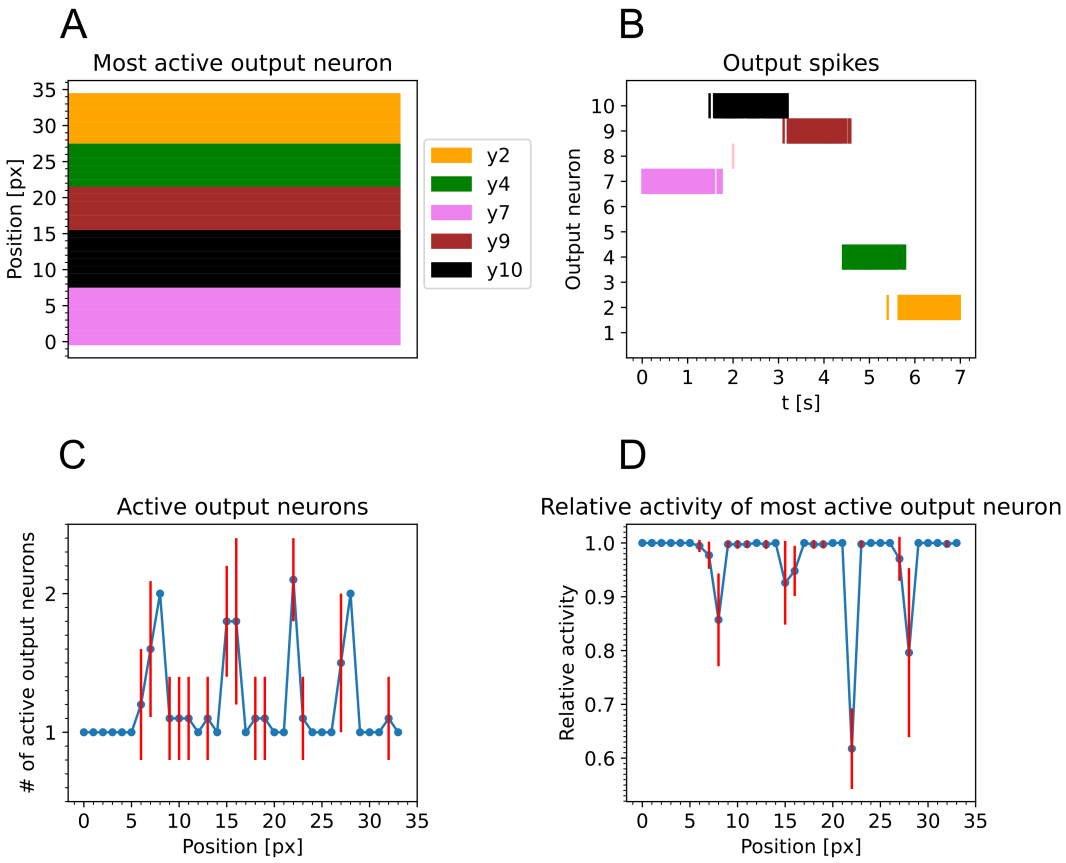

Figure 4.5: **Horizontal validation.** Horizontal bars with a height of seven pixels were shown to the network at different positions. **A** Most active output neuron, depending on the vertical position of the center of a horizontal bar. **B** Output spikes during the validation process. The ticks of the x-axis are in steps of 0.2 seconds, thus each tick signifies the start of a new image being shown. **C** The mean number of active output neurons, depending on the vertical position of a horizontal bar. The standard deviation is given by the red bars. **D** This shows the number of spikes that the most active output neuron generated, relative to the number of spikes all other output neurons generated combined. The blue dots indicate the mean and the red bars the standard deviation.

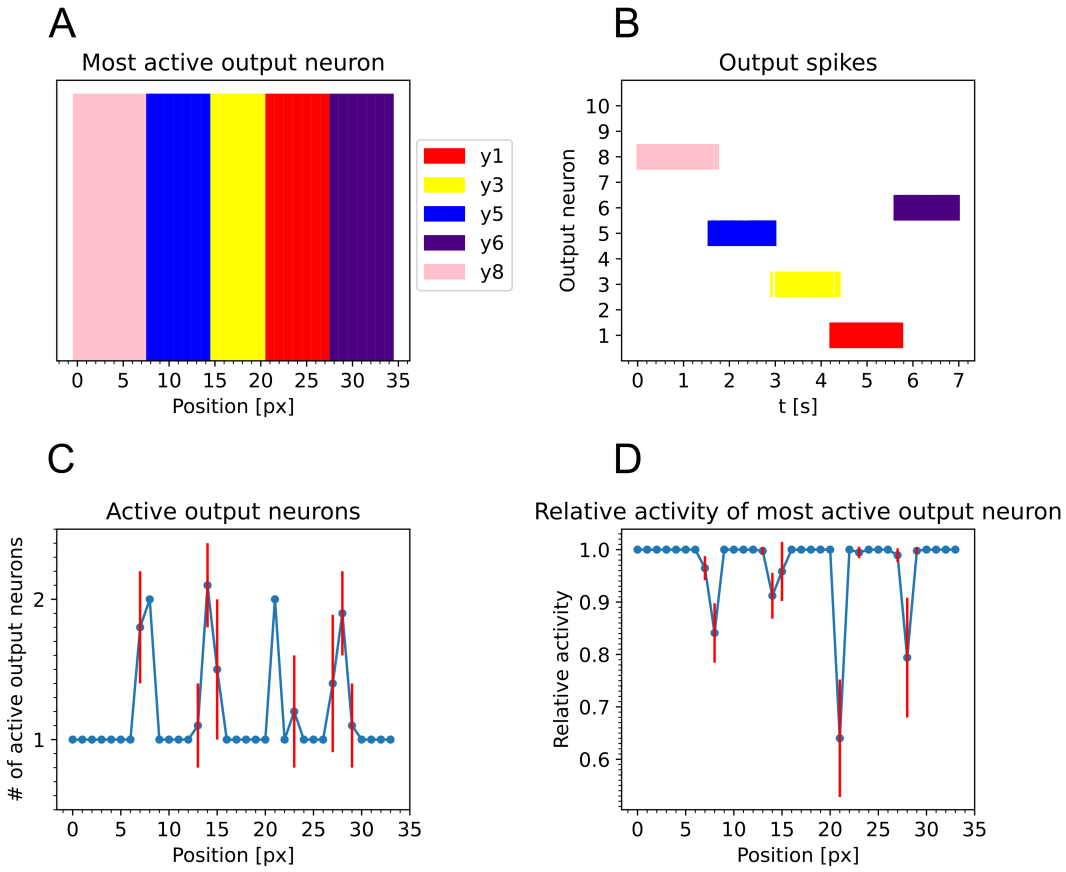

Figure 4.6: **Vertical validation.** Vertical bars with a height of seven pixels were shown to the network at different positions. **A** Most active output neuron, depending on the horizontal position of the center of a vertical bar. **B** Output spikes during the validation process. The ticks of the x-axis are in steps of 0.2 seconds, thus each tick signifies the start of a new image being shown. **C** The mean number of active output neurons, depending on the horizontal position of a vertical bar. The standard deviation is given by the red bars. **D** This shows the number of spikes that the most active output neuron generated, relative to the number of spikes all other output neurons generated combined. The blue dots indicate the mean and the red bars the standard deviation.

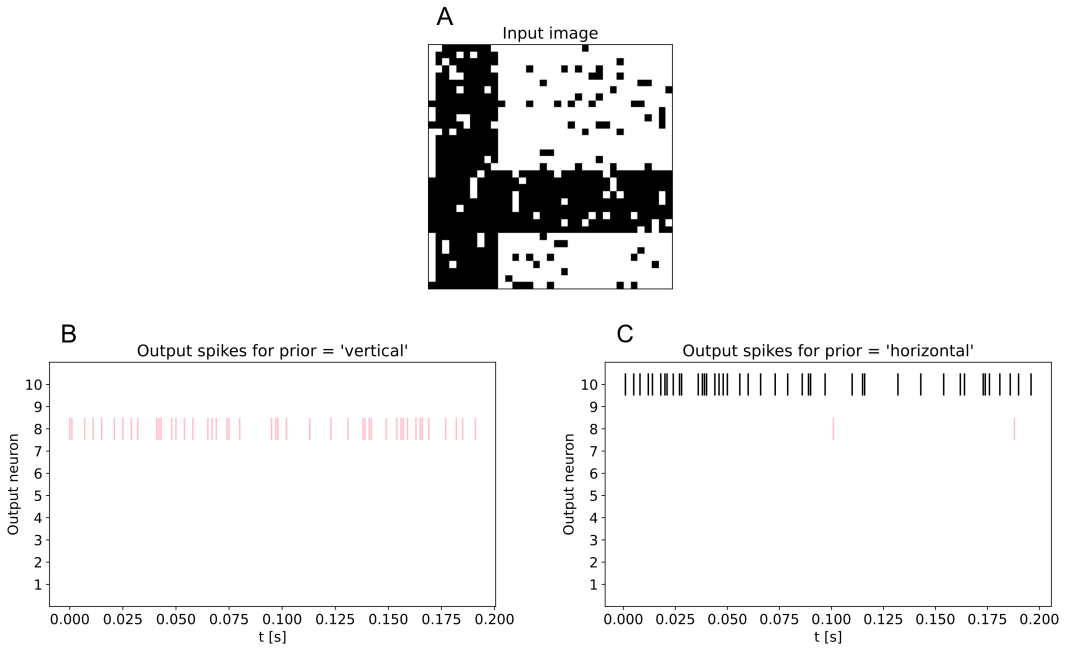

Figure 4.7: **Impact of the prior Neurons. A** Validation image with a horizontal and a vertical bar on it. **B** Spiking activity of the output neurons with z^v being active. **C** Spiking activity of the output neurons with z^h being active.

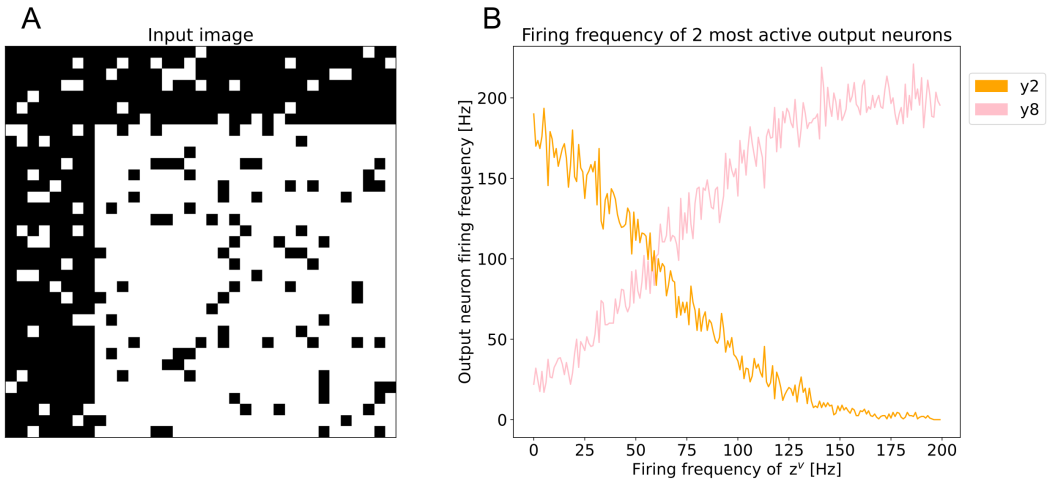

Figure 4.8: **Cross image with varying prior neuron activity. A** Validation cross image. **B** Firing frequency of the 2 most active output neurons, depending on the firing frequency of z^v. The output neuron firing frequencies were averaged over 10 runs to smooth the graphs.

whole width of the bars. After each image presentation duration the firing frequency of z^h was decreased by 1 Hz and increased by 1 Hz for z^v. The used cross image can be seen in Figure 4.8 A. The firing frequency of the 2 most active output neurons depending on the firing frequency of z^v can be seen in Figure 4.8 B. At the beginning y_2, which represents the horizontal part of the cross image, was the most active neuron, which is correct as the prior neurons fully supported the horizontal interpretation of the input image. With rising firing frequency of z^v the activity of the y_2 decreased and the activity of y_8 increased. This happened because the prior neurons gradually supported the interpretation of the image as vertical more and more. It was expected that the crossing point of the two graphs in Figure 4.8 B would be at a firing frequency of z^v of 100 Hz, however the graphs crossed at 58.6 Hz. In Figure 4.5 D the relative activity of y_2 at the edge of the bar at position 28 was 0.79. In Figure 4.6 D the relative activity of y_8 at the edge of the bar at position 6 was 1.0. y_8 has a higher relative activity at that position than y_2 at position 28, because y_8 has an active area of width 8, making position 6 not the border of the active area. Due to the larger active area of y_8 the crossing point shifted in favour of y_8.

4.2 Experiment 2: Mathematical analysis of the network with 1-D images

4.2.1 Introduction

The purpose of this experiment was to analyse a smaller network and provide a definitive way to verify the performance of the simulation. Furthermore, the weights were not learned as in the previous experiment, but rather determined analytically. This was done to experimentally prove the relation given by Nessler et al. (2013) that the weights stochastically converge toward the conditional probability that a presynaptic neuron fired shortly before the postsynaptic neuron. First 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 weights were determined by utilizing the logarithmic connection between them and the corresponding conditional probabilities, as given by Equation 3.18. The conditional probabilities of the input and output neurons of the network were calculated and then used to determine the posterior probability of the network. After the simulation of the network the distribution of the output spikes was used to calculate the posterior probability. The posterior probabilities of both the mathematical analysis and the simulation were compared. By varying three network hyperparameters it was tuned to approximate the analytical solution as closely as possible.

4.2.2 Methods

Input data The input image consisted of nine pixels in a horizontal line. Within those nine pixels, going from position 0 to 8, four output classes could be represented. Each output class had of three pixels next to each other. This resulted in each output class overlapping its neighbour classes by one pixel. Thus the centers of the output classes were at position 1, 3, 5 and 7.

Network architecture Compared to the architecture of the previous experiment only the amount of neurons was changed. 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 had 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 had four output neurons.

Mathematical Analysis The posterior probability of the network was calculated by using Equation 3.3. $P(X = x|Y = k)$ and $P(Y = k|Z = j)$ were derived corresponding to the paradigm of the experiment. The calculation of $P(X = x|Y = k)$ 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 x 9 and contains the conditional probabilities of each input neuron y_k being active, given that an output neuron x_i 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 architecture. 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. According to this $P^{X|Y}$ is given by

$$P_{k,i}^{X|Y} = \begin{cases} 1-\text{noise level} & \text{if } x_i \text{ is in the active area of } y_k, \\ 0+\text{noise level} & \text{if } x_i \text{ is not in the active area of } y_k \end{cases} . \tag{4.1}$$

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

$$P_1(X = x|Y = k) = P_{k,*}^{X|Y} \cdot x \tag{4.2}$$

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

$$P_2(X = x|Y = k) = (1 - P^{X|Y}_{k,*}) \cdot (1 - x). \tag{4.3}$$

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

$$P(X = x|Y = k) = P_1(X = x|Y = k) \odot P_2(X = x|Y = k). \tag{4.4}$$

$P(Y = k|Z = j)$ was determined by first calculating the matrix $P^{Y|Z}$. It has a dimension of 4 x 4 and contains the conditional probabilities for the output neuron y_k, 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

$$P^{Y|Z}_{k,j} = \begin{cases} 1-\text{noise level} & \text{if } i = j, \\ 0 + \frac{1}{3}\text{noise level} & \text{if } i \neq j. \end{cases} \tag{4.5}$$

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

$$P(Y = k|Z = j) = P^{Y|Z} \cdot z \tag{4.6}$$

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

Simulation The input weights for the simulation were calculated as two separate sets. First weights w^{I1} for the input neurons that are active for active input pixels were determined by

$$w^{I1} = \ln(P^{X|Y}). \tag{4.7}$$

Second complementary input weights w^{I2} were calculated for input neurons representing non active input pixels with

$$w^{I2} = \ln(1 - P^{X|Y}). \tag{4.8}$$

The prior weights w^P were derived by

$$w^P = \ln(P^{Y|Z}). \tag{4.9}$$

The network was simulated for six different input images for each hyperparameter set. These six images were hand-picked to include specific edge cases and to allow comparability of the results of different hyperparameter sets. After the image presentation period of each input image the numbers of output spikes of each class were counted and their proportions were calculated to yield the posterior $P_{simulation}(Y = k|X, Z)$. The Kullback-Leibler divergence was chosen to compare the divergence of the analytic and the simulated posteriors of the network. This metric indicates how much two probability distributions diverge from each other. The goal of the hyperparameter search of the simulation was to minimize the Kullback-Leibler divergence. After showing an output image to the network the Kullback-Leibler divergence was calculated as

$$\begin{aligned} &D_{KL}(P_{analysis}(Y = k|X, Z)||P_{simulation}(Y = k|X, Z)) = \\ &\sum_{k=1}^{K} P_{analysis}(Y = k|X, Z) \cdot \ln(\frac{P_{analysis}(Y = k|X, Z)}{P_{simulation}(Y = k|X, Z)}) \end{aligned} \tag{4.10}$$

where K is the number of output neurons and $P_{analysis}(Y = k|X, Z)$ and $P_{simulation}(Y = k|X, Z)$ are the posteriors of the network, later called "analysis output probabilities" and "simulation output probabilities" in plots. The six resulting Kullback-Leibler divergences for each hyperparameter set were averaged to create a single metric by which the performance of the different hyperparameter sets was compared. The three hyperparameters 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 as to approximate the analytical solution as closely as possible. Each input image was presented to the network for 20 seconds to reduce the variance between runs. Furthermore each simulation was repeated 20 times with the same hyperparameter set to obtain the mean and standard deviation of the simulation output probabilities and of the Kullback-Leibler divergence.

4.2.3 Results

Analytic Results First the matrix $P^{X|Y}$ was calculated using Equation 4.1

$$P^{X|Y} = \begin{bmatrix} 0.9 & 0.9 & 0.9 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.9 & 0.9 & 0.9 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.9 & 0.9 & 0.9 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.9 & 0.9 & 0.9 \end{bmatrix}. \tag{4.11}$$

Next the matrix $P^{Y|Z}$ was calculated using Equation 4.5

$$P^{Y|Z} = \begin{bmatrix} 0.9 & 0.0333 & 0.0333 & 0.0333 \\ 0.0333 & 0.9 & 0.0333 & 0.0333 \\ 0.0333 & 0.0333 & 0.9 & 0.0333 \\ 0.0333 & 0.0333 & 0.0333 & 0.9 \end{bmatrix}. \tag{4.12}$$

For each input image these two matrices were multiplied with the input vector and prior vector as described Equations 4.2, 4.3, 4.4 and 4.6 to yield $P(X = x|Y = i)$ and $P(Y = i|Z = j)$. Using 3.3 finally yielded the analytical $P(Y = i|X = x, Z = j)$.

Simulation results with prior disabled To simplify the hyperparameter fitting process the network was at first simulated with inactive prior neurons. Only after determining the best values for f_{input} and τ_{decay} the prior neurons were 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 looking for the value that yielded the smallest Kullback-Leibler divergence.

$\tau_{decay} = 0.015 seconds$, $f_{prior} = 0 Hz$ 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 $\pm$ 10 Hz in steps of 2 Hz. The result of this search can be seen in Figure 4.9. This process yielded the best input firing rate of 42 Hz. The results of this hyperparameter combination can be seen in Figure 4.10. In this figure six different input images can be seen in A. The active pixels

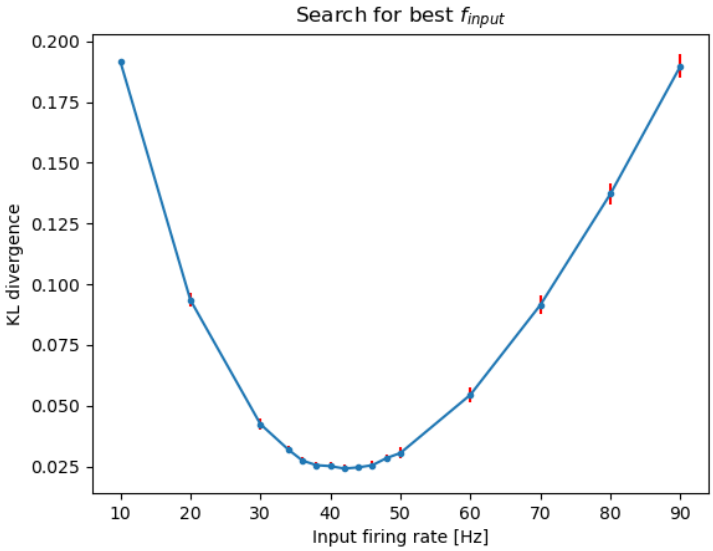

Figure 4.9: **KL divergence for different f_{input} values.** Hyperparameters: $f_{prior} = 0Hz, \tau_{decay} = 15ms$

are shown in black. In B the posterior probabilities of the mathematical analysis are given by the red bars and the posterior probabilities of the simulation are given by the blue bars. Each simulation output probability has its standard deviation marked by a black bar. At the bottom the value of the Kullback-Leibler divergence is given, with its standard deviation. The exact numeric probabilities are given in Table 4.1.

When f_{input} was 70 Hz the results of the analysis and the simulation differed more as can be seen in Figure 4.11 and Table 4.2.

$\tau_{decay} = 0.004seconds$, $f_{prior} = 0Hz$ For this hyperparameter combination values between 50 and 150 Hz for f_{input} in steps of 10 Hz were simulated. Analogously to the previous hyperparameter set after finding the best input firing rate the search was performed in finer steps until the best value was found at 88 Hz. The result of this search can be seen in Figure 4.12. The result of the simulation with those hyperparameters can be seen in Figure 4.13 and Table 4.3.

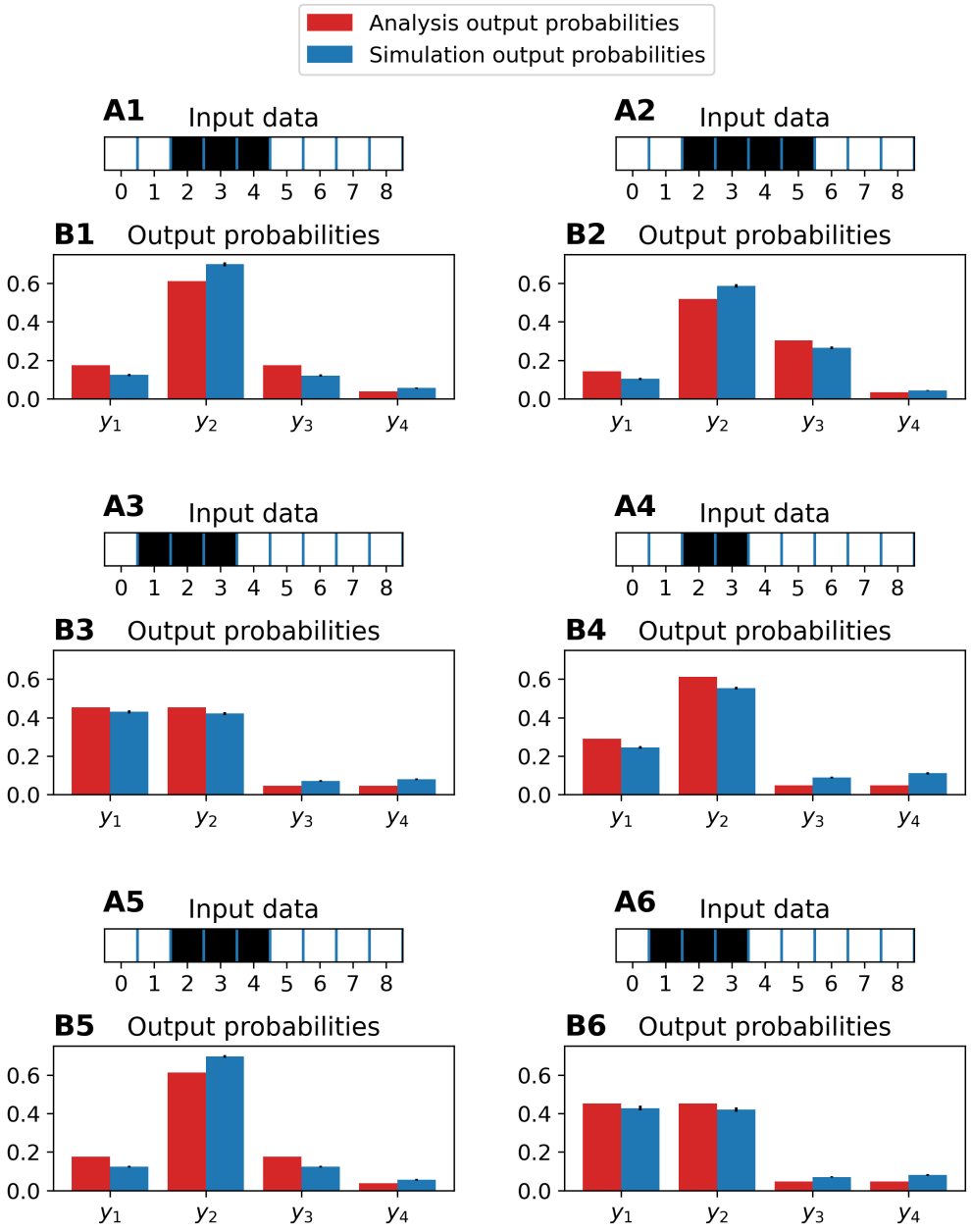

Figure 4.10: **Analysis and simulation result.** Hyperparameters: $f_{input} = 42Hz, f_{prior} = 0Hz, \tau_{decay} = 15ms$ **A** Input images with 9 x 1 pixels. **B** Analytically calculated posterior probabilities and simulated posterior probabilities. The standard deviations are given by the black bars.

	Image 1		Image 2	
	Analysis	Simulation	Analysis	Simulation
y_0	0.175	0.124 ± 0.0061	0.143	0.104 ± 0.0059
y_1	0.612	0.699 ± 0.0093	0.518	0.587 ± 0.0090
y_2	0.175	0.121 ± 0.0058	0.304	0.266 ± 0.0067
y_3	0.038	0.056 ± 0.0033	0.035	0.043 ± 0.0025
	Image 3		Image 4	
y_0	0.453	0.430 ± 0.0089	0.291	0.245 ± 0.0065
y_1	0.453	0.421 ± 0.0073	0.613	0.553 ± 0.0072
y_2	0.047	0.070 ± 0.0045	0.048	0.090 ± 0.0045
y_3	0.047	0.080 ± 0.0042	0.048	0.111 ± 0.0053
	Image 5		Image 6	
y_0	0.175	0.124 ± 0.0046	0.453	0.428 ± 0.0116
y_1	0.612	0.697 ± 0.0069	0.453	0.421 ± 0.0111
y_2	0.175	0.123 ± 0.0050	0.047	0.070 ± 0.0034
y_3	0.038	0.056 ± 0.0043	0.047	0.081 ± 0.0047

Table 4.1: **Analysis and simulation output probabilities.** Hyperparameters: $f_{input} = 42Hz, f_{prior} = 0Hz, \tau_{decay} = 15ms$

	Image 1		Image 2	
	Analysis	Simulation	Analysis	Simulation
y_0	0.175	0.061 ± 0.0038	0.143	0.049 ± 0.0027
y_1	0.612	0.861 ± 0.0065	0.518	0.715 ± 0.0083
y_2	0.175	0.061 ± 0.0041	0.304	0.226 ± 0.0070
y_3	0.038	0.017 ± 0.0016	0.035	0.011 ± 0.0017
	Image 3		Image 4	
y_0	0.453	0.472 ± 0.0100	0.291	0.207 ± 0.0067
y_1	0.453	0.464 ± 0.0101	0.613	0.685 ± 0.0080
y_2	0.047	0.027 ± 0.0027	0.048	0.043 ± 0.0037
y_3	0.047	0.037 ± 0.0032	0.048	0.065 ± 0.0042
	Image 5		Image 6	
y_0	0.175	0.061 ± 0.0033	0.453	0.475 ± 0.0086
y_1	0.612	0.864 ± 0.0062	0.453	0.459 ± 0.0087
y_2	0.175	0.057 ± 0.0036	0.047	0.028 ± 0.0018
y_3	0.038	0.018 ± 0.0022	0.047	0.038 ± 0.0032

Table 4.2: **Analysis and simulation output probabilities.** Hyperparameters: $f_{input} = 70Hz, f_{prior} = 0Hz, \tau_{decay} = 15ms$

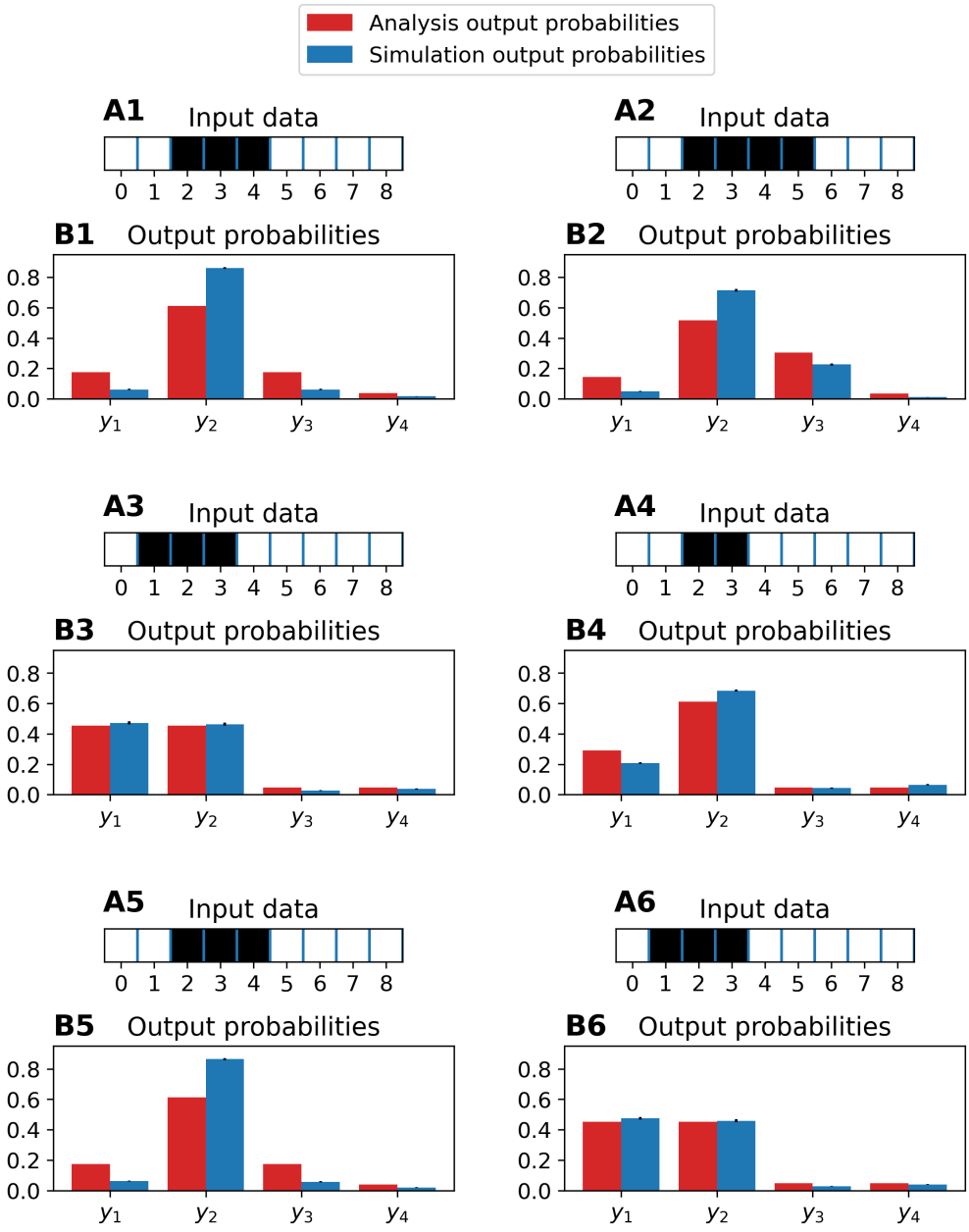

Figure 4.11: **Analysis and simulation result.** Hyperparameters: $f_{input} = 70Hz, f_{prior} = 0Hz, \tau_{decay} = 15ms$ **A** Input images with 9 x 1 pixels. **B** Analytically calculated posterior probabilities and simulated posterior probabilities. The standard deviations are given by the black bars.

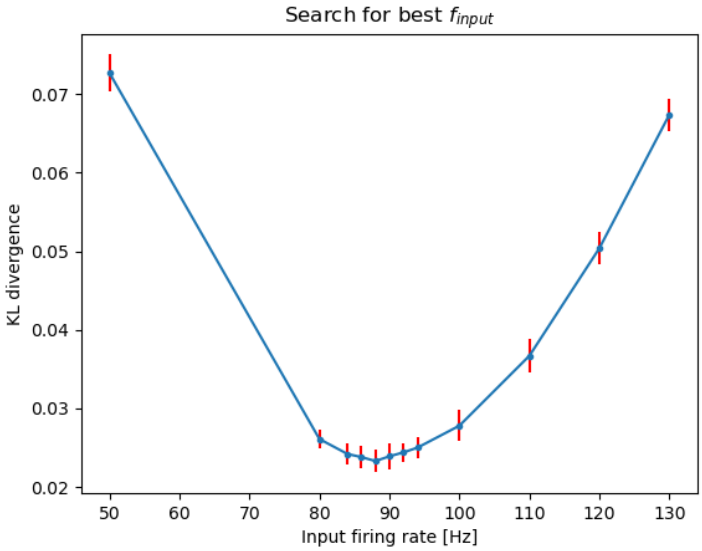

Figure 4.12: **KL divergence for different f_{input} values.** Hyperparameters: $f_{prior} = 0Hz, \tau_{decay} = 4ms$

	Image 1		Image 2	
	Analysis	Simulation	Analysis	Simulation
y_0	0.175	0.126 ± 0.0050	0.143	0.106 ± 0.0054
y_1	0.612	0.695 ± 0.0066	0.518	0.587 ± 0.0098
y_2	0.175	0.124 ± 0.0049	0.304	0.264 ± 0.0072
y_3	0.038	0.054 ± 0.0039	0.035	0.043 ± 0.0028
	Image 3		Image 4	
y_0	0.453	0.429 ± 0.0080	0.291	0.240 ± 0.0044
y_1	0.453	0.421 ± 0.0083	0.613	0.556 ± 0.0084
y_2	0.047	0.070 ± 0.0037	0.048	0.095 ± 0.0036
y_3	0.047	0.080 ± 0.0044	0.048	0.109 ± 0.0058
	Image 5		Image 6	
y_0	0.175	0.125 ± 0.0044	0.453	0.426 ± 0.0085
y_1	0.612	0.697 ± 0.0072	0.453	0.424 ± 0.0068
y_2	0.175	0.124 ± 0.0038	0.047	0.072 ± 0.0031
y_3	0.038	0.054 ± 0.0032	0.047	0.078 ± 0.0047

Table 4.3: **Analysis and simulation output probabilities.** Hyperparameters: $f_{input} = 88Hz, f_{prior} = 0Hz, \tau_{decay} = 4ms$

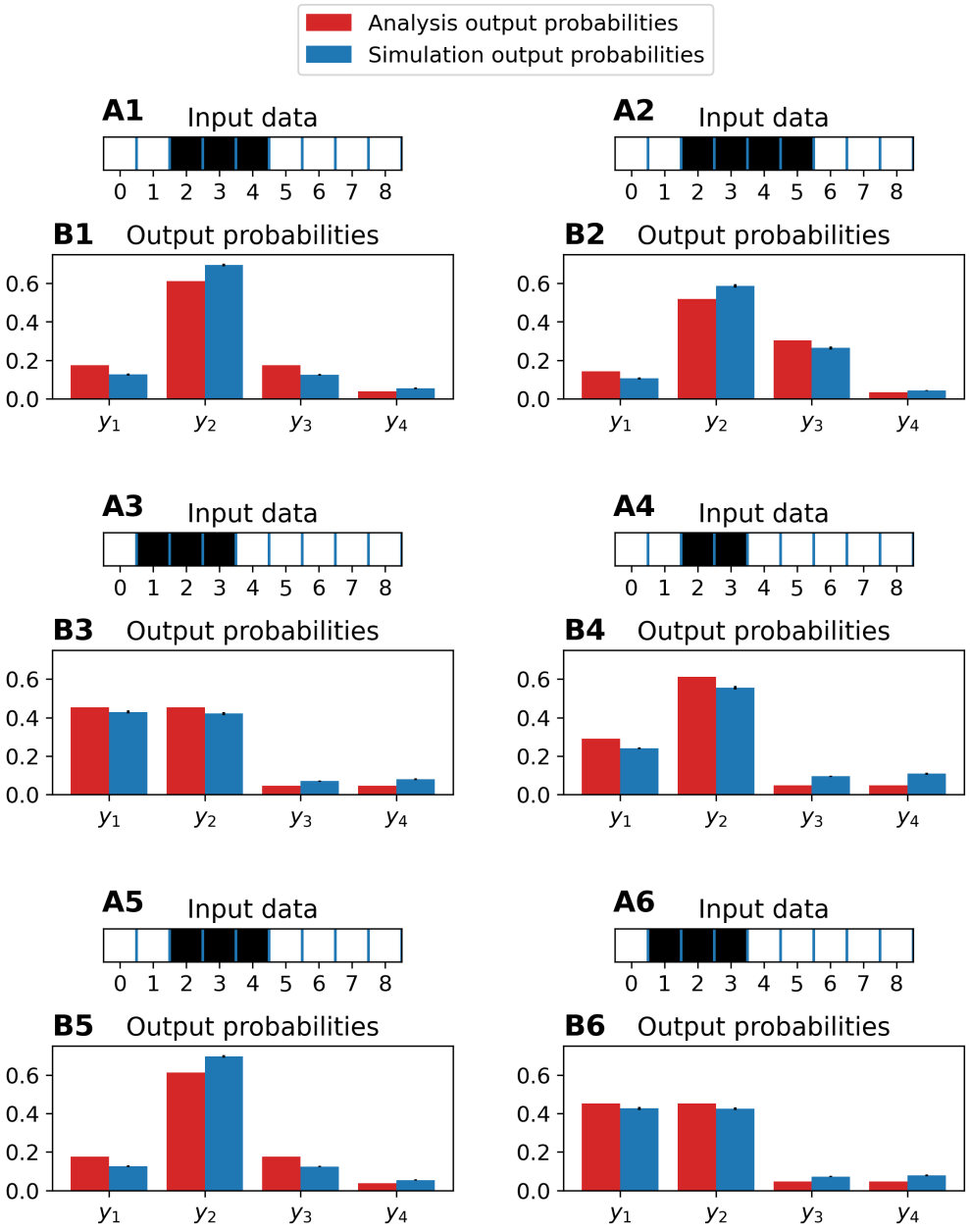

Figure 4.13: **Analysis and simulation result.** Hyperparameters: $f_{input} = 88Hz, f_{prior} = 0Hz, \tau_{decay} = 4ms$ **A** Input images with 9 x 1 pixels. **B** Analytically calculated posterior probabilities and simulated posterior probabilities. The standard deviations are given by the black bars.

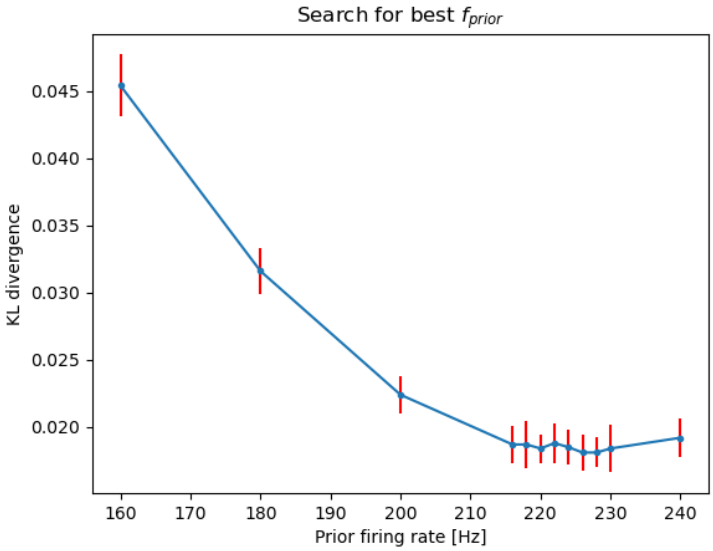

Figure 4.14: **KL divergence for different f_{prior} values.** Hyperparameters: $f_{input} = 42Hz, \tau_{decay} = 15ms$

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

$\tau_{decay} = 0.015seconds$, $f_{input} = 42Hz$ 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 4.14. The simulation results can be seen in Figure 4.15 and Table 4.4. In Figure 4.15 the value of the prior is indicated by the red border which is three pixels wide and centered at the center position of the corresponding output class.

$\tau_{decay} = 0.004seconds$, $f_{input} = 88Hz$ 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

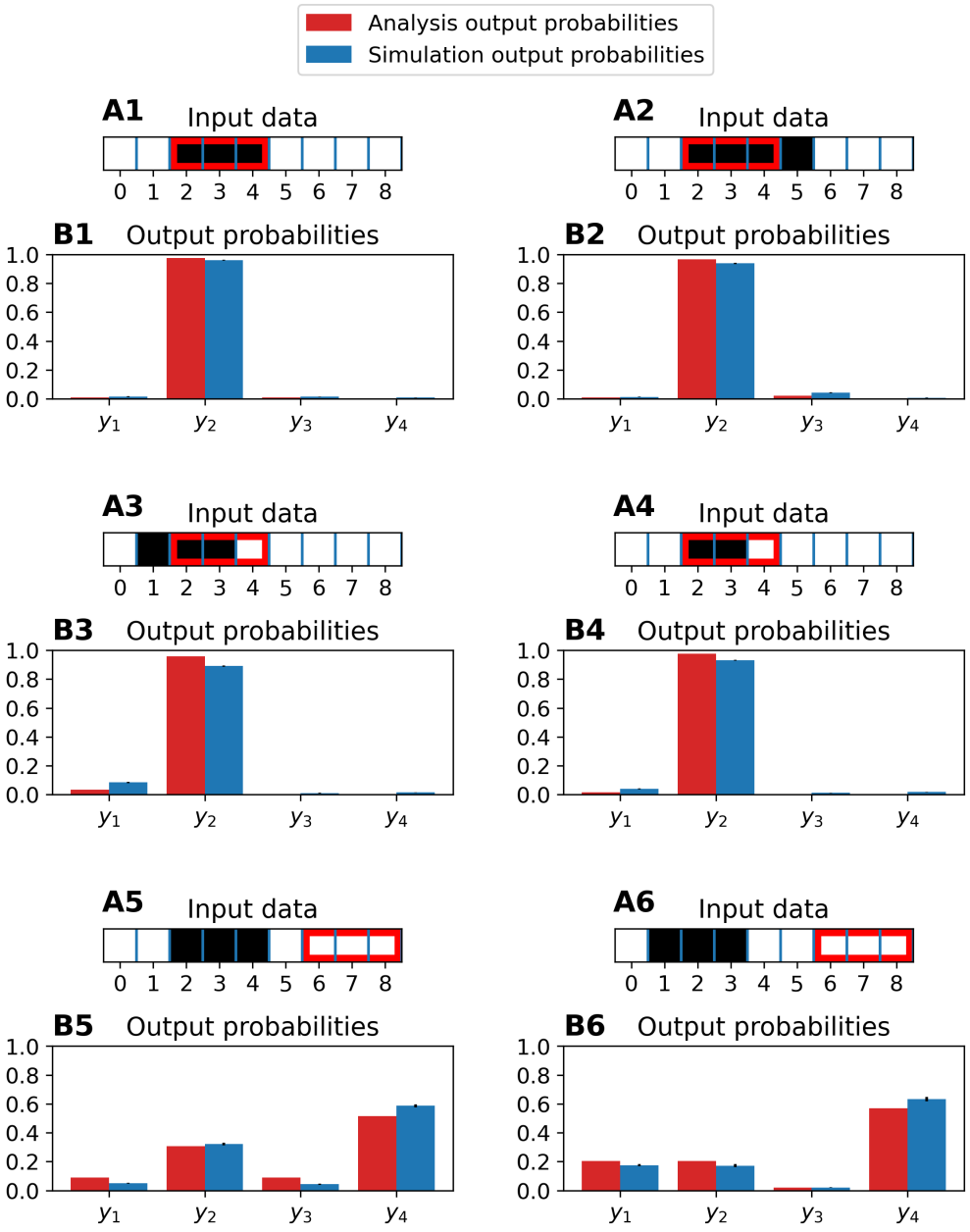

Figure 4.15: **Analysis and simulation result.** Hyperparameters: $f_{input} = 42Hz, f_{prior} = 222Hz, \tau_{decay} = 15ms$ **A** Input images with 9 x 1 pixels. Prior given as red border. **B** Analytically calculated posterior probabilities and simulated posterior probabilities. The standard deviations are given by the black bars.

	Image 1		Image 2	
	Analysis	Simulation	Analysis	Simulation
y_0	0.010	0.015 ± 0.0021	0.010	0.013 ± 0.0016
y_1	0.977	0.962 ± 0.0030	0.967	0.938 ± 0.0040
y_2	0.010	0.015 ± 0.0017	0.021	0.042 ± 0.0032
y_3	0.002	0.008 ± 0.0016	0.002	0.007 ± 0.0013
	Image 3		Image 4	
y_0	0.035	0.084 ± 0.0038	0.017	0.039 ± 0.0033
y_1	0.957	0.891 ± 0.0046	0.977	0.931 ± 0.0045
y_2	0.004	0.010 ± 0.0018	0.003	0.012 ± 0.0017
y_3	0.004	0.015 ± 0.0016	0.003	0.018 ± 0.0024
	Image 5		Image 6	
y_0	0.088	0.050 ± 0.0033	0.205	0.175 ± 0.0077
y_1	0.308	0.321 ± 0.0087	0.205	0.172 ± 0.0106
y_2	0.088	0.043 ± 0.0039	0.021	0.021 ± 0.0018
y_3	0.515	0.587 ± 0.0105	0.570	0.632 ± 0.0158

Table 4.4: **Analysis and simulation output probabilities.** Hyperparameters: $f_{input} = 42Hz, f_{prior} = 222Hz, \tau_{decay} = 15ms$

best. The result of this search can be seen in Figure 4.16. The results of this hyperparameter combination are given in Figure 4.17 and Table 4.5.

To demonstrate the impact of rising f_{prior} it was set to 600 Hz and the results can be seen in Figure 4.18 and Table 4.6.

$\tau_{decay} = 0.004seconds$, $f_{input} = 98$, $f_{prior} = 440Hz$ Finally, 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 4.19. The best result was obtained with an input firing rate of 98 Hz. The results of that simulation can be seen in Figure 4.20 and Table 4.7.

Approximating the analytic solution After first finding the optimal f_{input} for disabled prior activity and then finding the optimal f_{prior} with enabled

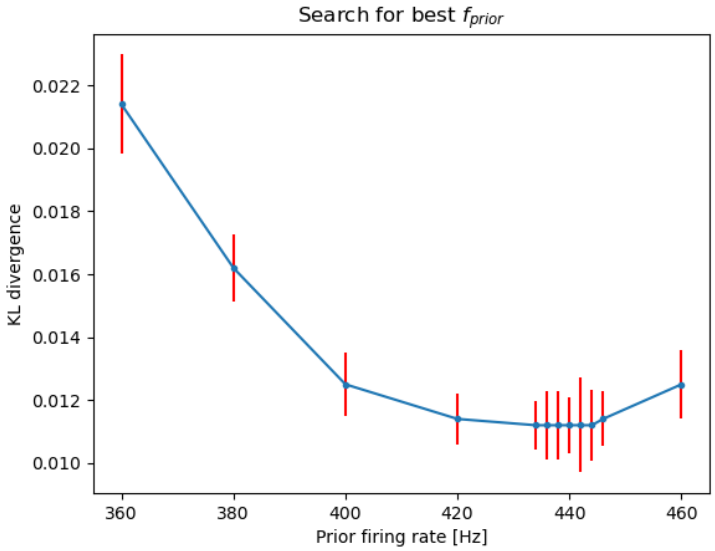

Figure 4.16: **KL divergence for different f_{prior} values.** Hyperparameters: $f_{input} = 88Hz, \tau_{decay} = 4ms$

	Image 1		Image 2	
	Analysis	Simulation	Analysis	Simulation
y_0	0.010	0.011 ± 0.0016	0.010	0.010 ± 0.0017
y_1	0.977	0.974 ± 0.0026	0.967	0.957 ± 0.0034
y_2	0.010	0.010 ± 0.0019	0.021	0.028 ± 0.0036
y_3	0.002	0.005 ± 0.0013	0.002	0.005 ± 0.0011
	Image 3		Image 4	
y_0	0.035	0.066 ± 0.0031	0.017	0.026 ± 0.0030
y_1	0.957	0.915 ± 0.0044	0.977	0.952 ± 0.0039
y_2	0.004	0.009 ± 0.0014	0.003	0.009 ± 0.0017
y_3	0.004	0.011 ± 0.0018	0.003	0.012 ± 0.0020
	Image 5		Image 6	
y_0	0.088	0.052 ± 0.0027	0.205	0.172 ± 0.0061
y_1	0.308	0.328 ± 0.0085	0.205	0.174 ± 0.0059
y_2	0.088	0.046 ± 0.0031	0.021	0.025 ± 0.0025
y_3	0.515	0.573 ± 0.0089	0.570	0.630 ± 0.0095

Table 4.5: **Analysis and simulation output probabilities.** Hyperparameters: $f_{input} = 88Hz, f_{prior} = 440Hz, \tau_{decay} = 4ms$

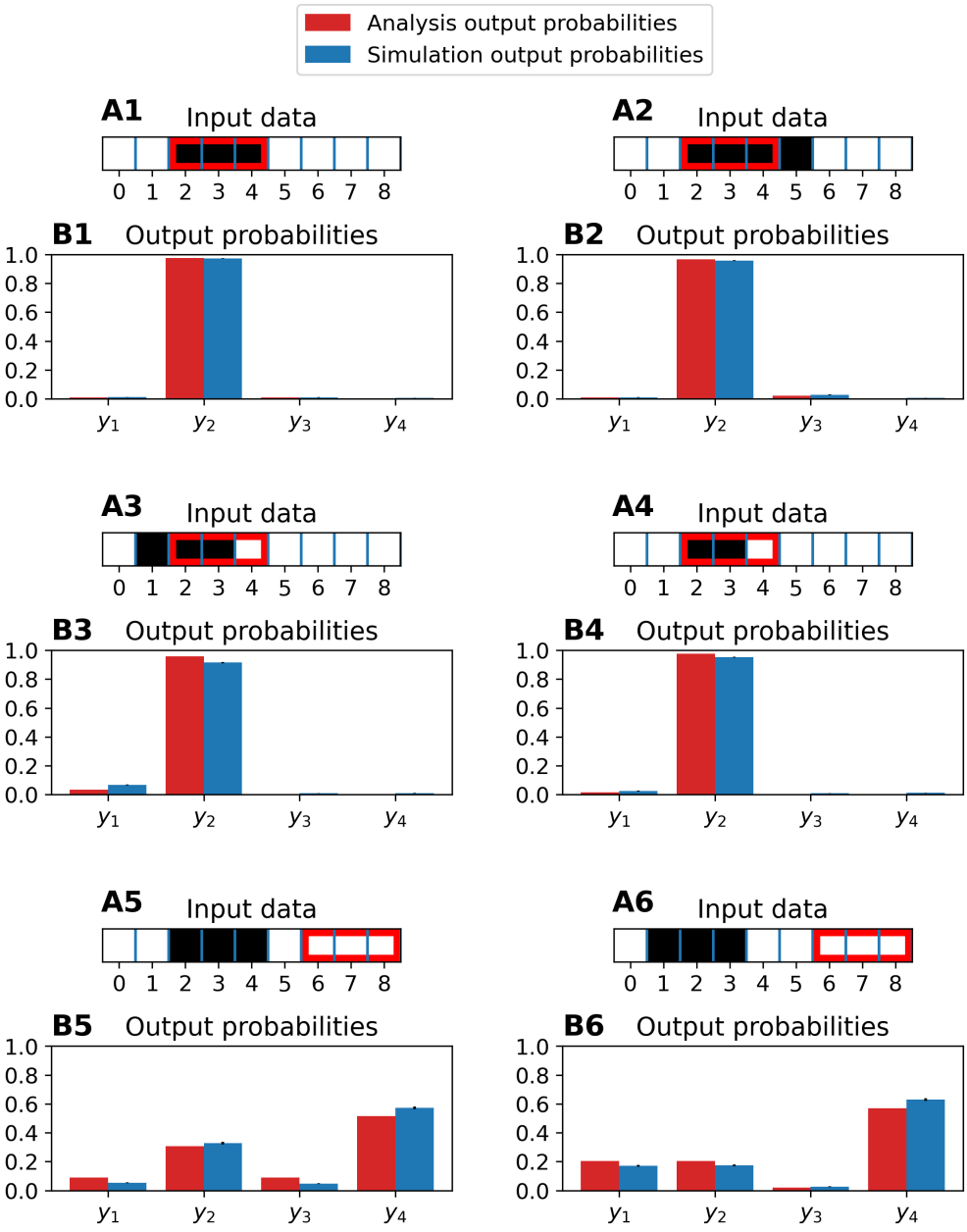

Figure 4.17: **Analysis and simulation result.** Hyperparameters: $f_{input} = 88Hz, f_{prior} = 440Hz, \tau_{decay} = 4ms$ **A** Input images with 9 x 1 pixels. Prior given as red border. **B** Analytically calculated posterior probabilities and simulated posterior probabilities. The standard deviations are given by the black bars.

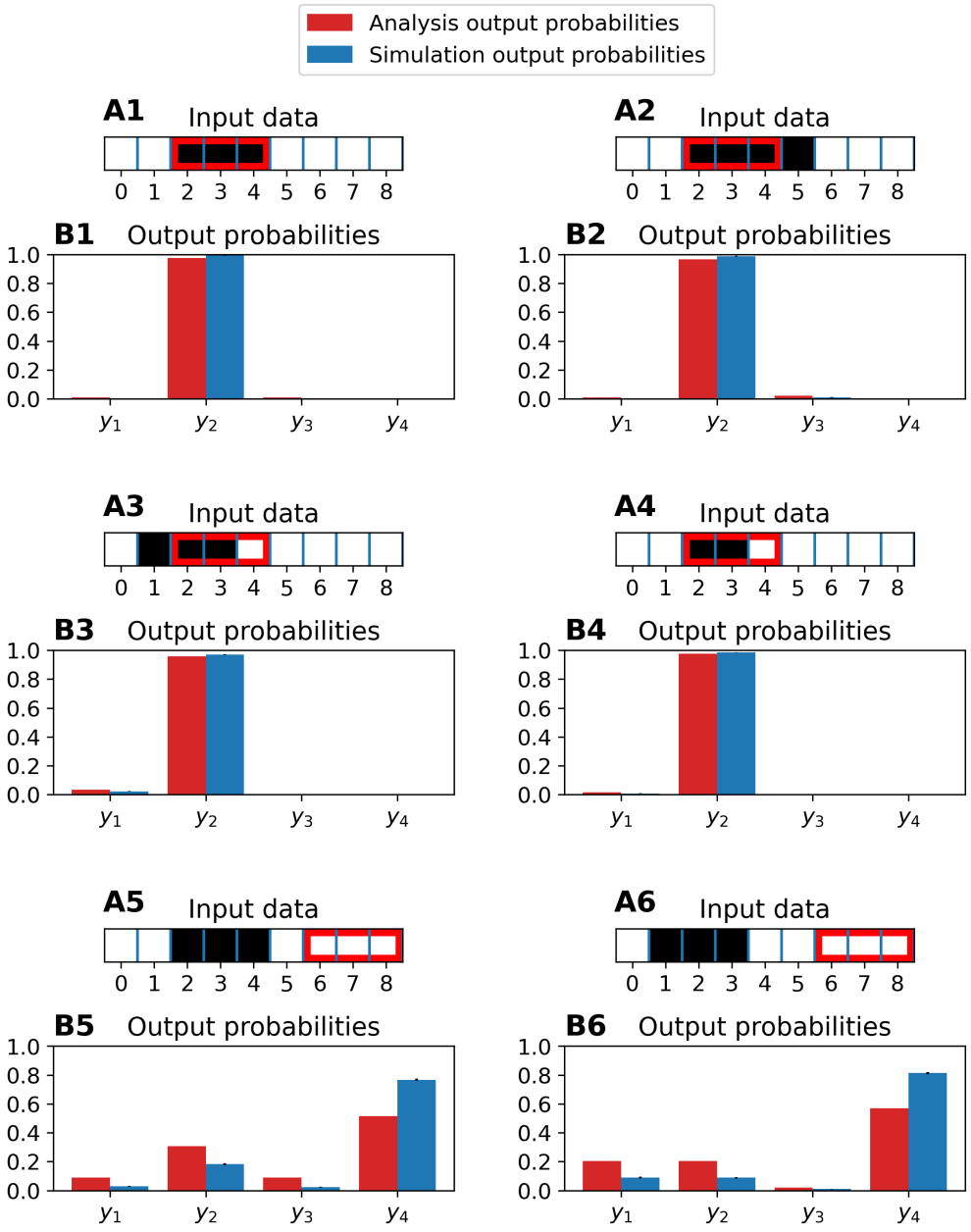

Figure 4.18: **Analysis and simulation result.** Hyperparameters: $f_{input} = 88Hz, f_{prior} = 600Hz, \tau_{decay} = 4ms$ **A** Input images with 9 x 1 pixels. Prior given as red border. **B** Analytically calculated posterior probabilities and simulated posterior probabilities. The standard deviations are given by the black bars.

	Image 1		Image 2	
	Analysis	Simulation	Analysis	Simulation
y_0	0.010	0.003 ± 0.0011	0.010	0.003 ± 0.0009
y_1	0.977	0.993 ± 0.0013	0.967	0.987 ± 0.0024
y_2	0.010	0.003 ± 0.0008	0.021	0.009 ± 0.0019
y_3	0.002	0.001 ± 0.0005	0.002	0.001 ± 0.0008
	Image 3		Image 4	
y_0	0.035	0.023 ± 0.0025	0.017	0.008 ± 0.0011
y_1	0.957	0.971 ± 0.0030	0.977	0.984 ± 0.0019
y_2	0.004	0.003 ± 0.0009	0.003	0.003 ± 0.0008
y_3	0.004	0.003 ± 0.0011	0.003	0.004 ± 0.0011
	Image 5		Image 6	
y_0	0.088	0.028 ± 0.0026	0.205	0.089 ± 0.0057
y_1	0.308	0.182 ± 0.0077	0.205	0.088 ± 0.0047
y_2	0.088	0.023 ± 0.0025	0.021	0.010 ± 0.0016
y_3	0.515	0.766 ± 0.0079	0.570	0.813 ± 0.0081

Table 4.6: **Analysis and simulation output probabilities.** Hyperparameters: $f_{input} = 88Hz, f_{prior} = 600Hz, \tau_{decay} = 4ms$

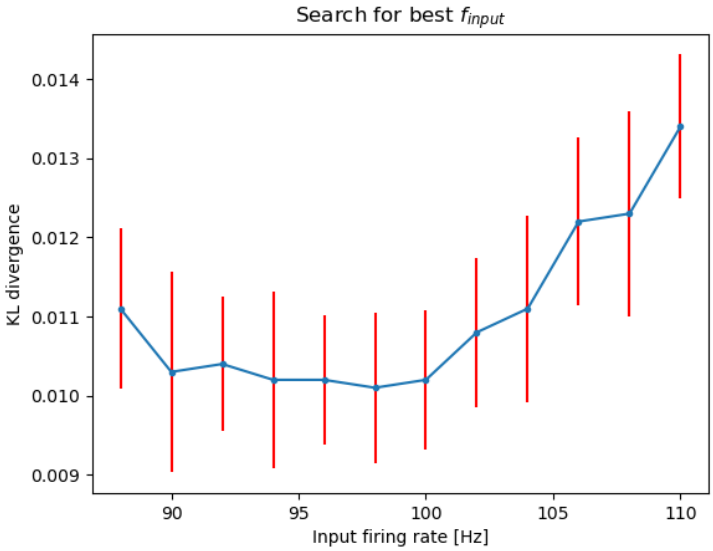

Figure 4.19: **KL divergence for different f_{input} values.** Hyperparameters: $f_{prior} = 440Hz, \tau_{decay} = 4ms$

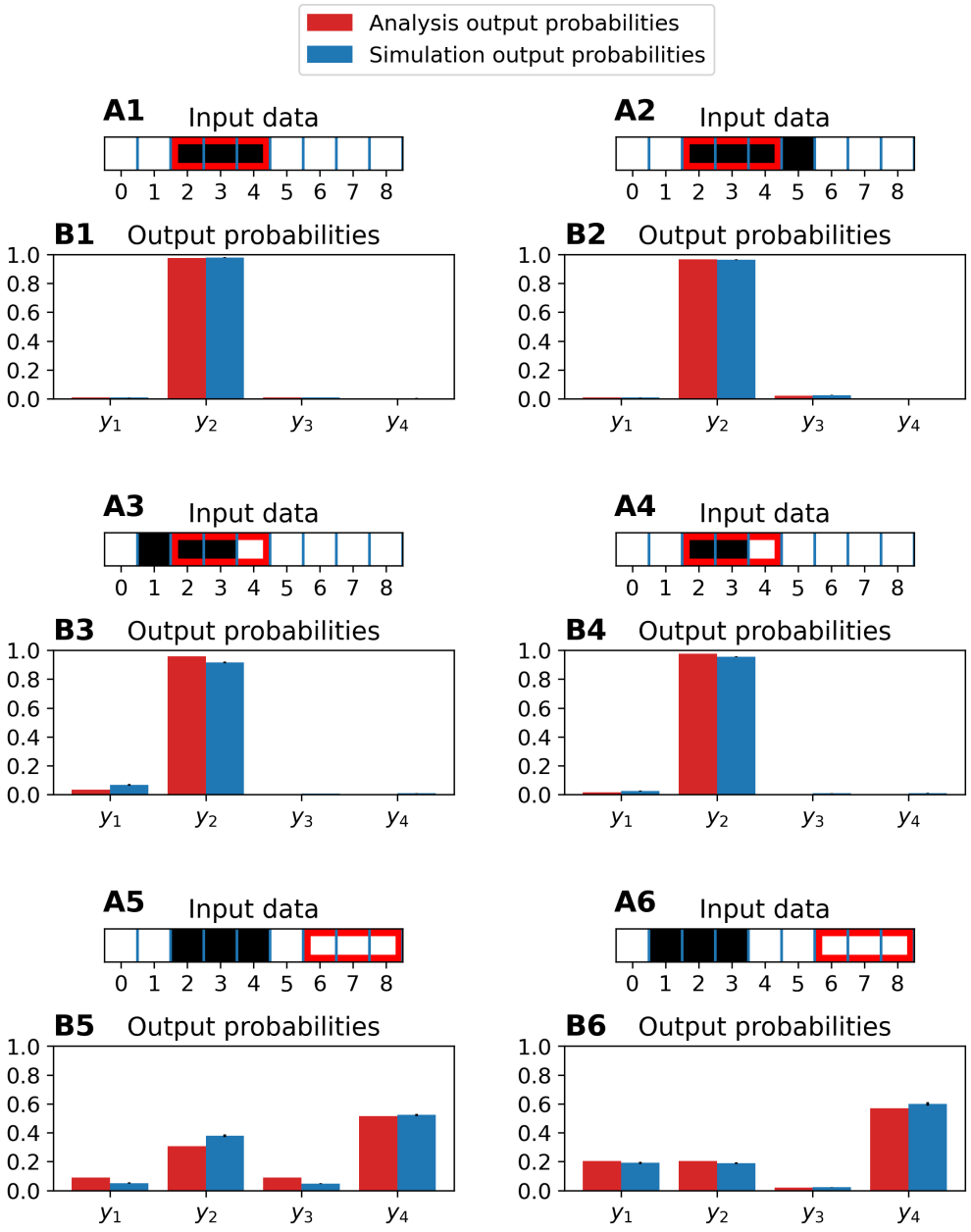

Figure 4.20: **Analysis and simulation result.** Hyperparameters: $f_{input} = 98Hz, f_{prior} = 440Hz, \tau_{decay} = 4ms$ **A** Input images with 9 x 1 pixels. Prior given as red border. **B** Analytically calculated posterior probabilities and simulated posterior probabilities. The standard deviations are given by the black bars.

	Image 1		Image 2	
	Analysis	Simulation	Analysis	Simulation
y_0	0.010	0.009 ± 0.0016	0.010	0.009 ± 0.0015
y_1	0.977	0.979 ± 0.0020	0.967	0.963 ± 0.0023
y_2	0.010	0.009 ± 0.0010	0.021	0.025 ± 0.0020
y_3	0.002	0.004 ± 0.0009	0.002	0.003 ± 0.0008
	Image 3		Image 4	
y_0	0.035	0.067 ± 0.0051	0.017	0.025 ± 0.0031
y_1	0.957	0.916 ± 0.0051	0.977	0.955 ± 0.0033
y_2	0.004	0.007 ± 0.0012	0.003	0.009 ± 0.0012
y_3	0.004	0.009 ± 0.0019	0.003	0.011 ± 0.0019
	Image 5		Image 6	
y_0	0.088	0.051 ± 0.0035	0.205	0.191 ± 0.0080
y_1	0.308	0.379 ± 0.0084	0.205	0.188 ± 0.0061
y_2	0.088	0.046 ± 0.0027	0.021	0.022 ± 0.0021
y_3	0.515	0.523 ± 0.0084	0.570	0.600 ± 0.0116

Table 4.7: **Analysis and simulation output probabilities.** Hyperparameters: $f_{input} = 98Hz$, $f_{prior} = 440Hz$, $\tau_{decay} = 4ms$

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 $\tau_{decay} = 0.004ms$ and $f_{prior} = 440Hz$, as this set of hyperparameters performed the best overall. When comparing the former optimal result in Figure 4.17 with $f_{input} = 88Hz$, to the result with $f_{input} = 98Hz$ in Figure 4.20, it can be seen that the Kullback-Leibler divergence decreased from 0.0112 to 0.0101. However, when comparing Tables 4.5 and 4.7, it can be seen that the higher f_{input} did not perform strictly better. The simulation output probabilities for Image 3 improved, while for Image 5 the output probability of y_2 diverged more. A further increase of f_{input} started to increase the Kullback-Leibler divergence again, thus the final optimum of the hyperparameter set was reached. It was assumed that increasing f_{input} after f_{prior} was fitted was necessary because the overall impact of f_{input} on the simulation decreased due to the introduction of the prior activity.

4.3 Experiment 3: Transferability of hyperparameters

4.3.1 Introduction

In this experiment the optimal hyperparameters determined in Experiment 2 were used, while doubling the size of the network and input images of Experiment 2. The goal was to determine if the hyperparameters are universally applicable, regardless of the network size.

4.3.2 Methods

The number input neurons was doubled from 18 to 36 neurons. This resulted in input images with 18 pixels. To double the effect of the prior neurons f_{prior} was doubled to 880 Hz, while the amount of the prior neurons was kept the same with 4 prior neurons. Furthermore the matrix $P^{X|Y}$ was doubled in size by copying each value of the matrix to the right of itself. The rest of the methods were performed analogously to Experiment 2.

4.3.3 Results

The expanded matrix $P^{X|Y}$ had its columns doubled, resulting in a dimension of $[18 \times 4]$. This yielded

$$P^{X|Y} = \begin{bmatrix} 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 \end{bmatrix}. \tag{4.13}$$

The simulation result for $f_{input} = 98Hz$, $f_{prior} = 880Hz$ and $\tau_{decay} = 4ms$ can be seen in Figure 4.21 and Table 4.8. Initially the Kullback-Leibler divergence was infinite in this result, because some simulation output probabilities were

	Image 1		Image 2	
	Analysis	Simulation	Analysis	Simulation
y_0	0.010	0.000 ± 0.0001	0.010	0.000 ± 0.0001
y_1	0.977	1.000 ± 0.0002	0.967	1.000 ± 0.0003
y_2	0.010	0.000 ± 0.0001	0.021	0.000 ± 0.0002
y_3	0.002	0.000 ± 0.0000	0.002	0.000 ± 0.0001
	Image 3		Image 4	
y_0	0.035	0.003 ± 0.0010	0.017	0.000 ± 0.0004
y_1	0.957	0.997 ± 0.0010	0.977	1.000 ± 0.0004
y_2	0.004	0.000 ± 0.0001	0.003	0.000 ± 0.0001
y_3	0.004	0.000 ± 0.0001	0.003	0.000 ± 0.0001
	Image 5		Image 6	
y_0	0.088	0.007 ± 0.0010	0.205	0.106 ± 0.0060
y_1	0.308	0.369 ± 0.0096	0.205	0.109 ± 0.0042
y_2	0.088	0.005 ± 0.0010	0.021	0.001 ± 0.0004
y_3	0.515	0.619 ± 0.0098	0.570	0.783 ± 0.0091

Table 4.8: **Analysis and simulation output probabilities.** Hyperparameters: $f_{input} = 98Hz, f_{prior} = 880Hz, \tau_{decay} = 4ms$

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 0.2392.

4.4 Experiment 4: Training of the network with 1-D images and predetermined hyperparameters

4.4.1 Introduction

In this Experiment the network of Experiment 2 was used, but the weights of the network were not derived from the matrices $P^{X|Y}$ and $P^{Y|Z}$, but they were learned from the input images. This was done to analyse how well

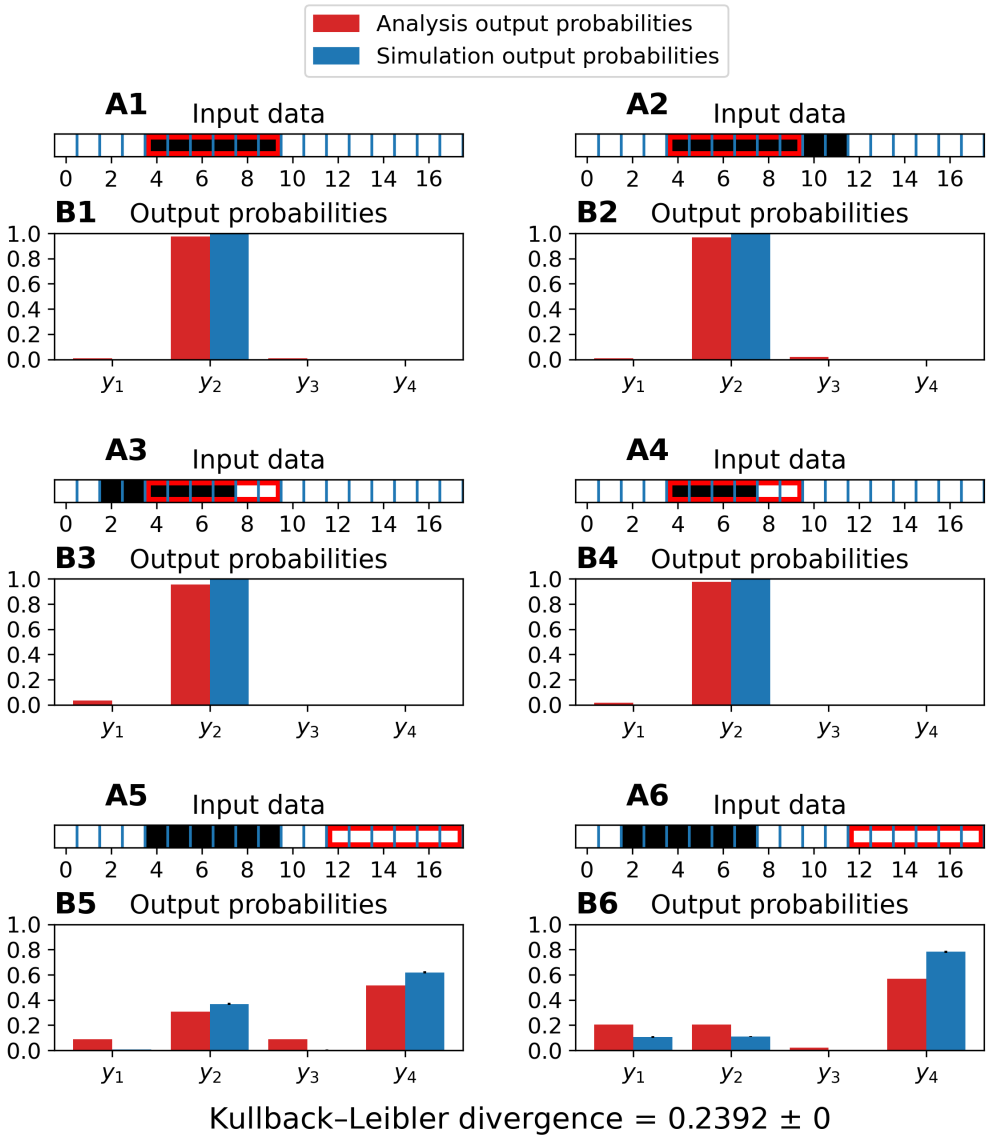

Figure 4.21: **Analysis and simulation result.** Hyperparameters: $f_{input} = 98Hz, f_{prior} = 880Hz, \tau_{decay} = 4ms$ **A** Input images with 18 x 1 pixels. Prior given as red border. **B** Analytically calculated posterior probabilities and simulated posterior probabilities. The standard deviations are given by the black bars.

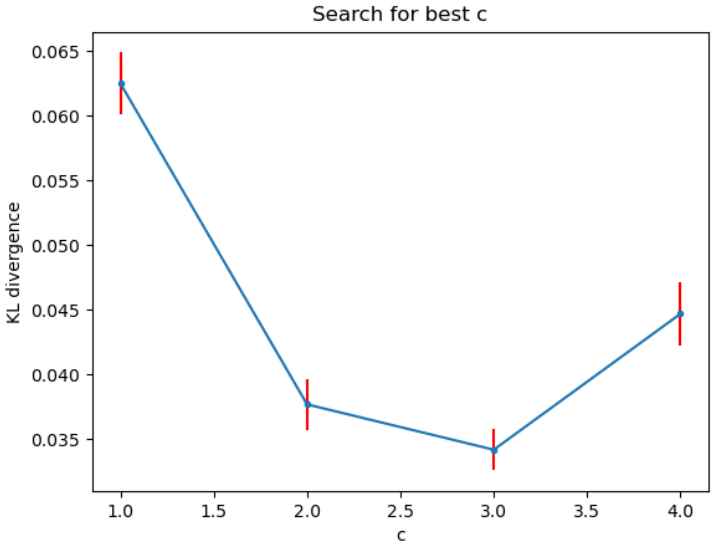

Figure 4.22: **KL divergence for different c values** $f_{input} = 98Hz, f_{prior} = 440Hz, \tau_{decay} = 4ms$

the optimal weights and the posterior could be approximated with the predetermined hyperparameters.

4.4.2 Methods

The network architecture was the same as in Experiment 2 and the training paradigm was the same as in Experiment 1. The used hyperparameters were $f_{input} = 98Hz$, $f_{prior} = 440Hz$ and $\tau_{decay} = 4ms$. The weight shifting hyperparameter c was determined via grid search.

4.4.3 Results

The Kullback-Leibler divergence for different values of c can be seen in Figure 4.22.

	Image 1		Image 2	
	Analysis	Simulation	Analysis	Simulation
y_0	0.010	0.004 ± 0.0010	0.010	0.005 ± 0.0009
y_1	0.977	0.988 ± 0.0016	0.967	0.977 ± 0.0028
y_2	0.010	0.005 ± 0.0011	0.021	0.014 ± 0.0.0018
y_3	0.002	0.003 ± 0.0008	0.002	0.004 ± 0.0012
	Image 3		Image 4	
y_0	0.035	0.035 ± 0.0026	0.017	0.012 ± 0.0018
y_1	0.957	0.953 ± 0.0035	0.977	0.977 ± 0.0025
y_2	0.004	0.005 ± 0.0010	0.003	0.004 ± 0.0009
y_3	0.004	0.006 ± 0.0014	0.003	0.006 ± 0.0014
	Image 5		Image 6	
y_0	0.088	0.039 ± 0.0027	0.205	0.144 ± 0.0053
y_1	0.308	0.202 ± 0.0070	0.205	0.099 ± 0.0049
y_2	0.088	0.032 ± 0.0023	0.021	0.019 ± 0.0021
y_3	0.515	0.727 ± 0.0063	0.570	0.738 ± 0.0074

Table 4.9: **Analysis and simulation output probabilities.** Hyperparameters: $f_{input} = 98Hz, f_{prior} = 440Hz, \tau_{decay} = 4ms, c = 3$

The training and evaluation results for the best value of $c = 3$ can be seen in Figures 4.23, Figure 4.24 and Table 4.9.

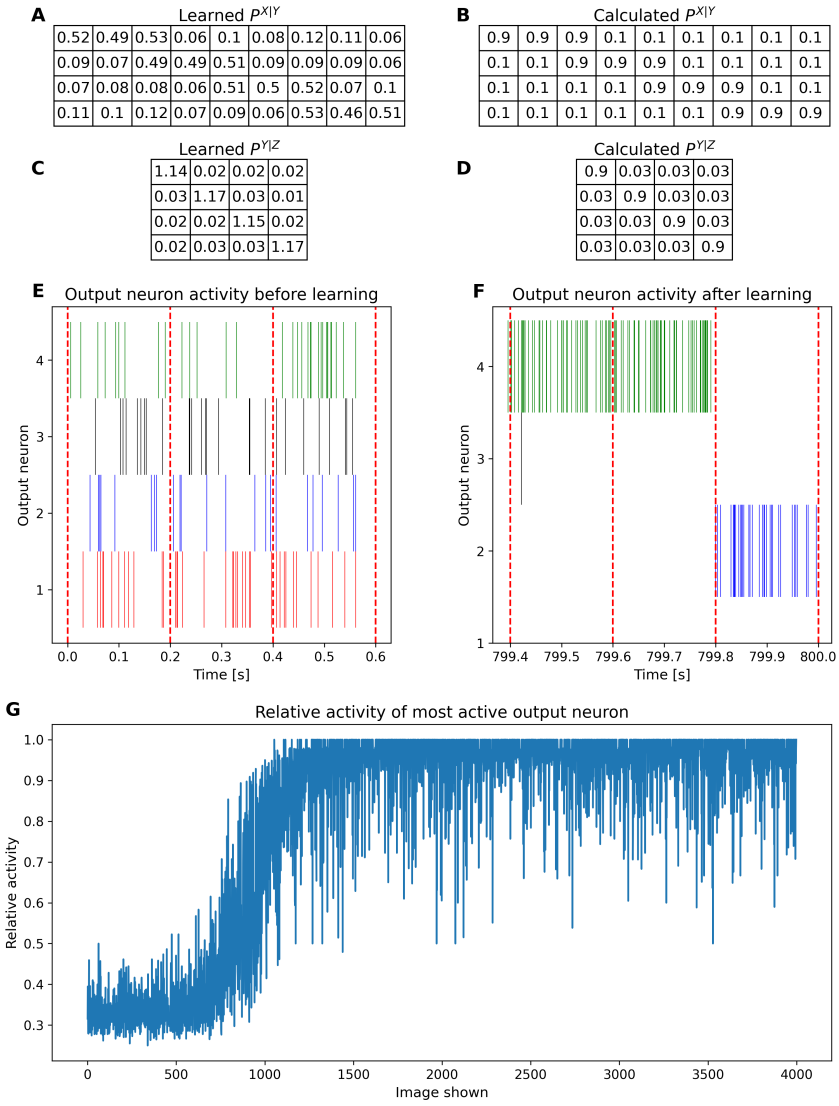

Figure 4.23: **Training result.** Hyperparameters: $f_{input} = 98Hz, f_{prior} = 440Hz, \tau_{decay} = 4ms, c = 3$ **A** The learned probability matrix $P^{X|Y}$. It was determined by taking the weights to the power of e. **B** The calculated probability matrix $P^{X|Y}$. **C** The learned prior probability 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** This shows the number of spikes the most active output neuron generated, relative to the number of spikes all other output neurons generated combined.

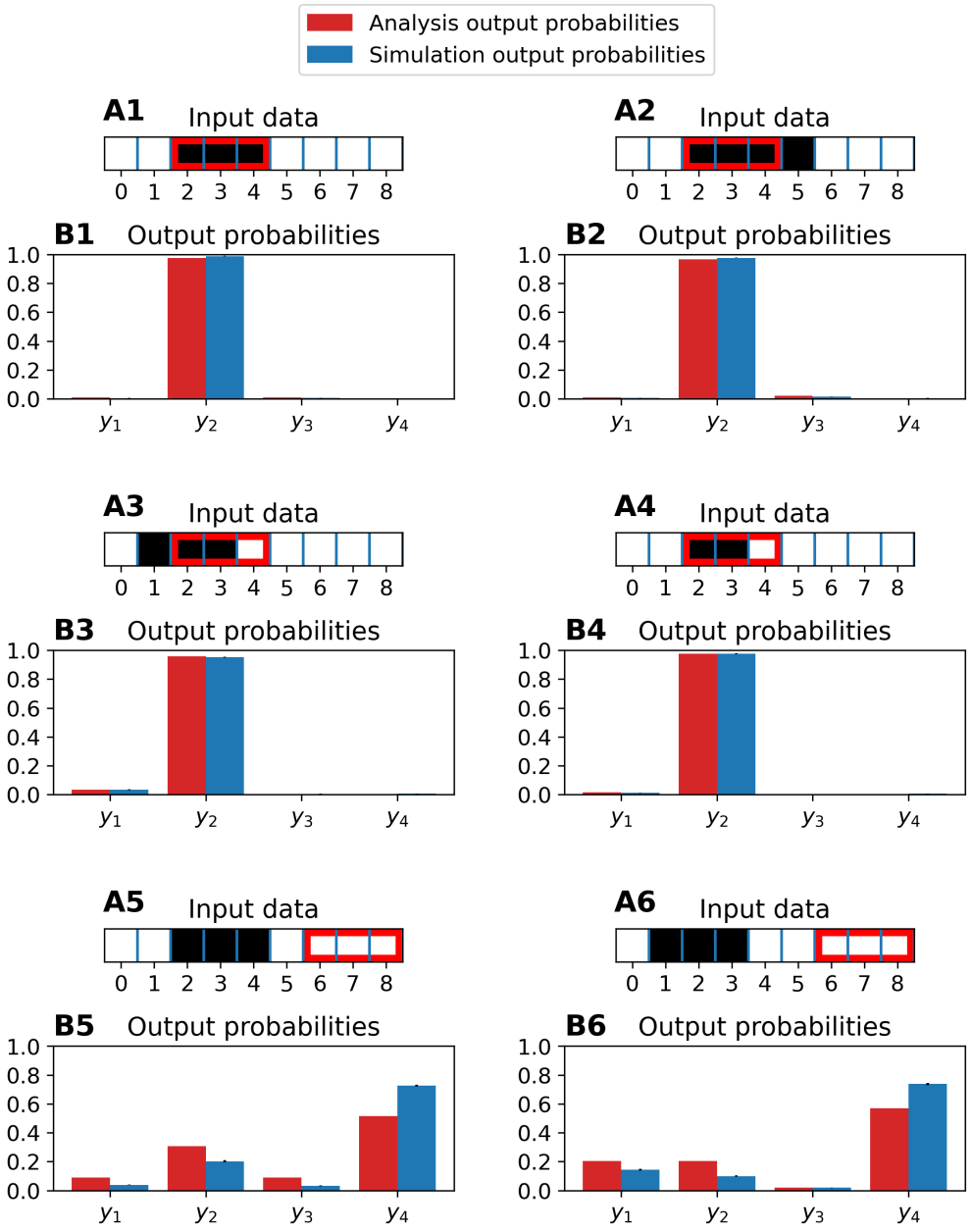

Figure 4.24: **Analysis and simulation result.** Hyperparameters: $f_{input} = 98Hz, f_{prior} = 440Hz, \tau_{decay} = 4ms, c = 3$ **A** Input images with 9 x 1 pixels. Prior given as red border. **B** Analytically calculated posterior probabilities and simulated posterior probabilities. The standard deviations are given by the black bars.

5 Discussion

5.1 Impact of the hyperparameters

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 following paragraph about f_{prior}. Furthermore, by raising f_{input} the membrane potentials of all output neurons are raised equally by the same percentage. However, this causes difficulties when trying to determine the optimal f_{input}. When raising f_{input} it was also observed that the posteriors of output classes adjacent to the active pixels decreased. This can be observed when comparing Tables 4.1 and 4.2 where f_{input} was raised from 42 Hz to 70 Hz. For example when looking at Image 4 in the mentioned tables, it can be seen that for $f_{input} = 42Hz$ the simulation output probability of y_1 is 0.245. This probability is due to the active pixel at position 2, which belongs to class 1 and 2 at the same time. y_2 has a higher simulation output probability of 0.553, as it has two active pixels within its active area. When increasing f_{input} to 70 Hz one might expect the probabilities for y_1 and y_2 to rise. This however does not happen, instead the simulation output probability for y_1 fell to 0.207, while for y_2 it rose to 0.685. It is assumed that this happened because the membrane potentials of the output neurons were never normalized. As the input neurons spike more quickly the membrane potential of y_1 rose by a smaller amount than the membrane potential of y_2, as there was one more active pixel within the active area of y_2. As given by Equation 3.15 q_k depends exponentially on u_k. It is important to notice that all q_k together yield the discrete probability density function of the posterior. When f_{input} is increased all membrane potentials increase by the same percentage. Due to the exponential dependency of the probability density function on the membrane potentials the

different q_k however, do not all change by the same percentage. Rather every q_k increases by a different percentage, the q_k of y_k with the most active pixels by the smallest amount and the q_k of the y_k with the least active pixels by the smallest percentage. However, this behaviour might be desired, as generating more input spikes corresponds stochastically to generating more samples from the likelihood. Taking more samples reduces the standard error of the posterior probability distribution, thus reducing the width of its probability density function around q_k whose y_k has the most active pixels. This means that the more samples of the input the network takes, the more certain it becomes of the most likely option, which is output class 2.

f_{prior} When increasing the prior firing rate the impact on the result of the prior neurons increased, while the impact of the input neurons decreased comparatively. When comparing the Figures 4.17 and 4.18 which show the results for $f_{input} = 88Hz$, $\tau_{decay} = 4ms$ and $f_{prior} = 440$ and $600Hz$ respectively it can be seen under B6 how the simulation output probability for y_4 rose, while all other probabilities fell. The opposite behaviour can be observed when raising f_{input} instead. The same shifting of the probability density function of the posterior as observed for f_{input} was expected, as the prior neurons contribute to u_k in the same way as the input neurons. However, it was impossible to observe it directly as increasing f_{prior} already has the effect of narrowing or widening the probability density function of the posterior around q_k which is supported by the prior, making the two effects indistinguishable from each other.

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 hyperparameters for τ_{decay} of 4 and 15 ms. While for $\tau_{decay} = 15ms$ the best input firing rate was 42 Hz and the prior firing rate was 222 Hz, for $\tau_{decay} = 4ms$ they had to be raised to 98 and 440 Hz to minimize the Kullback-Leibler divergence.

5.2 Basis of the network model

In this thesis a spiking Winner-Take-All neural network model which was taken from Nessler et al. (2013) and expanded by a neuron layer of prior neurons was used. The first hypothesis the network model is based on is that q_k is equal to the posterior probability $P(Y = k|X, Z)$. This relation was explained in Section 3.4 and found conclusive. This model is also based on the hypothesis, that the weights of the network, due to STDP-induced changes, converge stochastically to the log of the conditional probability that a presynaptic neuron has fired shortly before the postsynaptic neuron. This relation, given in Equation 3.18, was proven in Experiment 2. The likelihood $P(X = x|Y = k)$, the prior $P(Y = k|Z = j)$ and the posterior $P(Y = k|X, Z)$ were analytically calculated. Then the input weights were derived from the likelihood and the prior weights were derived from the prior. Then the network was simulated and its output was used to calculate the posterior of the simulation. The Kullback-Leibler divergence between the analytically calculated posterior and the posterior of the simulation was calculated to evaluate how closely the simulation was able to approximate the optimal analytical solution. The results of the best hyperparameter set are given in Figure 4.20 and Table 4.7. The simulation approximated the analytical solution closely and it can be concluded that the hypothesis of Equation 3.18 was correct. However, it was not possible to tune the network to perfectly match the analytical solution. One difficulty when tuning the network was that increasing or decreasing f_{input} had an unexpected effect of changing the probability density function of the posterior. This effect was explained in Section 5.1. However, it is unclear why the analytic posterior could not be approximated closer, as the impacts of f_{input}, as well as f_{prior} seemed correct.

5.3 Reproducing behaviours of the visual cortex

The experiments yielded evidence that a spiking Winner-Take-All neural network can recreate behaviours that were observed in the visual cortex. Neural feedback is most commonly associated with attention. Such attention

behaviour could be shown in Experiment 1, where ambiguous cross images were shown to the network. Without the feedback and with perfectly trained output neurons the network would have not known which part of the cross image to focus on. In reality the output neurons ended up with different sized active areas and unequal relative activities. Due to that, when both parts of the cross were exactly in the middle of the theoretical centers of two output neurons, one output neuron was more active than the other, signifying attention on the corresponding part of the cross. However, this unequal learning of the output neurons could be countered by the activity of the prior neurons that delivered feedback. If the prior activity was set to "horizontal" the corresponding output neuron was always more active than its vertical counterpart. As shown by Lee TS (2003) feedback from the inferior temporal cortex to V1 is able to convince V1 into "seeing" illusory lines. This behaviour was reproduced in Experiment 2. Figures 4.13 and 4.17 show the validation results of the same network with the same hyperparameter set, except once with disabled prior neurons and once with enabled ones. The sixth input image has active pixels at positions 1, 2 and 3. Thus it lies in the middle between classes 1 and 2. When looking at Figure 4.13 A6 and B6 it can be seen that with disabled prior neurons most of the output activity originates from y_1 and y_2, the output of the network matches the visual input. In Figure 4.17 with a prior that signals class 4 in Subfigures A6 and B6 it can be seen that the output probabilities shifted. The most active output neuron now is y_4, while y_1 and y_2 have smaller output probabilities. y_4 being the most active neuron contradicts the visual input, but it is enforced by the prior information it receives. This feedback induced belief matched the experimental evidence of the visual cortex.

5.4 Reusing hyperparameters for networks of different size

The possibility to reuse a hyperparameter set for a network of different size was examined in Experiment 3. In this experiment the number of input neurons, the prior firing frequency and the size of the input images was doubled. Then the weights were re-calculated accordingly to the bigger

size. When comparing Figures 4.20 and 4.21 it can be seen that the analysis output probabilities stayed the same. This was expected, as by doubling the pixels of the input images the information within them did not change. The areas corresponding to an output class doubled in size, as did the active pixels shown in black. The Kullback-Leibler divergence of this experiment was 0.2392 compared to 0.0101 of the original network. This clearly showed that the hyperparameters could not simply be reused for a network of different size. One possible reason for this could be the, discussed in Section 5.1, effect of different f_{input} changing the probability density function of q_k. By doubling the number of input neurons twice as many samples were pulled from the input probability distribution, which made the network prefer output classes with more active pixels within their active areas over output classes with fewer active pixels. Furthermore, the impact of the prior neurons was too high, as seen in Figure 4.21 B6. As discussed in Section 5.1 the exponential dependency of q_k on u_k might have caused this increased impact of the prior neurons. For each validation image only one of the four output neurons received EPSPs from the prior neurons, when noise is disregarded. By raising its u_k to the power of e its proportion to the other membrane potentials gets bigger than before the doubling of f_{prior}. This increased impact of the prior neurons is suspected to be the main reason to why the transfer of the hyperparameters performed poorly.

5.5 Training weights with predetermined hyperparameters

In Experiment 4 the best hyperparameter set of Experiment 2 was used to train the input and prior weights of the network. The goal of this was to determine how well the network could approximate the optimal weights, as well as the posterior. Only the weight shifting hyperparameter c was varied to minimize the Kullback-Leibler divergence. With best value of $c = 3$ the Kullback-Leibler divergence was 0.0342 compared to 0.0101 of the network with the calculated weights from Experiment 2. When comparing A with B and C with D in Figure 4.23 it can be seen that the network learned the correct discrimination function. However, the input weights were too small,

while the prior weights were too big. This observation explains why further raising the value of c did not improve the Kullback-Leibler divergence, as it increases the input and prior weights at the same time. The network was not able to further approximate the optimal weights further, primarily because only one weight shifting hyperparameter existed for both the input and prior weights. By adding a second separate weight shifting hyperparameter it would be possible to increase the values of the input weights and to decrease the values of the prior weights at the same time.

5.6 Future work

The narrowing and widening of the probability density function of the posterior when changing f_{input} caused problems in this thesis and the model could potentially be improved by analysing it further. It was caused by the exponential dependency of q_k on u_k, which seems intended and indicates that this behaviour was not caused by an implementation error of the model. It would be interesting to remove the exponential dependency to determine if the changing of the probability density function of the posterior disappears. Furthermore, it could be tested if the hyperparameters of the network would be reusable for networks of different sizes when the exponential dependency is removed.

The network architecture could be expanded by additional layers or other networks to increase the complexity of the experiment paradigms. For now the prior neurons were told what feedback to give, but they could also receive input from another network that looks at different characteristics of the input data. This way the network architecture would be more realistic. The prior neurons could also receive EPSPs from the output neurons, this way the prior neurons would receive feedback of how their EPSPs impacted the output neurons. This is a vital part that happens in the brain that was neglected in this network architecture.

Experiment 4 showed problems that were caused by having only a single weight shifting hyperparameter for both input and prior weights. A second separate weight shifting parameter could be implemented to quickly resolve this problem. A more sophisticated solution could be to design a variable

weight shifting parameter that depends on characteristics of the neuron. The experiment suggested that neurons with higher firing frequencies need lower values of c, so a variable c could depend inversely on the firing frequency.

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. 5).

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. 5).

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 (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. 8).

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/nn.4390 (cit. on pp. 1, 8).

Gerstner, Wulfram and Werner M. Kistler (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press (cit. on pp. 2, 8, 10, 11).

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 (cit. on p. 2).

Huff T Mahabadi N, Tadi P. (2024). *StatPearls [Internet]*. StatPearls Publishing. Chap. Neuroanatomy, Visual Cortex. (Cit. on p. 7).

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. URL: https://doi.org/10.1007/s10827-006-7074-5 (cit. on p. 17).

Jones EG; Hendry, SHC, ed. (1984). *Cerebral cortex: cellular components of the cerebral cortex.* ew York: Plenum Press. Chap. Basket cells, pp. 309–334 (cit. on p. 5).

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. 13).

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 (cit. on pp. vii, 1, 3, 8, 9, 66).

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. 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, 5).

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. URL: https://www.frontiersin.org/articles/10.3389/neuro.11.037.2009 (cit. on pp. 1, 6).

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. URL: https://doi.org/10.1371/journal.pcbi.1003037 (cit. on pp. 1, 2, 14–17, 19, 25, 35, 65).

Palmer, Stephen (Jan. 1999). *Vision Science: From Photons to Phenomenology*. Vol. 1, p. 153 (cit. on p. 7).

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. URL: https://www.frontiersin.org/articles/10.3389/fncom.2018.00090 (cit. on pp. 1, 8).

Pouget, Alexandre et al. (Sept. 2013). "Probabilistic brains: knowns and unknowns." In: *Nature Neuroscience* 16.9, pp. 1170–1178. ISSN: 1546-1726.

DOI: 10.1038/nn.3495. URL: https://doi.org/10.1038/nn.3495 (cit. on p. 8).

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. URL: https://www.sciencedirect.com/science/article/pii/S0959438802003045 (cit. on p. 7).

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 (cit. on pp. 2, 5).

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. 11).