

## 3 Theoretical background

### 3.1 Bayes' theorem

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

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

where

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

### 3.2 Bayesian inference

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

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

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

### 3 Theoretical background

after the evidence was observed.  $P(E|H)$  is called likelihood and it is the probability of observing the evidence, given the hypothesis. It is a measure of the compatibility of the observed evidence with the given hypothesis.  $P(H)$  is the prior probability, which is the probability of the hypothesis before any evidence is considered or obtained.  $P(E)$  is called marginal likelihood that represents the probability of the evidence being observed. However, it is 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  $V_1$  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  $V_1$ . This influences the prior probabilities and changes the posterior probability. Through that, the  $V_1$  neuron starts to see the hypothesis "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  $V_1$  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

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

### 3.3 Network model

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

14

Dafür muss man annehmen dass  $X$   
unabhängig von  $Z$  ist bei geg.  $Y$ .  
Das sollte gezeigt werden und das  
graphische Modell sollte gezeigt werden.



### 3.3 Network model

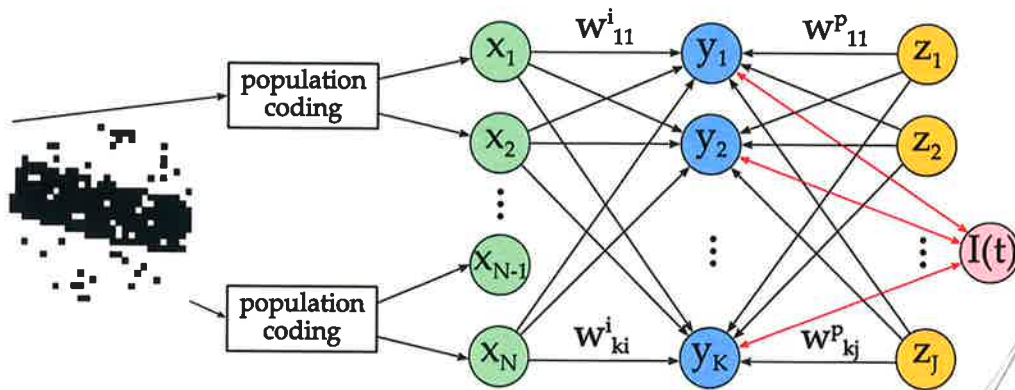


Figure 3.1: Architecture of the network.

**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  $x_1, \dots, x_N$ . These input neurons are fully connected to the excitatory output neurons  $y_1, \dots, y_K$ . This means that every input neuron  $x_i$  is connected to each output neuron  $y_k$ . To simulate the feedback from the inferior temporal cortex prior neurons,  $z_1, \dots, z_J$  were added to the network. 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  $x_1, \dots, x_N$  are firing according to a poisson process with an average firing rate  $f_{input}$  when active and with 0 Hz when in an inactive state. The input neurons receive binary input, which is either a black, or a 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  $\tau_{rise}$  and a time constant for the decay of the signal  $\tau_{decay}$ . The addition of the time step size  $\delta t$  was necessary to get the time  $t$  at the end of the current simulation step.  $t_f$  is the time at which the spike of  $x_i$  occurred

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

this is not correct.  
what if there are more input spikes?  
You define here a kernel for the EPSP  
(#1)

### 3 Theoretical background

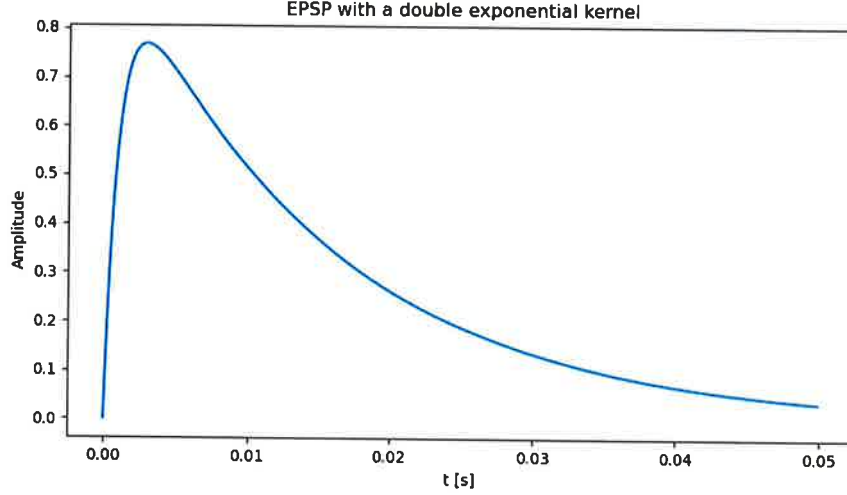


Figure 3.2: Form of an excitatory post synaptic potential generated by an input neuron over time. A double exponential kernel was used to generate this signal. Its value is passed on to the next layer of the network.

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

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

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

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

In Nessler et al. (2013) each output neuron  $y_k$  also had an intrinsic excitability  $w_{k0}$ , which was learned for each neuron. For the experiments of this thesis 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.

Es sollte vorher beschrieben werden dass die Aktivierung von output Neuron  $K$  ist.

### 3.3 Network model

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

The total firing rate of the output neurons is obtained <sup>by</sup> when summing up the firing rates of all output neurons, yielding

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

The inhibition signal  $I(t)$  was chosen to depend on the current membrane potential of the output neurons. Solving Equation 3.7 for  $I(t)$  yields

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

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

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

$$I(t) = -\ln R(t) + \ln \sum_{k=1}^K e^{u_k(t)}. \quad (3.11)$$

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

$$p(y_k \text{ fires at time } t) \propto e^{u_k(t) - I(t)}. \quad (3.12)$$

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

$$r_k(t) = e^{u_k(t) - I(t)}. \quad (3.13)$$

### 3 Theoretical background

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 probability of an individual output neuron to fire within a time step  $\delta t$  is given by

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

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

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

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

→ *Anmerken dass das eine Software Fkt ist!*

**Spike timing dependent plasticity** The input weights  $w_{ki}^I$  between neurons  $x_i$  and  $y_k$  are updated whenever an output neuron fires.  $\sigma$  determines 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

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


where  $\lambda$  is the learning rate, the hyperparameter  $c$  shifts the weight values,  $t^f$  is the time when  $y_k$  spiked and  $\sigma$  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  $\lambda$  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

*↓  
also ein soft-MTA*

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

---

might not respond at all. On the other hand if  $\lambda$  is too small the network will learn very slowly and may never converge. The prior weights  $w_{kj}^P$  are also updated whenever an output neuron fires, in the same way as  $w_{ki}^I$

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


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

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

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

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 explain the connection between the spiking Winner-Take-All network model and Bayesian inference it will be demonstrated, 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 3.12, the firing probability of an output neuron is proportional to the exponent of its



### 3 Theoretical background

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, which signifies the contribution of the input neurons, is given by

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

The second part gives the contribution of the prior neurons as

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

By inserting each of them into Equation 3.12 we get the likelihood

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

and the prior

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

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

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

*Ich glaube das kann man nicht beschreiben ohne vorher das graphische Modell erklärt zu haben.*



## 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. The network will train its weights through unsupervised learning. Through that, it will learn to cluster the images together, depending on the orientation and position of the bars. The learning of the network will be visualized and analysed. Further, the effect of the prior neurons will be demonstrated. Ambiguous images, which show a horizontal and a vertical bar at the same time, will be generated and given to the network. Through that, the network's capability to shift its attention to specific parts of the image, depending on the prior will be demonstrated.

#### 4.1.2 Methods

**Input data** Black bars, with a width of seven pixels, were drawn onto a white background, with a size of  $35 \times 35$  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 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

use post tense

were

↓ tense

and shifted in position

← was ~~there~~   
 needed   
 obs?

### Examples of training images

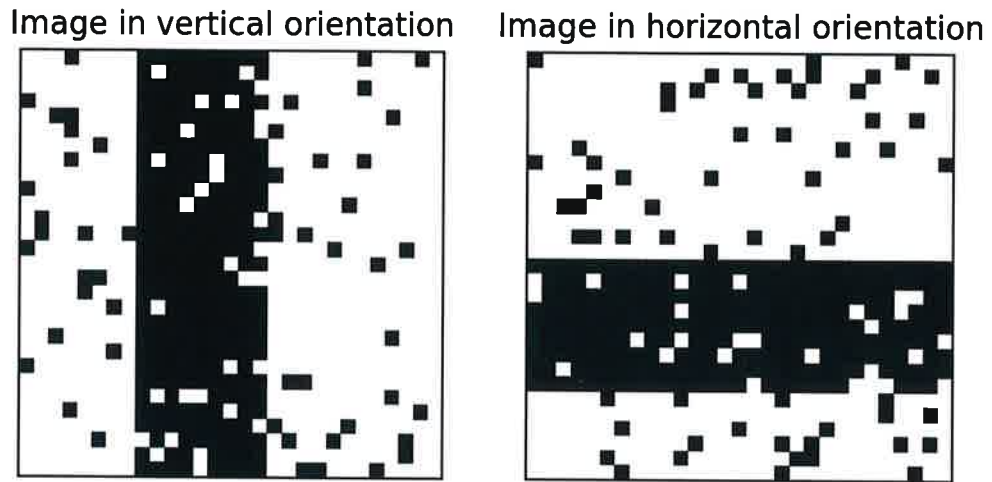


Figure 4.1: Generated training images. One image of each possible orientation at a random position.

distributed randomly, via a uniform distribution, along the 35 pixels 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, which were 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** The network had 2450 input neurons, as every pixel of an image was connected to two input neurons. Each of these neurons had a firing frequency  $f_{input}$  of 20 Hz, when in the active state and 0 Hz otherwise. Then there were ten output neurons, one for each possible

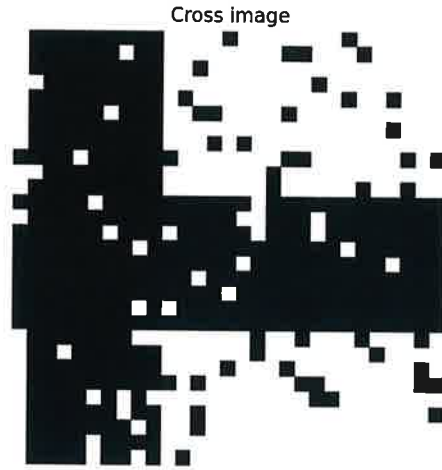


Figure 4.2: Generated cross image, which can represent either horizontal or vertical orientation.

group. Lastly, there were two equal sized groups of prior neurons with firing frequencies  $f_{prior}$  of 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. Just like the input images' pixels, the prior had a chance of 10 percent to flip. 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  $\sigma$  was 10 ms, the time constant for the rise of the EPSPs  $\tau_{rise}$  was 1 ms and the time constant for the decay of the EPSPs  $\tau_{decay}$  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.

## 4 Experiments

---

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

no space →  
4.3A

(\*)

(3.4) should be

$$E(s) = e^{-\frac{(s + \delta t_{\text{decay}})}{T_{\text{decay}}}} - e^{-\frac{(s + \delta t)}{T_{\text{rise}}}}$$

$$\text{then } x_i(t) = \sum_{t_i^{(f)}} E(t - t_i^{(f)})$$

see also Gerstner & Kistler.

$x_i(t)$  is actually the unweighted membrane potential response ~~at~~ due to input neuron  $i$  at the postsynaptic neurons.

Analogy für  $z_j(t)$

Um präzise zu sein sollte  $E$  auch eine Heaviside Stepfunktion beinhalten  
denn  $E(s) = 0$  für  $s < 0$

$$\text{Also } E(s) = \Theta(s) \cdot (e^{-\dots} - e^{-\dots})$$