Dissertation

submitted to the
Combined Faculty of Natural Sciences and Mathematics
of Heidelberg University, Germany
for the degree of

Doctor of Natural Sciences

Put forward by

Dominik Dold

born in: Titisee-Neustadt, Germany

Oral examination: 11.05.2020

Harnessing function from form: towards bio-inspired artificial intelligence in neuronal substrates

Referees: Prof. Dr. Andreas Mielke
Prof. Dr. Walter Senn

Abstract

Despite the recent success of deep learning, the mammalian brain is still unrivaled when it comes to interpreting complex, high-dimensional data streams like visual, auditory and somatosensory stimuli. However, the underlying computational principles allowing the brain to deal with unreliable, high-dimensional and often incomplete data while having a power consumption on the order of a few watt are still mostly unknown.

In this work, we investigate how specific functionalities emerge from simple structures observed in the mammalian cortex, and how these might be utilized in non-von Neumann devices like "neuromorphic hardware". Firstly, we show that an ensemble of deterministic, spiking neural networks can be shaped by a simple, local learning rule to perform sampling-based Bayesian inference. This suggests a coding scheme where spikes (or "action potentials") represent samples of a posterior distribution, constrained by sensory input, without the need for any source of stochasticity. Secondly, we introduce a top-down framework where neuronal and synaptic dynamics are derived using a least action principle and gradient-based minimization. Combined, neurosynaptic dynamics approximate real-time error backpropagation, mappable to mechanistic components of cortical networks, whose dynamics can again be described within the proposed framework.

The presented models narrow the gap between well-defined, functional algorithms and their biophysical implementation, improving our understanding of the computational principles the brain might employ. Furthermore, such models are naturally translated to hardware mimicking the vastly parallel neural structure of the brain, promising a strongly accelerated and energy-efficient implementation of powerful learning and inference algorithms, which we demonstrate for the physical model system "BrainScaleS–1".

Zusammenfassung

Trotz der jüngsten Erfolge im Bereich Deep Learning ist das Gehirn der Säugetiere immer noch ungeschlagen im Verarbeiten komplexer, hochdimensionaler Datenströme wie visuelle, auditive und somatosensorische Stimuli. Die zugrundeliegenden Rechenprinzipien, die es dem Gehirn ermöglichen mit unzuverlässigen, hochdimensionalen und oft unvollständigen Daten umzugehen, während eine Leistung in der Größenordnung weniger Watt verbraucht wird, sind jedoch noch größtenteils unbekannt.

In dieser Arbeit untersuchen wir, wie aus im Kortex von Säugetieren vorkommenden Strukturen Funktionalität hervorgeht, und wie diese in Nicht-von-Neumann-Geräten wie "neuromorpher Hardware" Anwendung finden könnte. Zuerst zeigen wir, dass Ensembles aus deterministischen, spikenden neuronalen Netzen, durch eine lokale Lernregel trainiert, Bayessche Inferenz implementieren. Dies deutet auf ein Codierungsschema hin, bei dem Spikes (oder "Aktionspotentiale") Stichproben einer A-posteriori-Verteilung darstellen – ohne dass eine Quelle für Stochastizität erforderlich ist. Darüberhinaus führen wir ein theoretisches Konzept ein welches die Herleitung neuronaler und synaptischer Dynamik durch ein Wirkungsfunktional sowie gradientenbasierter Minimierung ermöglicht. Vereint approximiert die neurosynaptische Dynamik Fehlerrückpropagation in Echtzeit und lässt sich auf mechanistische Komponenten des Kortex abbilden, deren Dynamik wieder durch dieselben Konzepte beschrieben werden kann.

Die vorgestellten Modelle reduzieren die Diskrepanz zwischen wohldefinierten, funktionalen Algorithmen und deren biophysikalischer Implementierung, was zu einem verbesserten Verständnis der Rechenprinzipien, die das Gehirn anwenden könnte, führt. Darüber hinaus sind diese Modelle ideale Kandidaten für Hardware Systeme, welche die massiv parallele neuronale Struktur des Gehirns nachahmen und eine stark beschleunigte und energieeffiziente Realisierung leistungsfähiger Lern- und Inferenzalgorithmen versprechen – was wir für das physikalische Modellsystem "BrainScaleS–1" demonstrieren.

Dedicated to my parents,

Bernhard Dold & Helga Dold (born Fischer)

and in loving memory of my grandfather

Josef Dold

who passed away during writing of this thesis.

This is only a foretaste of what is to come, and only the shadow of what is going to be. We have to have some experience with the machine before we really know its capabilities. It may take years before we settle down to the new possibilities, but I do not see why it should not enter any one of the fields normally covered by the human intellect, and eventually compete on equal terms.

Alan Turing, The Times, June 1949
"The Mechanical Brain"

Contents

1. Prolog **1**

2. Introduction: neural networks - the substrate of intelligence **5**
2.1. Computational models of neurons and synapses 5
2.1.1. The deep learning revolution: abstract neurons 5
2.1.2. To spike or not to spike: biological neurons 7
2.1.3. The neocortex: spatial structures and circuitry 10
2.2. Towards silicon brains: neuromorphic hardware 13
2.3. Outline: harnessing function from form 17

3. Spike-based coding as deterministic Bayesian inference **19**
3.1. The Bayesian brain hypothesis 19
3.1.1. Neuronal noise 19
3.1.2. Noise as the hallmark of probabilistic reasoning 21
3.2. Stochastic inference in spiking neural networks 25
3.2.1. The high-conductance state 25
3.2.2. Spikes as samples from a probability distribution 26
3.2.3. Training spike-based sampling networks 29
3.3. Spiking networks for spatio-temporal predictions 31
3.3.1. From static patterns to time-dependent input 31
3.3.2. Pattern completion of line segments 32
3.3.3. Short-term plasticity improves exploration 36
3.4. Sampling in deterministic ensembles 40
3.4.1. The self-noising brain 40
3.4.2. Characterizing ensemble-generated noise 41
3.4.3. Deterministic probabilistic inference with spikes 47
3.4.4. Deterministic inference in hierarchical networks 49
3.4.5. From many to one: ensembles with only one network 53
3.4.6. Functional ensembles on a physical neuronal substrate 55
3.5. Discussion 57
3.6. Contributions 58

4. Intermezzo **59**

5. Deep learning in mechanistic models of cortical networks **61**
5.1. Deep learning in the brain? . . . 61
5.1.1. The error backpropagation algorithm . . . 63
5.1.2. The challenges of backprop in the brain . . . 65
5.2. Real-time error backpropagation in cortical circuits . . . 67
5.2.1. A principled approach to learning in dynamical systems . . . 67
5.2.2. Gradient-based neuron dynamics . . . 68
5.2.3. From gradients to lookahead dynamics . . . 70
5.2.4. Deriving lookaheads via prospective least action . . . 72
5.2.5. Dendritic microcircuits for local error learning . . . 76
5.2.6. Summary: neuronal least action principle . . . 81
5.3. Folded autoencoders for unsupervised learning in cortical hierarchies . . . 83
5.3.1. Energy function for cortical autoencoders . . . 84
5.3.2. One learning rule, two optimizations . . . 86
5.3.3. Towards probabilistic generative models . . . 88
5.4. Discussion . . . 90
5.5. Contributions . . . 92

6. Epilog **93**

Acknowledgments **101**

Funding **103**

A. Appendix of chapter 3 **105**
A.1. Calculations . . . 105
A.1.1. Free membrane potential distribution with colored noise . . . 105
A.1.2. Width of free membrane potential distribution . . . 106
A.1.3. Approximate inverse slope of LIF activation function . . . 109
A.1.4. Origin of side-peaks in the noise autocorrelation function . . . 109
A.1.5. Cross-correlation of free membrane potentials receiving correlated input 112
A.1.6. State space switch from {0,1} to {-1,1} . . . 114
A.1.7. Translation from Boltzmann to neurosynaptic parameters . . . 115
A.2. Additional results . . . 116
A.2.1. Video S1: ensemble sampling on BrainScaleS-1 . . . 116
A.2.2. Video S2: ensemble of networks dreaming of MNIST . . . 116
A.2.3. Video S3: single-network ensemble dreaming of MNIST . . . 117
A.3. Simulation details . . . 117
A.3.1. Figs. 3.10 to 3.12: spatio-temporal predictions . . . 117
A.3.2. Figs. 3.15, 3.16 and 3.18 to 3.23: general details . . . 118
A.3.3. Fig. 3.15: autocorrelations of ensemble background . . . 119
A.3.4. Fig. 3.16: cross-correlations of ensemble background . . . 119
A.3.5. Figs. 3.18 and 3.19: calibration and plasticity scheme . . . 120
A.3.6. Figs. 3.20 to 3.22: ensemble-based inference on (E)MNIST . . . 120

A.3.7. Fig. 3.23: single-network ensemble dreaming of MNIST 121
A.3.8. Fig. 3.24: neuromorphic deterministic sampling 121
A.4. Implementation details 122
A.4.1. Simulation software 122
A.4.2. Emulation software 123

B. Appendix of chapter 5 **125**
B.1. Calculations 125
B.1.1. Lookaheads arising from spike-generating mechanisms 125
B.1.2. Variation of weights leading to variation of potentials 126
B.1.3. Formulation as 5-compartment neuron model 128
B.2. Additional results 129
B.2.1. Learning MNIST with lookahead dynamics 129
B.2.2. Truncated backprop for recurrent networks 130
B.3. Simulation details 131
B.3.1. Fig. 5.5: supervised learning of iEEG traces 131
B.3.2. Figs. 5.6 and B.2: real-time learning of MNIST 131
B.3.3. Fig. 5.8: error learning with cortical microcircuits 132
B.3.4. Fig. 5.9: microcircuit learning without top-down nudging 132
B.3.5. Fig. 5.12: folded autoencoder learning CIFAR10 132
B.3.6. Fig. 5.13: dreaming of MNIST using frozen noise 133
B.3.7. Fig. B.3: truncated backprop for recurrent networks 133
B.4. Implementation details 134
B.4.1. Numerical methods 134
B.4.2. Simulation software 137

C. List of published articles **139**

Bibliography **141**

Notation **163**

Acronyms **165**

Index **167**

1 | Prolog

> *My feeling is, if you wanna understand a really complicated device like a brain, you should build one. I mean you could look at cars and you can think you can understand cars. When you try and build a car, you suddenly discover there's this stuff that has to go under the hood, otherwise it doesn't work.*
>
> **Geoffrey Hinton**, Bloomberg documentary *"This Canadian Genius Created Modern AI"*

The human brain is a complex network consisting of nodes (various neuron types and dendritic branches) and vertices (synaptic connections and axions), structured into a vast network of distinct brain regions that take on different tasks, like relaying information between brain regions ("thalamus", *Moustafa, McMullan, Rostron, Hewedi, and Haladjian* 2017) or higher-level cognitive functions ("cerebral cortex", *Frith and Dolan* 1996). What we consider to be intelligence is nowadays believed to be an emergent phenomenon of the complex interactions and interplay between these components – neurons, dendritic branches, synapses and biochemical processes – which together orchestrate a symphony of network activity that far surpasses the functionality of its individual components. Similar to elementary particles in physics, neurons have been identified as the protagonists of this spectacle, although different from elementary particles, neurons come with a complex internal structure, are not identical and do not necessarily interact symmetrically ("actio et reactio") with each other – in fact, what we call "memory" or "learning" is believed to be mostly consolidated in the strength of these interactions. Therefore, interaction strengths between neurons change over time, a process known as "synaptic plasticity", and are molded by the complex interaction between environment and nervous system, coupled together by our window into reality – our senses of vision, hearing, smell, taste and touch.

Still, how the brain almost effortlessly achieves a high performance in the myriad of complex tasks that we face every day while only consuming around 20W of power (*Mink, Blumenschine, and Adams*, 1981) is still one of many open questions. For instance, how does the brain encode information? To transmit information between neurons, the brain uses all-or-nothing events, so-called "action potentials" or "spikes" that contain precise temporal information – the time of spike initiation. The possibly simplest method to encode information with spikes is to count them, i.e., to average over a certain time window or neuron population to arrive at an analog quantity that carries information: spike "rates". However, by averaging, the precise temporal information of spikes is lost. Possible coding schemes

that utilize the temporal aspect of spike times are, to name a few, (i) latency coding, where information is encoded in the delay between input arrival and output spike time of a neuron, (ii) phase coding, where information is encoded in the phase relation between spike time and an oscillatory signal, or (iii) rank order coding, where the order of spike times in a population of neurons encodes information (*Thorpe, Delorme, and Van Rullen*, 2001; *Van Rullen, Guyonneau, and Thorpe*, 2005). Although many candidates exist, which coding scheme, or rather what combination of schemes the brain uses is still an open question. Another example of open questions concerns learning in the brain, i.e., how does synaptic plasticity, supported by additional neuronal structures, shape the connectivity in the brain? The basic mechanism of synaptic plasticity between neurons is nowadays quite well understood and believed to be largely "Hebbian" in nature (*Hebb*, 1949), i.e., *what fires together wires together*, or in simpler terms: if one neuron is involved in the spiking of another, their synaptic connection is strengthened and weakened otherwise. A famous spike-based realization of Hebbian plasticity, based on phenomenological observations (*Bi and Poo*, 2001), is spike time dependent plasticity (STDP). However, how local synaptic changes have beneficial effects on a global network level, e.g., by improving the behavioral response of an animal to a visual stimulus, is still shrouded in mystery. Considering the recent success in training abstract hierarchical neural networks to reach or even exceed human performance in perception tasks like image recognition ("deep learning", *Ivakhnenko* 1971; *LeCun, Bengio, and Hinton* 2015; *Schmidhuber* 2015), the quest of unlocking the algorithm guiding learning in the brain has obtained particularly much traction lately (*Kuśmierz, Isomura, and Toyoizumi*, 2017; *Hassabis, Kumaran, Summerfield, and Botvinick*, 2017; *Pfeiffer and Pfeil*, 2018; *Sacramento, Costa, Bengio, and Senn*, 2018; *Lillicrap and Kording*, 2019; *Richards et al.*, 2019; *Krotov and Hopfield*, 2019; *Illing, Gerstner, and Brea*, 2019; *Whittington and Bogacz*, 2019).

In this work, we try to shed some light on these questions by investigating the possible computational, and especially functional, purpose of features observed in biology – to see how function emerges from form. Such an approach is destined to disregard many aspects of the biological brain, in an attempt to uncover the minimally required structures and mechanisms to perform computations in biologically plausible neural networks. Apart from improving our knowledge of the brain, this duality of form and function – or mechanistic realization (*Dayan and Abbott*, 2001) and algorithmic purpose – also enables us to research novel computing paradigms and architectures that are much closer to the brain than computers nowadays, called neuromorphic hardware (*Schuman, Potok, Patton, Birdwell, Dean, Rose, and Plank*, 2017). These devices promise a new generation of hardware for artificial intelligence (AI) systems, mimicking aspects of the brain to increase energy and computational efficiency. Moreover, a subset of these systems raise the additional question of how we can perform powerful computations using the laws of physics aside from digital logic circuits and Boolean algebra – and a paragon for such a type of device is the human brain. This search for alternative computing methods becomes more and more relevant the closer we get to an era beyond Moore's law (*Waldrop*, 2016).

Throughout this work, we are mostly interested in two of the probably most discussed topics in computational neuroscience at the time of writing this thesis: spike-based coding and the mechanisms behind learning in the brain. We will show that spikes can act both as enablers and executors of stochastic sampling, introducing an elegant realization of

probabilistic computing in biological neural systems. Furthermore, we will demonstrate how biologically plausible concepts and components like advanced responses, local microcircuits and extended neuron models can (i) be unified to realize real-time gradient-based learning for cortical networks and (ii) be tied to top-down concepts like the principle of least action. While investigating these topics, we stay close to our initial philosophy of coupling computation and its mechanistic implementation. By doing so, the proposed models can be neatly summarized in terms of Marr's three levels of analysis (*Marr*, 1982), offering a structured understanding of information processing in neural systems via decreasingly abstract descriptions: the computational level (*what is being done?*), the algorithmic level (*how is it being done?*) and the implementation level (*how is it physically realized?*). Further, both models can be unified under the concept of "predictive coding" (*Rao and Ballard*, 1999): in the spike-based sampling model, networks learn an internal, predictive model of their environment, enabling them to adjust their own activity ("top-down modulation of perception") even when exposed to sensory stimuli ("bottom-up input"). In the presented learning framework, both network-internal as well as temporal predictions are essential to calculate error signals that flow via feedback connections through the network ("top-down modulation of activity") and shape cortical plasticity. Ultimately, the presented results further our understanding of bio-inspired AI and pave the way to efficient neuromorphic algorithms.

In the next chapter, we first give an introduction to the basic principles required to follow this work, covering neuron models, the structural organization of the mammalian neocortex as well as neuromorphic hardware. Afterwards, the main results of this thesis are presented, separated into two larger chapters, one covering spike-based coding and the other learning in the brain. Both result chapters further include introductory sections that explore the specific background of each chapter's topic in more detail. The thesis ends with a brief epilog where all major results are summarized and discussed in a broader context.

2 | Introduction: neural networks - the substrate of intelligence

2.1 Computational models of neurons and synapses

2.1.1 The deep learning revolution: abstract neurons

Recently, deep learning models have shown astonishing results in various tasks like image classification (*Ciregan, Meier, and Schmidhuber*, 2012; *Krizhevsky, Sutskever, and Hinton*, 2012; *He, Zhang, Ren, and Sun*, 2016), playing games on (super)human niveau (*Mnih, Kavukcuoglu, Silver, Graves, Antonoglou, Wierstra, and Riedmiller*, 2013; *Silver et al.*, 2016, 2017, 2018; *Vinyals et al.*, 2019), language translation (*Sutskever, Vinyals, and Le*, 2014), content creation (*Zhu, Krähenbühl, Shechtman, and Efros*, 2016; *Liu, Allamanis, Brockschmidt, and Gaunt*, 2018), physics engines (*Holden, Duong, Datta, and Nowrouzezahrai*, 2019), speech synthesis (*Jia et al.*, 2018) and many more. At the heart of all these models are small elements called neurons, which can be seen as a rather high-level abstraction of biological neurons: they perform a weighted sum over their inputs, apply a non-linear transformation and return the resulting scalar value as an output to be processed by other neurons (Fig. 2.2A), i.e.,

$$u_k = \sum_{\text{syn. } j} w_j \bar{r}_j \,, \qquad \bar{r}_j = \varphi(u_j) \,, \tag{2.1}$$

where w_j are real, scalar weights for the ("synaptic") inputs $\bar{r}_j$, and φ is a monotonic, differentiable non-linear function. The non-linearity is inspired by biology, where neurons transform analog inputs into discrete outputs (see Section 2.1.2) and is required for neural networks to model non-linear relationships. Although single neurons are not very powerful, the insight that lead to the deep learning revolution lies in stacking areas[1] of such neurons together[2] (*Ivakhnenko*, 1971), a so-called multilayer perceptron (MLP) (*Rosenblatt*, 1958) which can then be trained end-to-end (i.e., by only using the input and target output) using the error backpropagation algorithm (*Linnainmaa*, 1970; *Werbos*, 1982; *Rumelhart, Hinton, and Williams*, 1986) – a gradient-based optimization method with the goal of minimizing a

[1] Also called "layers" in the deep learning literature.

[2] If φ was linear, stacking areas of neurons would be like concatenating linear functions, which is again a linear function. Thus, to model non-linear relationships in data, even for hierarchical networks, we require non-linear activation functions.

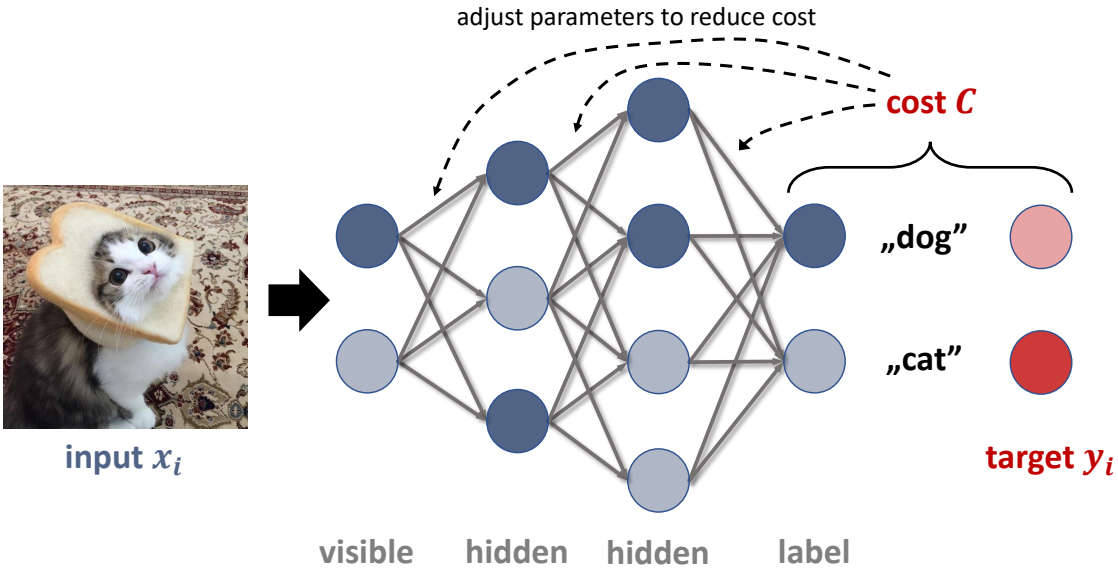

Figure 2.1.: Schematic illustration of an abstract neural network. An input is processed through areas of hidden neurons, i.e., neurons that are not constrained by data. The output is generated by the last area, also known as label area. A cost function evaluates how similar network output and target label are. From this, parameter updates are derived to improve the network's prediction. Color intensity corresponds to neuronal activity. Photograph taken from favim.com, 18.02.2020.

cost function measuring the task-specific performance of the trained network. For instance, lets assume we want to classify images of animals with a network of four areas: a "visible" area receiving the input, a "hidden" area processing the output of the visible area, a second hidden area processing the output of the first hidden area, and a "label" or output area where each neuron represents a possible class, i.e., in this case animals (Fig. 2.1). Given a training image with class label, the cost function measures how well network output and label match. For classification and regression, prominent choices for the cost function are cross-entropy and Euclidean norm, respectively. The goal of training now consists in adjusting the weights between areas – visible to first hidden, first hidden to second hidden and second hidden to label – such that the average cost over the training data set is minimized. Such a set of weights can be found by incrementally descending along the gradient of the cost function, which is achieved in an especially elegant way by the error backpropagation algorithm (see Section 5.1.1 for details). After training, each area of such a network learned useful features that can be used for classification, e.g., the first area learned edge detectors, while the second learned more complex features like corners – forming a hierarchy of increasing complexity, without any need for human feature engineering. In the same spirit, abstract neurons can be connected recurrently and trained on temporal sequences like speech using the error backpropagation through time (BPTT) algorithm (*Rumelhart, Hinton, and Williams*, 1986; *Werbos et al.*, 1990), which interprets the temporal dimension as hierarchical stacking, making it possible to apply error backpropagation again.

Although very successful, deep learning algorithms currently only reach (super)human performance in a narrow set of applications and are still far away from an artificial general intelligence (AGI), i.e., an AI that, like humans, performs well over a large set of tasks.

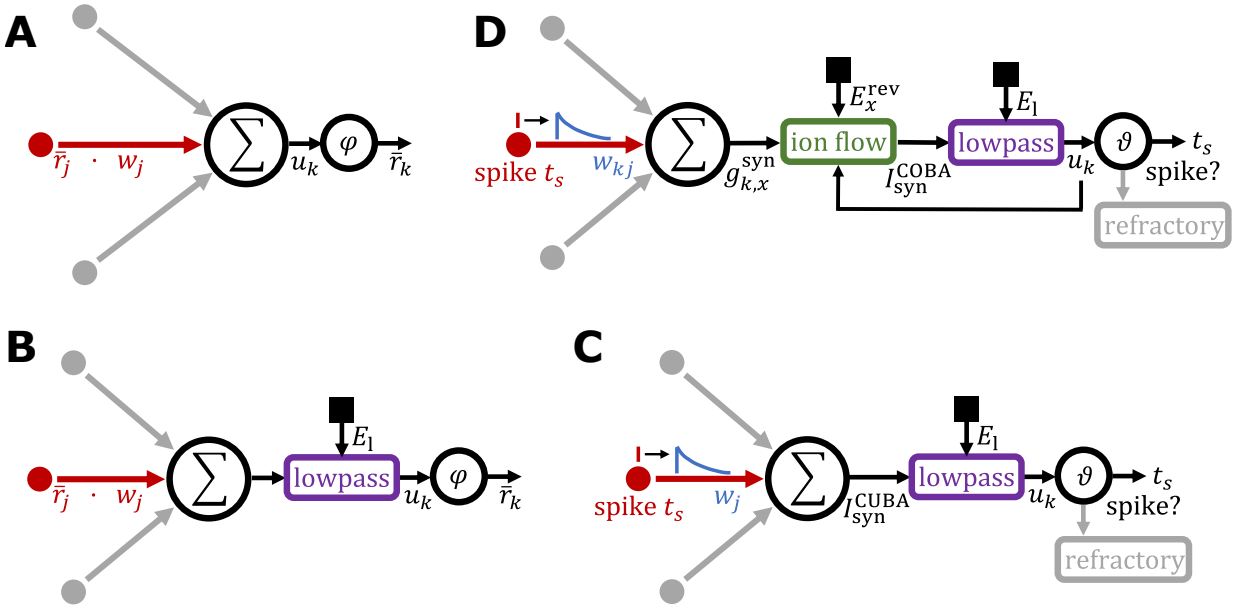

Figure 2.2.: Illustration of different neuron models. **(A)** Simple neuron model mostly used in deep learning that sums over inputs and applies a non-linear transformation. **(B)** Same as (A), but inputs are low-pass filtered, similar to how the membrane of biological neurons integrates inputs. **(C)** Current-based leaky integrate-and-fire model. **(D)** Conductance-based leaky integrate-and-fire model. For convenience, excitatory and inhibitory conductances are not shown separately.

Moreover, even state-of-the-art classification models suffer from defects like adversarial attacks, where small changes in the input can lead to drastic changes in the output and thus wrong classification[3] (*Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, and Fergus*, 2013). To further our understanding of AGI, brain-like or not, it is therefore inevitable to look in both directions, abstract neural networks in deep learning as well as their biological counterparts studied in neuroscience (*Hassabis, Kumaran, Summerfield, and Botvinick*, 2017; *Richards et al.*, 2019; *Sejnowski*, 2020).

2.1.2 To spike or not to spike: biological neurons

Biological neurons are dynamic systems made up of many components like a cell membrane made of a phospholipid bilayer, proteins forming ion channels, ion pumps, etc. Thus, they are much more complicated than simply integrating inputs and returning a non-linear version thereof. An illustration of the general structure of biological neurons is shown in Fig. 2.3. As the crudest model of a biological neuron, we can modify the abstract neuron model (Eq. 2.1)

[3]It is hard to quantify whether humans show a similar vulnerability to adversarial attacks, since white box attacks (finding weaknesses of the network by having complete access to its inner workings) through the cortical visual system are, for now, impossible. However, it has been found that adversarial examples for abstract networks can also fool humans as long as images are only presented for a brief period of time (around 60-70ms, *Elsayed, Shankar, Cheung, Papernot, Kurakin, Goodfellow, and Sohl-Dickstein* 2018).

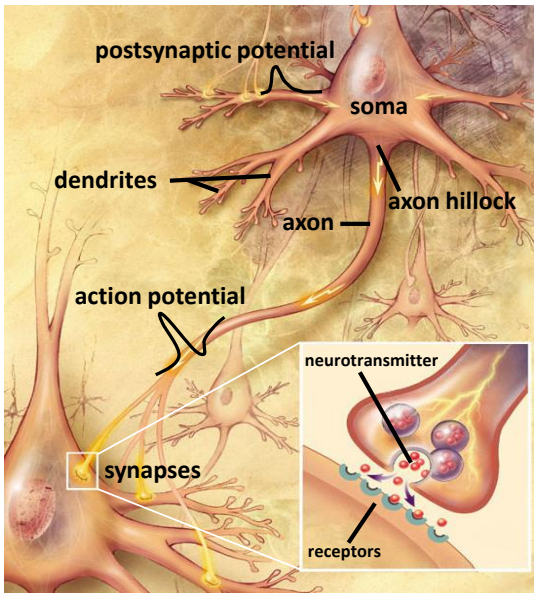

Figure 2.3.: Artistic illustration of biological neurons. A neuron is composed of its cell body, the soma, and tree-like structures called dendrites that integrate input from other neurons. Action potentials are triggered at the axon hillock and leave the neuron via the axon. Where the axon connects to other neurons, a synapse forms. At the synapse, presynaptic action potentials lead to a release of neurotransmitter into the synaptic cleft. These neurotransmitters dock to receptors of ligand-gated ion channels of the postsynaptic membrane, leading to an increased membrane conductance, i.e., passive ion flow. Since the electrical action potential is transformed into a chemical signal, this type of synapse is called "chemical synapse". Image adapted from https://en.wikipedia.org, 18.02.2020.

and introduce temporal dynamics (Fig. 2.2B):

$$\tau_\mathrm{m}\dot{u}_k = -u_k + E_\mathrm{l} + \sum_{\text{syn. } j} w_j \bar{r}_j \,, \qquad \bar{r}_j = \varphi(u_j) \,, \tag{2.2}$$

where we can interpret u_j now as a neuron's membrane potential and $\bar{r}_j$ as its instantaneous firing rate. If no synaptic input is present, the membrane potential decays to its rest value E_l called leak potential. This model works similar to a capacitance with time constant τ_m charged by input currents $\sum_{\text{syn. } j} w_j \bar{r}_j$, also called a leaky integrator. Hence, neurons are modeled as capacitors integrating over synaptic inputs $\bar{r}_j$ weighted by synaptic strengths w_j, i.e., biological synapses are reduced to scalar values measuring the strength of the synaptic connection. The membrane time constant τ_m dictates how fast the neuron can follow changes in the synaptic input. Therefore, its membrane acts as a low-pass filter of the synaptic input with characteristic time constant τ_m.

One remarkable aspect of biological neurons is that information accumulation uses analog values, but information transmission is done digitally, i.e., by forming all-or-nothing events called action potentials (or spikes) transmitted via the neuron's "output cable", the axon, to other neurons[4]. Action potentials are stereotypical strong depolarizations of the membrane potential, which basically form due to the dynamic properties of voltage-gated ion channels in the neuron's membrane, see *Gerstner and Kistler* (2002) for more details. They are followed by a ms-long phase of hyperpolarization during which it is harder to excite the neuron again as the ion channels have to return to their baseline configuration first, called

[4]Whether this is simply a consequence of metabolism and energy-efficient information transmission in biological tissue (*Roberts and Bush*, 1981) or spikes carry additional algorithmic advantages the brain utilizes is still an open question (*Pfeiffer and Pfeil*, 2018). For building AI systems, spike-based coding promises asynchronous, fast and energy-efficient computing, while most deep learning systems nowadays use analog coding for mathematical and numerical convenience (*LeCun, Bengio, and Hinton*, 2015; *Roy, Jaiswal, and Panda*, 2019).

refractory period. Thus, given its presynaptic[5] input, a neuron either spikes, influencing its postsynaptic partners, or remains silent, having no influence on other neurons.

We can further extend Eq. 2.2 to account for spikes and refractory periods, resulting in the leaky integrate-and-fire (LIF) model (Fig. 2.2C,D, *Lapique* 1907):

$$C_\mathrm{m}\frac{\mathrm{d}u_k}{\mathrm{d}t} = g_\mathrm{l}\left(E_\mathrm{l} - u_k\right) + I_\mathrm{syn}\,, \tag{2.3}$$

$$u_k(t_s) \geq \vartheta \;\;\Rightarrow\;\; u_k(t \in (t_s, t_s + \tau_\mathrm{ref}]) = \varrho\,, \tag{2.4}$$

where C_m and g_l are the membrane capacitance and leak conductance, respectively, yielding $\tau_\mathrm{m} = \frac{C_\mathrm{m}}{g_\mathrm{l}}$. If the membrane potential crosses the threshold ϑ from below at time t_s, the neuron elicits a spike and the membrane potential is clamped to the reset value ϱ during the refractory period τ_ref. Different from the previously introduced models (Eqs. 2.1 and 2.2), the LIF model features no hard-coded activation function φ anymore, but is governed by the spike-response behavior of the model, see for instance Section 3.2. The synaptic input can be modeled in two ways:

- current-based (CuBa) synapses model the direct current-flow into the soma (i.e., the main cell body) following a presynaptic spike (Fig. 2.2C):

$$I_\mathrm{syn}^\mathrm{CuBa} = \sum_{\text{syn. j}}\sum_{\text{spikes s}} w_j\kappa(t - t_s)\,, \tag{2.5}$$

  where κ is the postsynaptic response kernel, for instance given by an exponential response $\kappa(t - t_s) = \theta(t - t_s)\exp\left(-\frac{t-t_s}{\tau_\mathrm{syn}}\right)$ with synaptic time constant τ_syn and θ the Heaviside step function. Thus, a presynaptic spike leads to a current flow characterized by κ and the synaptic strength w_k that gets integrated by the neuron's soma.

- Conductance-based (CoBa) synapses model more closely how chemical synapses work. Here, a presynaptic spike leads to the release of neurotransmitters[6], which dock to receptors of ligand-gated ion channels at the postsynaptic membrane. This leads to an increased conductivity $g_{k,x}^\mathrm{syn}$, resulting in a passive ion flow that pulls the membrane potential towards the reversal potential E_x^rev of the activated ion channel (Fig. 2.2D):

$$I_\mathrm{syn}^\mathrm{CoBa} = \sum_{x\in\{\mathrm{e,i}\}} g_{k,x}^\mathrm{syn}(E_x^\mathrm{rev} - u_k)\,, \tag{2.6}$$

$$g_{k,x}^\mathrm{syn}(t) = \sum_{\text{synapses j}}\sum_{\text{spikes s}} w_{kj}\;\kappa(t - t_s)\,. \tag{2.7}$$

  The reversal potential E_x^rev is the membrane potential value at which the inside and outside of the cell membrane is balanced, i.e., no ions flow through the channels. Crossing it leads to a reversal of flow direction. Whether a response is excitatory (*increase of membrane potential*) or inhibitory (*decrease of it*) depends on the reversal potential of the ion channel.

[5] Pre- and postsynaptic means: if neuron A connects via a synapse to neuron B, A is the presynaptic neuron (*what comes before the synapse*) and B the postsynaptic neuron (*what comes after the synapse*).

[6] Examples of neurotransmitters are Glutamate for excitatory synapses and γ-aminobutyric acid (GABA) for inhibitory synapses.

The LIF model with CuBa or CoBa synapses is a widely used model in neuroscience. Even though it is rather simple and neglects many biological features like action potential generation, it is a sophisticated abstraction that reproduces the most essential aspects of real neurons. Furthermore, it is analytically tractable and can be efficiently simulated (*Rauch, La Camera, Lüscher, Senn, and Fusi*, 2003; *Gerstner and Naud*, 2009; *Teeter et al.*, 2018). LIF models can be even further extended by introducing an adaptive threshold, whose dynamics are also governed by a differential equation, as well as an exponential term that approximates the shape of action potentials whenever the neuron spikes, i.e., it also models the effect of action potentials on the membrane potential of the firing neuron. This adaptive exponential leaky integrate-and-fire (AdEx) model has been shown to faithfully reproduce the firing behavior of biological neurons, making it a very general but also more demanding model than LIF (*Brette and Gerstner*, 2005).

To summarize, when interested in a certain aspect of brain dynamics, it is reasonable to reduce the complexity of neurons to a minimal set of features needed for the purpose at hand. For instance, the metabolism of neurons is of negligible importance when describing information processing in neural networks. In case of Eq. 2.1, synaptic integration and non-linear outputs are the only features used. But this model is extendable to include additional biological details like spiking, refractory periods and membrane dynamics (Eq. 2.4) as well as synaptic integration (Eqs. 2.5 and 2.7). In general, choosing which aspects of biology are relevant for a model is a non-trivial problem and almost always involves some educated guess.

2.1.3 The neocortex: spatial structures and circuitry

In the previous sections, neurons are modeled as so-called point models, i.e., without any spatial structure. However, in the brain, many different neuron types exist, with different morphologies depending on the function and brain area the neuron resides in (Fig. 2.4A). Throughout this work, we are mostly interested in the neocortex, which is the seat of higher-level reasoning and functionality (*Frith and Dolan*, 1996). The neocortex is made of a uniform structure consisting of six stacked layers of neurons[7] (Fig. 2.4B), featuring different neuron types and connectivity patterns (*Douglas, Martin, and Whitteridge*, 1989). These layers, also called gray matter, are positioned at the surface of the cortex, while the inner parts are mostly made of white matter, i.e., axons connecting different brain areas covered in myelin[8] for insulation (*Gerstner and Kistler*, 2002), increasing the speed of passive information transmission via the axon's membrane. The most prominent neuron types in the cortex are excitatory pyramidal cells, called so due to their pyramidal-shaped soma / cell body, making up 70-80% of cortical neurons, followed by inhibitory interneurons making up the remaining 20-30% (*Markram, Toledo-Rodriguez, Wang, Gupta, Silberberg, and Wu*, 2004). Pyramidal neurons have a stereotypical structure, consisting of three integration zones: apical dendrites, basal dendrites and the soma (*Spruston*, 2008). Besides housing the cell nucleus of the neuron, the soma (with a diameter of 20–120μm, *Johns* 2014) contains

[7] Not to be confused with the term "layer" used in the deep learning literature. Cortical areas correspond to what is known as "layers" in deep learning.

[8] Myelin is white, hence white matter, while unmyelinated material like neurons is gray.

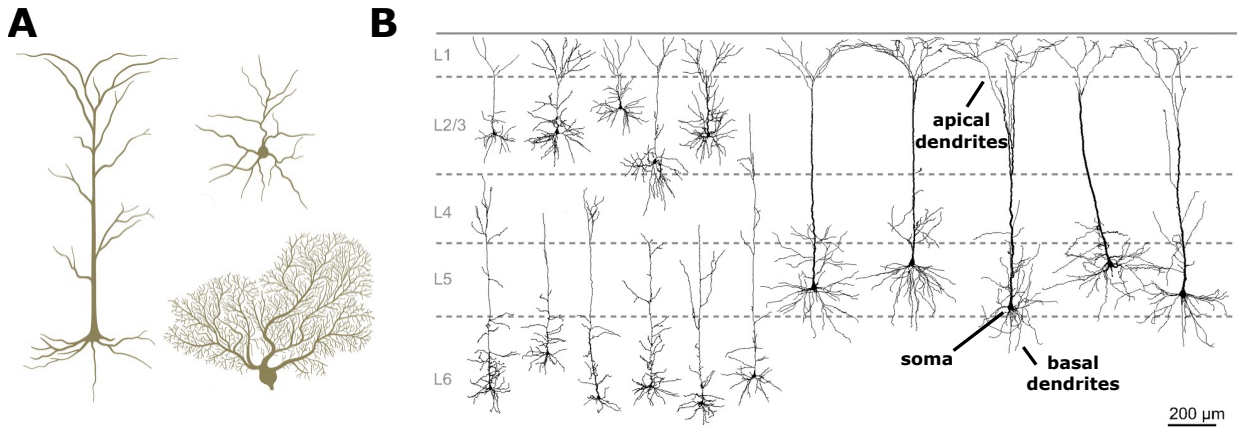

Figure 2.4.: **(A)** Three examples of different neuron types in the human brain. (left) Cortical pyramidal neuron with soma and basal dendrites at the bottom and apical dendrites at the top, (top right) granule cells found in the cerebellum, cortex and hippocampus and (bottom right) a cerebellar Purkinje cell. Image taken from *Johns* (2014). **(B)** Pyramidal neurons have two integration sites that stretch over several layers (L). Here, several L6, L5 and L2/3 pyramidal neurons are shown. The black dot in the home layer is the soma, and the tree-like structures are basal and apical dendrites. Image adapted from *Ledergerber and Larkum* (2010).

the spike initiation zone ("axon hillock"), whose mechanism we primarily model with point neurons in the previous section. Both apical and basal dendrites are tree-like structures that integrate inputs coming from other neurons. Basal dendrites are at the base of the neuron, close to the soma, and primarily integrate bottom-up information ("feedforward"), whereas apical dendrites are further away from the soma and integrate long-range top-down information ("feedback", *Larkum, Zhu, and Sakmann* 1999; *Spruston* 2008; *Petreanu, Mao, Sternson, and Svoboda* 2009; *Larkum* 2013), see Fig. 2.5A. For instance, layer 5 pyramidal neurons have their soma and basal tree in layer 5 and the apical tree in layer 1 (with apical shaft length $(739 \pm 186)\mu$m and diameter $(4.94 \pm 1.10)\mu$m), whereas layer 2/3 neurons stretch from layer 2/3 to layer 1 (with apical shaft length $(323 \pm 244)\mu$m and diameter $(2.11 \pm 1.02)\mu$m, *Ledergerber and Larkum* 2010). Since all compartments are electronically coupled, an action potential elicited at the axon hillock does not only propagate forwards through the axon, but also backwards (hence called "backpropagating action potential") into the dendritic trees (*Larkum*, 2013) to potentially drive plasticity (*Markram, Gerstner, and Sjöström*, 2012; *Urbanczik and Senn*, 2014) as well as dendritic non-linearities like Ca^{2+} plateau potentials (or "Ca^{2+} spikes", *Larkum, Zhu, and Sakmann* 1999; *Larkum* 2013)[9].

Interneurons can be divided into several classes through their morphology, but also differ in their electrical properties and connectivity to other neurons, e.g., which region of a pyramidal neuron they target (soma-targeting, dendrite-targeting, axon-targeting, dendritic-tuft-targeting cells, etc., see *Markram, Toledo-Rodriguez, Wang, Gupta, Silberberg, and Wu*

[9]The spikes used to communicate between neurons have an all-or-nothing structure (strong rise, fast decay). These spikes mainly originate from the dynamics of Na^+ and K^+ ion channels, hence they are also called "sodium spikes". In dendrites, a different kind of spikes can occur: Ca^{2+} and NMDA "spikes" with long and strong plateau potentials (*Schiller, Major, Koester, and Schiller*, 2000; *Larkum*, 2013).

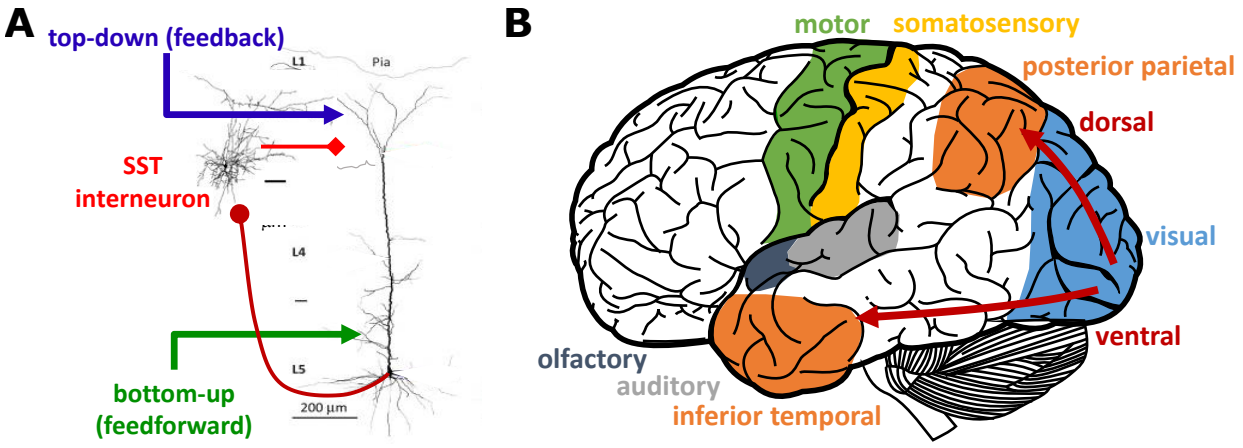

Figure 2.5.: **(A)** Different integration sites are used to accumulate different input streams. Long-distance, top-down inputs and SST interneurons project to the apical dendrites of pyramidal neurons, while bottom-up (thalamic) input projects to the basal dendrites. Figure adapted from *Larkum* (2013). **(B)** Schematic illustration of some common brain areas (many areas are missing in this illustration). Figure adjusted from https://commons.wikimedia.org/wiki/File:Brain_Surface_-Gyri.SVG (version: 11:12, 30 March 2010).

(2004) for a more detailed description). Different from pyramidal neurons, interneurons in the human neocortex are mostly inhibitory[10] and do not show a clear separation of integration zones over several layers like pyramidal neurons. Furthermore, they only connect to nearby neurons, i.e., they do not project via white matter to distant brain regions, but form local circuits (*Markram, Toledo-Rodriguez, Wang, Gupta, Silberberg, and Wu*, 2004). In Chapter 5, due to their specific connectivity, we will be mostly interested in somatostatin-expressing (SST) interneurons (representing $\approx 30\%$ of GABAergic neurons, i.e., neurons that produce the neurotransmitter γ-aminobutyric acid (GABA), *Rudy, Fishell, Lee, and Hjerling-Leffler* 2011), especially layer 2/3 Martinotti cells, which are low-threshold, fast spiking neurons with high spontaneous activity. SST interneurons are locally strongly connected, receiving input from nearby pyramidal neurons as well as top-down input (*Leinweber, Ward, Sobczak, Attinger, and Keller*, 2017), while mainly targeting the apical compartments of pyramidal neurons in layer 1 (*Markram, Toledo-Rodriguez, Wang, Gupta, Silberberg, and Wu*, 2004; *Urban-Ciecko and Barth*, 2016).

The specific shape of pyramidal neurons as well as the local circuits formed with interneurons is believed to have vast functional implications (*Spruston*, 2008; *Markram, Toledo-Rodriguez, Wang, Gupta, Silberberg, and Wu*, 2004), e.g., for perception and information coding (*Larkum*, 2013; *Takahashi, Oertner, Hegemann, and Larkum*, 2016; *Jordan, Sacramento, Petrovici, and Senn*, 2019b; *Gidon, Zolnik, Fidzinski, Bolduan, Papoutsi, Poirazi, Holtkamp, Vida, and Larkum*, 2020) as well as plasticity and learning (*Guerguiev, Lillicrap, and Richards*, 2017; *Costa, Assael, Shillingford, de Freitas, and Vogels*, 2017; *Sacramento, Costa, Bengio, and Senn*, 2018; *Wilmes and Clopath*, 2019).

On a larger scale, the brain shows small-world structure (*Muldoon, Bridgeford, and Bassett*,

[10]Some interneurons are in fact excitatory, e.g., the spiny stellate cell (SSC) found in layer 4 of primary sensory areas.

2016), i.e., strong clustering and short path lengths, and forms a network of networks (or modules) that are spatially and / or functionally separated (*Chen, He, Rosa-Neto, Germann, and Evans*, 2008; *Bullmore and Sporns*, 2009; *Meunier, Lambiotte, and Bullmore*, 2010; *Bertolero, Yeo, and D'Esposito*, 2015; *Song, Sjöström, Reigl, Nelson, and Chklovskii*, 2005). The most general form of this separation can be seen by the functional organization into cortical areas for sensory processing, i.e., visual cortex, auditory cortex, somatosensory cortex, motor cortex, etc., which further separate into smaller functional units. For instance, the visual cortex splits into two hierarchies of visual (V) areas, the dorsal (V1 – V2 – V3 – MT (V5) – parietal cortex) and ventral (V1 – V2 – V4 – inferior temporal cortex) pathway (see Fig. 2.5B). The organization into different cross-wired areas is believed to allow complex processing of sensory input – e.g., each area applies a different function to its input, and areas can be stacked or recurrently connected to form spatial and temporal hierarchies of functions – which has been the inspiration for the recent success of deep learning (*LeCun, Bengio, and Hinton*, 2015; *Schmidhuber*, 2015). In particular, the hierarchical feedforward structure and the response properties of neurons in the visual system (*Hubel and Wiesel*, 1959, 1962) have been the inspiration for convolutional neural networks (*Fukushima*, 1988; *LeCun, Bottou, Bengio, and Haffner*, 1998), the current state-of-the-art architecture for image processing (*Ciregan, Meier, and Schmidhuber*, 2012; *Krizhevsky, Sutskever, and Hinton*, 2012; *He, Zhang, Ren, and Sun*, 2016).

2.2 Towards silicon brains: neuromorphic hardware

How the brain processes information is substantially different from contemporary computer systems, for both the underlying computing architecture and the theoretical principles that govern it. The brain forms a vast network of single units (neurons) and their connections (synapses) that are simultaneously used for information processing and storage (*Hopfield*, 1982; *Indiveri and Liu*, 2015). Information is transmitted via all-or-nothing events, allowing fast, power efficient and asynchronous computations that utilize spatio-temporal coding schemes (*Thorpe, Fize, and Marlot*, 1996; *Gütig and Sompolinsky*, 2006). Thus, the brain efficiently intermixes analog (membrane potentials) and digital (action potentials) computing paradigms for power efficiency (*Sarpeshkar*, 1998), and most likely for computational effectiveness as well (*Maass*, 1997; *Sarpeshkar*, 1998; *Thorpe, Delorme, and Van Rullen*, 2001; *Buesing, Bill, Nessler, and Maass*, 2011; *Mostafa*, 2017). Different from the brain, current computer technology is based on the von Neumann architecture, which separates processing units from memory storage (*Von Neumann*, 1945) – with the consequence that processing speed is fundamentally limited by (i) the access speed of said memory storage and (ii) the transmission speed between processing units and storage, commonly known as the von Neumann bottleneck (*Miller*, 2011; *McKee et al.*, 2004; *Wulf and McKee*, 1995; *Riley*, 1987), see Fig. 2.6A. The von Neumann architecture is nowadays used to implement realizations of the universal Turing machine (*Boole*, 1847; *Turing*, 1937), a theoretical construct able to implement arbitrary algorithms, i.e., sequential instruction sequences, which to our current knowledge differs considerably from how the brain operates on a mechanistic level (*Von Neumann and Kurzweil*, 1958; *Zylberberg, Dehaene, Roelfsema, and Sigman*, 2011).

The consequences of these architectural differences become apparent when comparing the average power consumption of a human brain and the hardware used to, e.g., train AlphaGo, Deepmind's AI system that defeated Lee Sedol in the game of Go (*Deepmind*, 2016; *Silver et al.*, 2017): while the brain consumes merely[11] 20W (*Mink, Blumenschine, and Adams*, 1981), the final version of AlphaGo used 8 GPUs (consuming typically around $100 - 300$W per GPU) and 48 CPUs (around $50 - 100$W per CPU) on a single machine[12] (*Silver et al.*, 2016), while only being capable of playing Go.

This difference in performance spawned the neuromorphic doctrine: building hardware that mimics the brain to harness its extreme parallelism and asynchronous nature for power efficiency and computing speed (*Mead*, 1990; *Indiveri et al.*, 2011; *Schuman, Potok, Patton, Birdwell, Dean, Rose, and Plank*, 2017; *Roy, Jaiswal, and Panda*, 2019), i.e., building the substrate to hold AI in the future. In neuromorphic hardware, the von Neumann architecture is replaced by a more distributed design, where neurons (or processing cores simulating a certain number of neurons) and synapses (or local memory storing neuron and synapse parameters as well as instructions) are intertwined, forming a highly parallel and distributed network. In consequence, individual components of the hardware are ideally constrained to information that is only locally available, similar to the brain, avoiding costly read-and-write operations to global memory. Neuromorphic hardware is either focused on digital implementations (*Akopyan et al.*, 2015; *Davies et al.*, 2018; *Mayr, Hoeppner, and Furber*, 2019; *Pei et al.*, 2019), where neurons are still simulated with conventional processing units (so-called "digital neuromorphic cores") or analog ones (*Mead*, 1990; *Liu, Kramer, Indiveri, Delbra, Douglas et al.*, 2002; *Schemmel, Fieres, and Meier*, 2008; *Schemmel, Brüderle, Grübl, Hock, Meier, and Millner*, 2010; *Indiveri et al.*, 2011; *Benjamin et al.*, 2014; *Friedmann, Schemmel, Grübl, Hartel, Hock, and Meier*, 2016), where neurons and synapses are realized in physical form, i.e., the physical substrate itself implements the differential equations of models of biological neural networks. Thus, instead of using transistors as "switches" (a gating voltage is used to control whether current flows through the transistor) to build logic gates for Boolean algebra – as in digital computing – the physics of the substrate perform the computations by virtue of their dynamics in analog computing.

Such analog devices are often realized using complementary metal-oxide-semiconductor (CMOS) technology, which can be operated in two different modes: subthreshold and suprathreshold, depending on whether the applied gating voltage is sufficient to allow currents to flow through the transistor or not (turning it either "on" or "off"). In the subthreshold mode, the CMOS transistor is in the "off" state and only very weak currents flow, allowing the construction of neuromorphic devices with ultra low power consumption and real-time modelling capacity (*Indiveri, Chicca, and Douglas*, 2006). In the suprathreshold regime, the transistor has a continuous response instead, that is, increasing the applied voltage leads to an increased current flow through the transistor. Neuromorphic devices using transistors in the suprathreshold region offer low energy consumption and large acceleration of the emulated system due to the characteristically low time constants of the analog components, e.g., the BrainScaleS–1 neuromorphic system used in Section 3.4.6 is accelerated by a factor

[11] Mostly used for synaptic transmission (*Harris, Jolivet, and Attwell*, 2012; *Li and Van Rossum*, 2019).
[12] Or 176 GPUs and 1202 CPUs distributed on several machines.

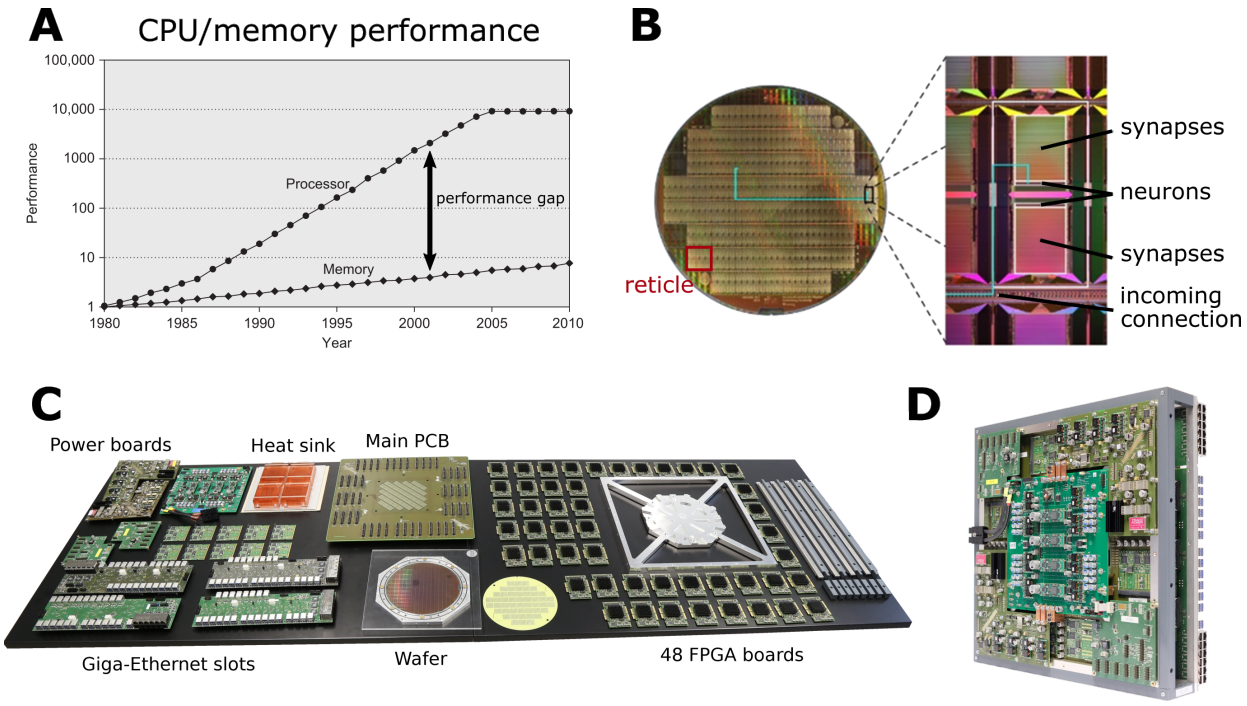

Figure 2.6.: **(A)** The von Neumann bottleneck is a result of memory increasing slower in performance than processors over the years, here shown as DRAM accesses and processor memory requests per second. Thus, the total speed of a computer is limited by memory operations, since both instructions and data have to be loaded from memory. Processor performance is given per core, which stagnated in 2005 and multicore systems started being built. Image adapted from *Hennessy and Patterson* (2012). **(B)** The BrainScaleS–1 wafer (left) and its center piece, the HICANN neuromorphic chip (right). Image adapted from *Kungl et al.* (2019b). **(C)** Individual components of the BrainScaleS–1 wafer scale system, including support infrastructure like cooling and power boards. **(D)** A single BrainScaleS–1 wafer in its final assembled form.

of $10^3 - 10^5$ compared to biology (*Schemmel, Fieres, and Meier*, 2008; *Friedmann, Schemmel, Grübl, Hartel, Hock, and Meier*, 2016). Increasing the gating voltage further, the transistor current saturates – in this mode, it can be used as a switch to implement digital logic circuits, with small and large voltages leading to minimum or maximum currents. Apart from CMOS technology, novel materials like "memristors", a type of circuit element that has a history-dependent resistance (*Xia and Yang*, 2019), are currently also heavily investigated in the field of neuromorphic engineering. However, since we are only concerned with CMOS-based devices in this thesis, we omit a detailed description of other approaches like memristive devices here (see, e.g., *Lee et al.* 2019 for a review).

Despite the advantages, working with analog neuromorphic hardware comes with challenges of its own. For instance, due to variations in the manufacturing process ("fixed-pattern noise"), each hardware neuron and synapse is unique, different from simulated neural networks where commonly all neurons share the same set of neuron parameters and behave identically. In addition, both the resolution and control over neuron parameters, i.e., setting exact values, is limited. Since components of neural networks occupy physical space, it is not possible to map arbitrary networks onto the chip, and since transmission buffers are

used for communication, spike-loss can occur due to congestion under high spike activity. To some extent, the aforementioned challenges can be regarded as attributes of the substrate, intended by the underlying design, and mirror the challenges the brain faces, i.e., performing computations with unreliable connections and non-identical, low precision circuit elements. However, there also exist disturbance effects that are not intended, like cross-talk between neighboring circuits, i.e., neurons can behave differently depending on whether neighboring circuits are active or not in hardware. Although some controllability can be regained by calibrating hardware circuits and blacklisting defect components (*Koke*, 2017; *Kleider*, 2017), all these effects and limitations remain to some extent (*Schmitt et al.*, 2017; *Kungl et al.*, 2019b; *Wunderlich et al.*, 2019).

One such analog implementation used in Section 3.4.6 is the BrainScaleS–1 physical wafer-scale system (*Schemmel, Fieres, and Meier*, 2008; *Schemmel, Brüderle, Grübl, Hock, Meier, and Millner*, 2010), see Fig. 2.6B-D. The BrainScaleS–1 system is a mixed-signal analog implementation of biologically inspired neural networks using 180nm CMOS technology (suprathreshold), realizing a modular version of the AdEx neuron model, i.e., individual terms like adaptation can be turned off. In fact, in this thesis we only use the CoBa LIF functionality. It features analog neurons and synapses, with flexible connectivity capability and digital spike transmission. The key-component of the system, the HICANN chip (Fig. 2.6B), is designed for wafer-scale integration, potentially enabling the emulation of large neural networks with up to $1.8 \cdot 10^5$ neurons and $4 \cdot 10^7$ synaptic connections. A single wafer consists of 48 reticles[13], each containing 8 HICANN chips (Fig. 2.6B). Every HICANN chip features 512 analog neurons (so-called "denmems", dendritic membranes) with up to 220 possible synaptic input connections (synaptic fan-in) per denmem. Furthermore, denmems can be combined to form larger neurons with higher maximum synaptic fan-in. Flexible connectivities are offered by a synapse array: the digital input signals enter the synapse array through a synapse driver, which redirects them to the correct synapse line where output pulses with strengths given by the synaptic weights (stored in local SRAM with 4bit precision) are generated. This leads to a postsynaptic conductance at the postsynaptic neuron according to the CoBa LIF equation of motion (Eq. 2.7). Analog parameters that store neuron parameters and scaling factors for the weights are stored in floating-gates (*Srowig, Loock, Meier, Schemmel, Eisenreich, Ellguth, and Schüffny*, 2007) and are set up through field-programmable gate arrays (FPGA), with one FPGA per reticle (Fig. 2.6C). Additional connections added in a postprocessing stage (*Zoschke, Güttler, Böttcher, Grübl, Husmann, Schemmel, Meier, and Ehrmann*, 2017) enable communication between HICANN-chips on the whole wafer. Thus, the BrainScaleS–1 system offers a physical implementation of the AdEx neuron model, fully specified by a set of neuron parameters that can be set via FPGAs and a synapse array that allows programmable neuron connectivity.

Apart from imperfections of the underlying neuronal substrate, similarly to the brain, neuromorphic hardware has to respect physical constraints like locality and finite transmission times (delays). This introduces an additional challenge faced both by digital and analog realizations of neural networks: how to perform computations with a device that, architecturally, resembles the brain? As of now, the two most pressing problems on the

[13] A reticle is the maximum area that can be simultaneously exposed during photolithography.

algorithmic side are (i) finding local learning rules that perform as well as error backpropagation used in deep learning[14] (*Lillicrap, Cownden, Tweed, and Akerman*, 2016; *Guerguiev, Lillicrap, and Richards*, 2017; *Sacramento, Costa, Bengio, and Senn*, 2018; *Kaiser, Mostafa, and Neftci*, 2018; *Whittington and Bogacz*, 2019; *Illing, Gerstner, and Brea*, 2019; *Lillicrap and Santoro*, 2019) and (ii) finding coding schemes that utilize the asynchronous and all-or-nothing nature of action potentials efficiently (*Gerstner*, 1998; *Buesing, Bill, Nessler, and Maass*, 2011; *Maass*, 2016; *Mostafa*, 2017; *Pfeiffer and Pfeil*, 2018; *Davies*, 2019). Hence, finding algorithms that both implement functionality equivalent or even more powerful than current AI standards and take the form of biological neural networks, i.e., obeying physical constraints as well as being robust against imperfections and variability of the underlying neuronal substrate, is at the heart of current neuromorphic computing research which mimics the brain, but still lacks sufficient computational understanding to harness its form for function.

2.3 Outline: harnessing function from form

The remaining parts of this thesis are separated into two large chapters, one focusing on spike-based coding (Chapter 3) and the other on learning in biological neuronal systems (Chapter 5), bridged by a brief intermediate chapter (Chapter 4).

In Chapter 3, we first discuss the preliminaries of Bayesian computing in the brain (Section 3.1) and review a spike-based model realizing such computations (Section 3.2). Afterwards, we demonstrate that this model can be augmented with a simple biological mechanism to improve its generative performance in spatio-temporal prediction tasks (Section 3.3). Finally, we show how spike-based probabilistic inference can be realized in a self-sustained and closed system by embedding single networks into an ensemble of functional networks (Section 3.4) – like a "heat bath" for neurons. This way, networks in the ensemble can utilize the background activity of other networks to perform probabilistic computations, while also providing its functional spikes to the ensemble as "noise" in return.

In Chapter 5, we discuss whether the famous error backpropagation algorithm (*Linnainmaa*, 1970; *Werbos*, 1982; *Rumelhart, Hinton, and Williams*, 1986) can be realized in a biologically plausible way. After reviewing error backpropagation (Section 5.1) and its challenges concerning a biological (or neuromorphic) realization (Section 5.1.2), we propose a top-down approach that derives neuronal and synaptic dynamics from first principles (Section 5.2.4). The derived dynamics implement an approximate version of real-time error backpropagation, which lends itself to a biologically plausible realization using advanced neuronal firing responses, dendritic compartments and local microcircuits.

Both models presented in this thesis tie computational theory and physical realization closely together, making it possible to transfer the presented network structures to adequate neuronal substrates while preserving functionality, as demonstrated in Section 3.4.6 and further discussed in Chapter 6.

[14] This is obviously also of great interest for deep learning in general, see, e.g., *Jaderberg, Czarnecki, Osindero, Vinyals, Graves, Silver, and Kavukcuoglu* (2017).

3 | Spike-based coding as deterministic Bayesian inference

Our cortex builds a predictive, internal model of the outside world to navigate ourselves through reality. This enables us to not only passively integrate sensory information, but to perform mental simulations as well, e.g., to play through several scenarios (to "think") before we act (*Keller and Mrsic-Flogel*, 2018). But how can neural circuits shape such a model under the constraints imposed by biology and physics? This will be the focus of the following sections, where we first discuss neuronal behavior that is believed to be a hallmark of a Bayesian computing scheme suitable to model the environment in a probabilistic way (Section 3.1). We then present a spike-based sampling framework that allows the realization of stochastic inference under in-vivo-like conditions (Sections 3.2 to 3.4). Consequently, this framework enables a self-sustained and self-consistent implementation of spike-based sampling on neuromorphic hardware (Section 3.4).

3.1 The Bayesian brain hypothesis

> *It is critical to realize, however, that variability and uncertainty go hand in hand: if neuronal variability did not exist, that is, if neurons were to fire in exactly the same way every time you saw the same object, then you would always know with certainty what object was presented.*
>
> **Wei Ji Ma et al.**, Nature Neuroscience, 2006
> *"Bayesian inference with probabilistic population codes"*

3.1.1 Neuronal noise

One of the most prominent features of the mammalian cortex is the strongly varying response of neurons to repeated presentations of identical stimuli in vivo (Fig. 3.1A) that occurs both in anesthetized and awake animals (*Henry, Bishop, Tupper, and Dreher*, 1973; *Schiller, Finlay, and Volman*, 1976; *Tolhurst, Movshon, and Dean*, 1983; *Vogels, Spileers, and Orban*, 1989; *Snowden, Treue, and Andersen*, 1992; *Arieli, Sterkin, Grinvald, and Aertsen*, 1996; *Azouz and Gray*, 1999; *Yarom and Hounsgaard*, 2011). The variability with respect to the mean response strength over trials (i.e., number of spikes) is often characterized by a power

function var $\approx \alpha \cdot \text{resp}^{\beta}$ with $\alpha = \mathcal{O}(1)$ and $\beta \approx 1$ (*Tolhurst, Movshon, and Dean*, 1983; *Vogels, Spileers, and Orban*, 1989; *Snowden, Treue, and Andersen*, 1992), see Fig. 3.1B-C, revealing considerable differences between identical experimental trials. However, even though individual trials vary strongly, the average response of cells over many identical trials is very consistent and reproducible (*Arieli, Sterkin, Grinvald, and Aertsen*, 1996). Consequently, the trial-to-trial variability observed in vivo has, for a long time, been interpreted as a nuisance or, simply put, pure "noise" the cortex has to deal with in order to perform its functions (*Shadlen and Newsome*, 1998; *Carandini*, 2004). In contrast, experiments done in vitro show that neurons are mostly deterministic units that reliably transform neuronal presynaptic input into temporal spike sequences – with a temporal spike precision between trials of less than a millisecond (*Mainen and Sejnowski*, 1995). This naturally raises the question about the origin of neuronal variability observed in vivo.

Since the spike-generating mechanism behaves mostly deterministic in vitro, we can exclude sources like ion channel noise from our consideration. Thermal noise only has a very weak effect on the membrane potential of neurons[1] and can hence be neglected. What remains is synaptic noise, both in the form of synaptic transmission noise and synaptic background noise. The former results from the unreliable release of neurotransmitters at individual release sites, resulting in an unreliable transmission of presynaptic spikes (*Allen and Stevens*, 1994). However, this contribution largely averages out when integrating over several release sites, contacts and connections (*Zador*, 1998; *Markram et al.*, 2015). The latter is due to the huge amount of presynaptic partners each neuron has – on the order of $10^3 - 10^4$ neurons (*Peters*, 1987) – resulting in a strong "background bombardment" each neuron is exposed to under in vivo conditions (*Arieli, Sterkin, Grinvald, and Aertsen*, 1996; *Tsodyks, Kenet, Grinvald, and Arieli*, 1999). In this case, the trial-to-trial variability observed in experiments is actually due to the changing state of the whole cortical network between trials – i.e., the network is not only driven by the external input imposed by the experimenter, but also strongly depends on the internally generated activity. In simpler terms: the "background noise" changes between trials, and the initial conditions of the different experimental trials are not identical. In fact, such irregular behavior has also been reproduced in simulation studies of large balanced networks[2] without the inclusion of any other noise sources, see e.g. *Van Vreeswijk and Sompolinsky* (1996); *Vreeswijk and Sompolinsky* (1998); *Brunel* (2000); *Mehring, Hehl, Kubo, Diesmann, and Aertsen* (2003); *Vogels, Rajan, and Abbott* (2005).

At first glance, one might think that neuronal noise prohibits coding schemes that rely on precise spike timings and one has to fall back to population-based codes, where noise mostly averages out. However, recent studies suggest that the observed trial-to-trial variability might actually be a hallmark of probabilistic neuronal computation, enabling the brain to

[1] If we model the membrane of a neuron as a simple RC circuit, then the root mean square (RMS) deviation of the capacitor's potential V_C due to thermal motion is given by $V_C^{\text{RMS}} = \sqrt{\frac{k_B T}{C}}$, where k_B is the Boltzmann constant. For room temperature $T = 300\text{K}$ and a typical membrane conductance C on the order of μF (*Golowasch and Nadim*, 2013), we obtain $V_C^{\text{RMS}} = \mathcal{O}(10^{-8})\text{V}$, much smaller than the typical dynamical range of biological neurons (mV).

[2] A balanced network consists of an excitatory and inhibitory population, where the total excitatory and inhibitory postsynaptic currents each neuron receives cancel on average.

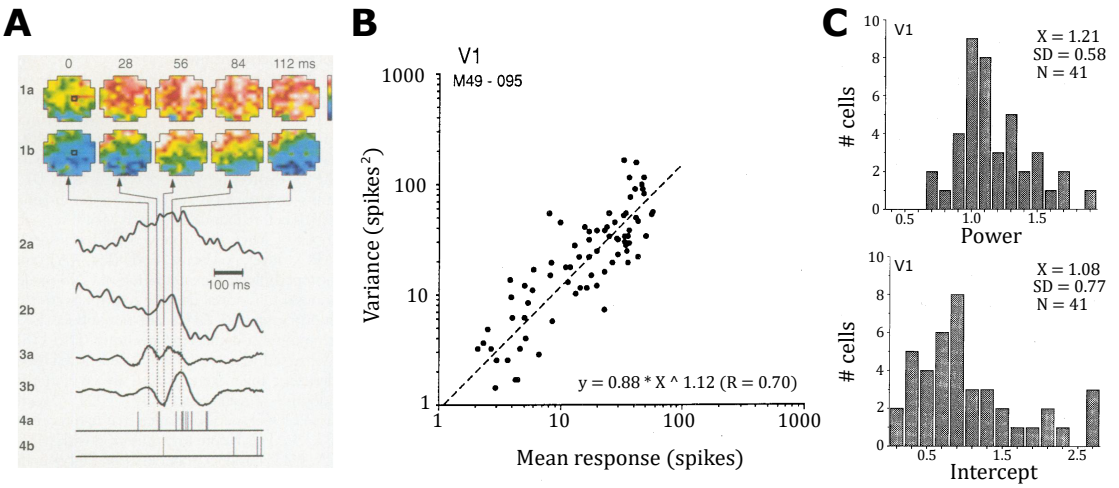

Figure 3.1.: **(A)** Optical and electrophysiological recordings of a small area in the primary visual cortex of a cat for two presentations (a,b) of the same stimulus. The trial-to-trial variability can be seen in all recordings: (1) Real-time optical imaging showing the response in a 2mm by 2mm area of cortex, where red corresponds to a response above and blue below mean activity. (2) Exemplary optical trace of a single photodiode cell from (1). (3) Local field potential in the same area. (4) Spike train recording from a neuron in the same area. Taken from *Arieli et al.* (1996). **(B)** Response variance as a function of the mean response (log-log plotted) for a cell from area V1 of an alert macaque monkey while being presented with different visual patterns. Each point corresponds to a certain visual pattern. Mean and variance were calculated over trials. **(C)** Distribution of fit parameters (top) β and (bottom) α over all 41 recorded cells. Figures adapted from *Snowden et al.* (1992).

cope with incomplete, unreliable and uncertain sensory stimuli in a near Bayes-optimal way (*Körding*, 2007; *Fiser, Berkes, Orbán, and Lengyel*, 2010) – even when spike-coding schemes are employed. This is generally known as the "Bayesian brain hypothesis".

3.1.2 Noise as the hallmark of probabilistic reasoning

That the observed neural response variability might be the hallmark of a Bayesian computation scheme has already been discussed as early as 2003, when *Hoyer and Hyvärinen* (2003) proposed that the visual cortex implements Monte Carlo sampling of a posterior distribution in order to cope with ambiguous or incomplete visual stimuli. In such a computational model, neuronal activity is assumed to represent either probabilities or samples from a posterior distribution $p(\boldsymbol{y}, \boldsymbol{x})$ that encodes the probability of causes $\boldsymbol{y}$ and visual observations $\boldsymbol{x}$ to occur. When constrained by visual stimuli $\boldsymbol{x}$, this can be interpreted as a probabilistic generative model that estimates which hidden variables $\boldsymbol{y}$ caused the stimulus via the posterior $p(\boldsymbol{y}|\boldsymbol{x})$.

For instance, we might be faced with the problem of classifying objects in an image. In this case, $\boldsymbol{x}$ is an array containing the image's pixel values that can either be revealed completely, only be partly shown or even contain corrupt pixels, and $\boldsymbol{y}$ is a vector encoding possible class labels. Before making any observations, our knowledge about how likely it is to observe the state $(\boldsymbol{y}, \boldsymbol{x})$ is given by the prior probability distribution $p(\boldsymbol{x}, \boldsymbol{y})$. After making a single

observation x_0 (e.g., revealing a single pixel value), the probability distribution is updated to the marginal posterior distribution (with $\backslash 0$ meaning all but index 0)

$$p(\boldsymbol{y} \,|\, x_0) = \int p(\boldsymbol{y}, \boldsymbol{x}_{\backslash 0} \,|\, x_0) \, \mathrm{d}\boldsymbol{x}_{\backslash 0} \tag{3.1}$$

adjusting our belief and uncertainty about the prediction $\boldsymbol{y}$ conditioned on the observation x_0. In fact, if we do not marginalize, such a model is able to also predict yet unknown observations $\boldsymbol{x}_{\backslash 0}$ via the posterior distribution $p(\boldsymbol{y}, \boldsymbol{x}_{\backslash 0} \,|\, x_0)$, i.e., guess how the whole image looks like. If we obtain more observations, the shape of the posterior distribution is further constrained until all observations are obtained and we arrive at the full posterior distribution $p(\boldsymbol{y} \,|\, \boldsymbol{x})$, i.e., the probability distribution over class labels given the whole image. The distribution can be even further constrained if knowledge about the class label $\boldsymbol{y}$ is available, e.g., after obtaining the information that one of the classes is definitely not in the image.

This process of updating one's belief when faced with accumulating observations in a probabilistic way is generally known as Bayesian inference. The posterior distribution can be evaluated either by calculating the distribution analytically or by sampling from it. Calculating it is, at least for reasonably interesting distributions, intractable as it requires calculating the partition function – meaning we have to evaluate the probability function for all possible states. The computational complexity of this task grows exponentially with the number of random variables, hence being insurmountable. Sampling is a computational method that allows an approximate evaluation of the probability distribution without knowledge of the partition function. Instead of calculating probabilities directly, we generate random numbers $\boldsymbol{y}$ that are distributed according to the distribution $p(\boldsymbol{y}|\boldsymbol{x})$ we want to evaluate, i.e., we slowly build up a histogram resembling $p(\boldsymbol{y}|\boldsymbol{x})$ by drawing samples $\boldsymbol{y}^i$. The samples will mostly lie in regions of high probability, and thus not every possible state has to be visited in order to get a good approximation of the overall probability distribution. One of the most famous sampling algorithms is the Metropolis-Hastings algorithm (*Hastings*, 1970), which is a Markov chain Monte Carlo (MCMC) algorithm. In MCMC, we generate a sequence of states (or samples) $\boldsymbol{y^1}, \boldsymbol{y^2}, ..., \boldsymbol{y^N}$ (see Fig. 3.2A) that follows two properties:

1. The transition probability $P(\boldsymbol{y'}|\boldsymbol{y})$ to change the state (i.e., generate a sample that is different from the current one) only depends on the current state $\boldsymbol{y}$ (Markov chain).

2. The transition probability is evaluated via drawing a random number (Monte Carlo).

Furthermore, two properties have to be met such that the sequence obtained from the MCMC follows the desired distribution after collecting enough samples:

1. The chain has to be non-cyclic and capable of reaching every possible state $\boldsymbol{y'}$ (ergodicity).

2. The distribution has to be invariant under the transition operator $p(\boldsymbol{y'}|\boldsymbol{x}) = \int \mathrm{d}\boldsymbol{y} \, P(\boldsymbol{y'}|\boldsymbol{y}) p(\boldsymbol{y}|\boldsymbol{x})$, i.e., the distribution $p(\boldsymbol{y}|\boldsymbol{x})$ is the unique stationary distribution under transitions $P(\boldsymbol{y'}|\boldsymbol{y})$.

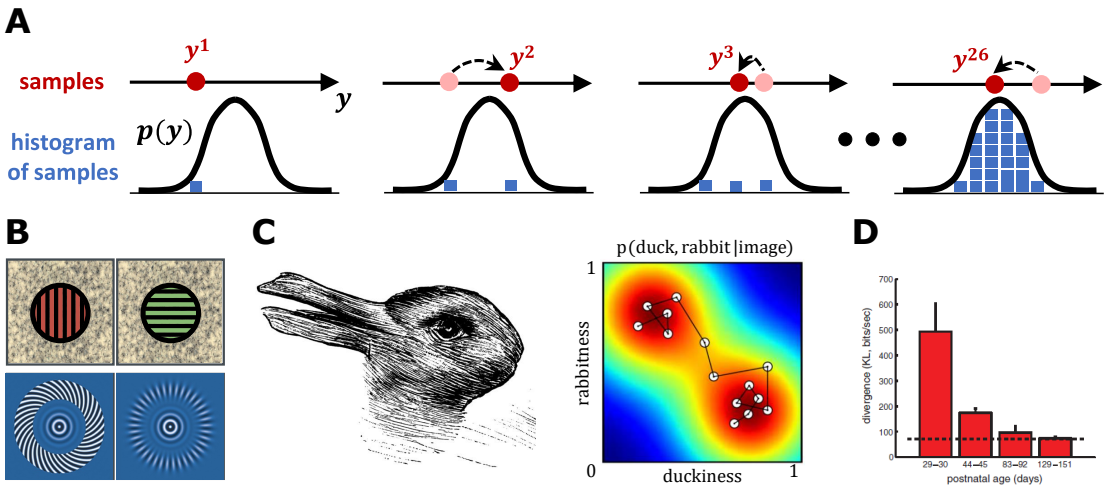

Figure 3.2.: **(A)** Illustration of Markov chain Monte Carlo (MCMC) sampling. In every sample step, a realization of y is drawn, such that for a large number of samples, the collected values follow the underlying target distribution $p(y)$ sampled from. **(B)** Illustration of binocular rivalry. The images have to be viewed by crossing the eyes to superimpose the two half-images binocularly. The perceived stimulus switches between the two images, i.e., red and green dot or circular and radial patterns. Image taken from *Blake and Logothetis* (2002). **(C)** Phenomena like binocular rivalry or bistable (ambiguous) stimuli that switch between several plausible interpretations (here: duck and rabbit, left image) can be explained by sampling-based Bayesian inference: perceptual switches correspond to jumps in the MCMC between interpretations of high probability (right). Image adapted from *Petrovici* (2016). **(D)** During development of ferrets, spontaneous activity and the evoked response to natural visual stimuli increasingly match. This is shown here via the Kullback-Leibler divergence (*Kullback and Leibler*, 1951), which measures how similar the distribution of spontaneous and evoked responses are. Image taken from *Berkes et al.* (2011).

For instance, the Metropolis-Hastings algorithm generates a chain of samples in the following way:

1. Assume the current sample is $\boldsymbol{y}^i$.
2. A new sample $\boldsymbol{y'}$ is proposed from a proposal distribution $g(\boldsymbol{y'}|\boldsymbol{y}^i)$ that is arbitrary, but has to guarantee that every possible state can be reached.
3. After a new sample has been proposed, the acceptance ratio is calculated as $\alpha(\boldsymbol{y'}, \boldsymbol{y}^i) = \min\left(1\,,\, \frac{p(\boldsymbol{y'}|\boldsymbol{x})g(\boldsymbol{y}^i|\boldsymbol{y'})}{p(\boldsymbol{y}^i|\boldsymbol{x})g(\boldsymbol{y'}|\boldsymbol{y}^i)}\right)$.
4. Draw a uniformly distributed random number U from the interval $[0, 1]$. If $U \leq \alpha(\boldsymbol{y'}, \boldsymbol{y}^i)$, the new sample is $\boldsymbol{y}^{i+1} = \boldsymbol{y'}$. Otherwise, it is $\boldsymbol{y}^{i+1} = \boldsymbol{y}^i$.

The previously mentioned transition probability is, in this case, given by $P(\boldsymbol{y'}|\boldsymbol{y}^i) = \alpha(\boldsymbol{y'}, \boldsymbol{y}^i)g(\boldsymbol{y'}|\boldsymbol{y}^i)$, and partition functions do not have to be calculated since only quotients of $p(\boldsymbol{y}|\boldsymbol{x})$ enter. It can be shown that this way of generating and accepting samples fulfills all properties discussed above[3]. If we assume that the cortex performs sampling-based Bayesian

[3]The first property, ergodicity, is guaranteed by choosing the proposal distribution adequately. For

inference, the observed response variability is a necessary consequence of the stochastic sampling process – generating a MCMC that samples from the (un)constrained posterior distribution – as each realization of the sampling chain will be different.

Apart from the response variability in the cortex, further evidence for a sampling-based computation scheme can be found in experimental studies. For instance, it is well known that for mammals, when the visual stimulus is insufficient for a definite interpretation, the visual perception starts switching between rivaling interpretations (*Knill and Richards*, 1996; *Blake and Logothetis*, 2002; *Brascamp, Van Ee, Noest, Jacobs, and van den Berg*, 2006). This can, for instance, be seen in binocular rivalry, where the left and right eye receive different visual patterns (see Fig. 3.2B). Instead of merging both stimuli, the perception switches between the two stimuli at random intervals, even if one of the stimuli might be more important than the other. Interestingly enough, a similar switching behavior has also been observed on the neuronal level for such phenomena (*Logothetis and Schall*, 1989). Bi-stable images, like the Necker cube or Rabbit-Duck image (Fig. 3.2C, left), lead to a similar phenomenon, where the visual perception switches between two possible interpretations of the image. Such a behavior can be explained as sampling from a posterior distribution that is constrained to be bi-modal by the visual input, where each mode corresponds to one of the two possible interpretations. Switching of the perception then occurs when the sampling chain jumps, with the help of noise, from one mode to the other (see Fig. 3.2C, right).

In addition, studies with awake ferrets show that during development, spontaneous activity in the visual cortex more and more resembles the averaged evoked activity in response to natural stimuli (*Berkes, Orbán, Lengyel, and Fiser*, 2011). That is, in older ferrets, the spontaneous activity can be interpreted as representing samples from a prior distribution $p(\boldsymbol{y}) = \int \mathrm{d}\boldsymbol{x}\, p(\boldsymbol{y}, \boldsymbol{x})$, which can be reconstructed by averaging over the evoked responses $p(\boldsymbol{y}) \approx \int \mathrm{d}\boldsymbol{x}\, p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$, with $p(\boldsymbol{x})$ being the natural stimuli distribution (see Fig. 3.2D) – as expected from a probabilistic coding scheme where $p(\boldsymbol{y}) = \int \mathrm{d}\boldsymbol{x}\, p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$.

Even though the evidence for a Bayesian computation scheme in the cortex is becoming stronger, it is not at all clear how such computations might be implemented on a neuronal level. Beside sampling-based implementations, probabilistic population codes have been proposed where the activity of a population encodes moments of a posterior probability distribution (*Ma, Beck, Latham, and Pouget*, 2006). As a sampling-based approach, so-called Gaussian Scale Mixtures (*Wainwright and Simoncelli*, 2000), where sampling is done on the level of the membrane potentials or rates, have been used to model Bayesian inference in the visual cortex and were demonstrated to reproduce several biological phenomena like trial-to-trial variability, transient overshoots on stimuli on-set and cortical oscillations originating from recurrently connected excitatory and inhibitory neural populations (*Orbán, Berkes, Fiser, and Lengyel*, 2016; *Aitchison and Lengyel*, 2016; *Echeveste, Aitchison, Hennequin, and Lengyel*, 2019). A proposal on how Bayesian inference might be implemented with spike-based coding has first been done in *Buesing, Bill, Nessler, and Maass* (2011); *Pecevski, Buesing, and Maass* (2011), where a MCMC was constructed that samples from a Boltzmann

the second property, invariance of the sampled distribution, we first calculate $P(\boldsymbol{y'}|\boldsymbol{y}^i)p(\boldsymbol{y}^i|\boldsymbol{x}) = \min\left(p(\boldsymbol{y}^i|\boldsymbol{x})g(\boldsymbol{y'}|\boldsymbol{y}^i)\,,\, p(\boldsymbol{y'}|\boldsymbol{x})g(\boldsymbol{y}^i|\boldsymbol{y'})\right) = P(\boldsymbol{y}^i|\boldsymbol{y'})p(\boldsymbol{y'}|\boldsymbol{x})$, a condition called "detailed balance". From this, we get $\int \mathrm{d}\boldsymbol{y}^i\, P(\boldsymbol{y'}|\boldsymbol{y}^i)p(\boldsymbol{y}^i|\boldsymbol{x}) = \int \mathrm{d}\boldsymbol{y}^i\, P(\boldsymbol{y}^i|\boldsymbol{y'})p(\boldsymbol{y'}|\boldsymbol{x}) = p(\boldsymbol{y'}|\boldsymbol{x}) \int \mathrm{d}\boldsymbol{y}^i\, P(\boldsymbol{y}^i|\boldsymbol{y'}) = p(\boldsymbol{y'}|\boldsymbol{x})$, as required.

distribution with N binary units (where the binary states represent spiking '1' and being silent '0') that have refractory periods. This model was further expanded to networks of LIF neurons with CoBa or CuBa synapses (*Petrovici, Bill, Bytschok, Schemmel, and Meier*, 2016; *Probst, Petrovici, Bytschok, Bill, Pecevski, Schemmel, and Meier*, 2015; *Neftci, Das, Pedroni, Kreutz-Delgado, and Cauwenberghs*, 2014), building a bridge between probabilistic inference with binary random variables and spike-based, biological models of neural networks. Compared to probabilistic population codes or Gaussian Scale Mixtures that are based on continuous variables (membrane potentials or rates), the attractive feature of such spiking models is that spikes naturally take the role of representing samples from a binary posterior distribution.

3.2 Stochastic inference in spiking neural networks

In the following, we will briefly review sampling-based inference in networks of LIF neurons, which forms the basis for the following studies in this chapter. The discussed results are mainly from *Petrovici, Bill, Bytschok, Schemmel, and Meier* (2016); *Petrovici* (2016).

3.2.1 The high-conductance state

CoBa LIF neurons are, by definition of the model, deterministic (see Eq. 2.3). Thus, if the steady state of the free membrane potential (FMP) is below threshold ϑ, the neuron will never elicit a spike, and when it is above threshold, the neuron will burst with a frequency ν that depends both on the refractory period τ_{ref} and the time needed to cross the threshold again starting from the reset potential ϱ, i.e.,

$$\nu = \left(\tau_{\text{ref}} + \tau_{\text{m}} \ln \frac{\varrho - E_{\text{l}} - \frac{I^{\text{ext}}}{g_{\text{l}}}}{\vartheta - E_{\text{l}} - \frac{I^{\text{ext}}}{g_{\text{l}}}} \right)^{-1} , \tag{3.2}$$

which approaches the maximum firing rate of $\nu_{\max} = \tau_{\text{ref}}^{-1}$ if the external stimulating current $I^{\text{ext}} \to \infty$ goes to infinity (Fig. 3.3A). The relation $\nu(I^{\text{ext}})$ is also known as activation or response function.

In this deterministic state, we are not able to show that LIF networks sample from any probability distribution. However, this changes when each LIF neuron is exposed to a high-frequency bombardment of independent excitatory and inhibitory Poisson spike input η, satisfying $\langle \eta \rangle = \nu_{\text{p}} = \text{const.}$ and $\langle \eta(t)\eta(t') \rangle = \nu_{\text{p}}\delta(t-t') + \nu_{\text{p}}^2$. In the diffusion limit, i.e., in the limit of small coupling strengths of the Poisson noise, the dynamics of the free membrane potential can be described by an Ornstein-Uhlenbeck process (*Uhlenbeck and Ornstein*, 1930), basically performing a random walk around a fixed mean value with a Gaussian free membrane potential distribution (see Fig. 3.3B). Similarly to biological neurons (*Carandini*, 2004), this random movement of the membrane potential can kick the membrane potential above threshold even when the mean value is below threshold (or kick it below threshold when the mean value is above), leading to stochastic firing. In addition, the strong synaptic

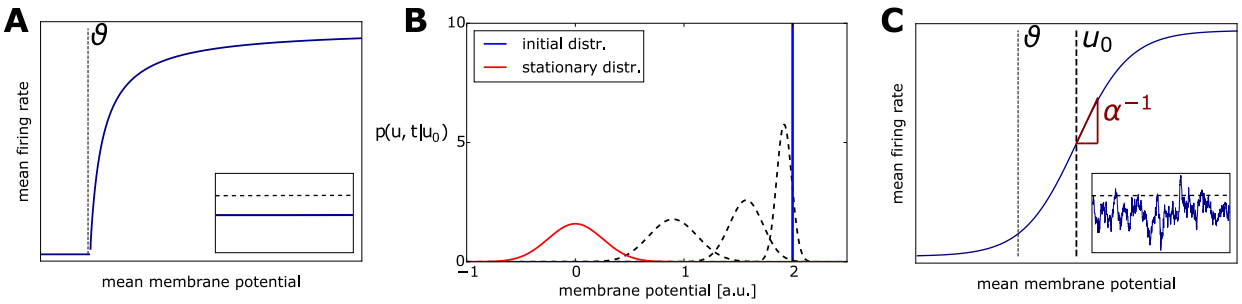

Figure 3.3.: **(A)** Activation function of a LIF neuron with CoBa synapses and no external noise. Below threshold ϑ, the neuron is silent and only starts eliciting spikes after crossing the threshold. Inlet: trace of an LIF neuron's free membrane potential (no spiking) with a constant offset current as input (blue). Only by shifting the membrane potential upwards by increasing the offset current can the threshold (black, dashed) be reached. **(B)** The free membrane potential dynamics of a CoBa LIF neuron can be described as an Ornstein-Uhlenbeck process. In the beginning, the membrane potential follows a Dirac delta distribution (blue). As time evolves, the distribution widens (diffusion) and the mean value shifts (drift) until the stationary distribution (red) is reached – here: a normal distribution with mean 0 and standard deviation 0.5. Figure taken from *Dold* (2016). **(C)** Activation function of a CoBa LIF neuron under high-frequency excitatory and inhibitory Poisson noise. Even if the mean membrane potential is below threshold, the neuron can have a non-zero firing rate. The activation function can approximately be described by a logistic function $\varphi(\mu) = (1 + \exp(-(\mu - u_0)/\alpha))^{-1}$. Inlet: trace of an LIF neuron's free membrane potential driven by Poisson noise.

background bombardment leads to a reduced effective membrane time constant, allowing the neuron to attain a high reaction speed to synaptic inputs.

In this state, commonly known as high-conductance state (HCS) (*Destexhe, Rudolph, and Paré*, 2003; *Kumar, Schrader, Aertsen, and Rotter*, 2008), the neuron's activation function becomes approximately logistic

$$\varphi(\mu) = \left(1 + \exp\left(-(\mu - u_0)/\alpha\right)\right)^{-1}, \tag{3.3}$$

where α is the inverse slope and u_0 the inflection point of the activation function (Fig. 3.3C). μ is the mean free membrane potential of the neuron in the HCS (see Eq. A.2b). With this result, we can construct a connection between sampling from Boltzmann distributions with binary random variables and the dynamics of LIF networks.

3.2.2 Spikes as samples from a probability distribution

In deep learning, it was found that a particularly useful distribution for inference and learning are Boltzmann distributions of the following form

$$p(\boldsymbol{z}) \propto \exp\left(\frac{1}{2}\boldsymbol{z}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{z} + \boldsymbol{z}^{\mathrm{T}}\boldsymbol{b}\right), \tag{3.4}$$

with binary state space $\boldsymbol{z} \in \{0, 1\}^N$ over N random variables. Each random variable can be depicted as a node in a graph, with $\boldsymbol{W}$ being a symmetric and zero-diagonal matrix representing the interaction strengths between nodes and $\boldsymbol{b}$ a vector representing each node's

bias. The bias determines which state each random variable prefers given the nodes are unconnected, i.e., $\boldsymbol{W}$ vanishes. Models that shape such distributions by learning both $\boldsymbol{W}$ and $\boldsymbol{b}$ to approximate the underlying distribution of a data set are called Boltzmann machines (*Ackley, Hinton, and Sejnowski*, 1985) and can be used to perform powerful Bayesian inference tasks like pattern classification, pattern completion, dimensionality reduction and generation of new data samples (*Hinton and Salakhutdinov*, 2006; *Hinton, Osindero, and Teh*, 2006; *Salakhutdinov and Hinton*, 2009; *Buesing, Bill, Nessler, and Maass*, 2011; *Fisher, Smith, and Walsh*, 2018). Recently, they have also been proposed as suitable models for solving the quantum many-body problem (*Carleo and Troyer*, 2017).

To sample from such a Boltzmann distribution, we split the sampling process in several steps and sample each variable sequentially while updating the state vector accordingly. For instance, for $N = 3$, to draw the $i+1$'th sample we first sample z_0^{i+1} conditioned on (z_1^i, z_2^i), then z_1^{i+1} conditioned on (z_0^{i+1}, z_2^i) and finally z_2^{i+1} conditioned on (z_0^{i+1}, z_1^{i+1}). Since the conditional probabilities are proportional to the joint probability distribution

$$p(z_k|\boldsymbol{z}_{\setminus k}) = \frac{p(\boldsymbol{z})}{p(\boldsymbol{z}_{\setminus k})} \propto p(\boldsymbol{z})\,, \tag{3.5}$$

sampling from the conditional distribution in each step conserves the stationary distribution $p(\boldsymbol{z})$ of the whole update sequence. For the individual sampling steps, we use the Metropolis-Hastings algorithm (Section 3.1.2) where the proposal distribution is chosen to be the conditional distribution of the Boltzmann distribution $p(z_k|\boldsymbol{z}_{\setminus k})$. This way, the acceptance ratio of the Metropolis-Hastings step becomes

$$\alpha(z_k^{i+1}, z_k^i) = \min\left(1\,, \frac{p(z_k^{i+1}|z_0^{i+1}...z_{k-1}^{i+1}z_{k+1}^i...z_{N-1}^i)p(z_k^i|z_0^{i+1}...z_{k-1}^{i+1}z_{k+1}^i...z_{N-1}^i)}{p(z_k^i|z_0^{i+1}...z_{k-1}^{i+1}z_{k+1}^i...z_{N-1}^i)p(z_k^{i+1}|z_0^{i+1}...z_{k-1}^{i+1}z_{k+1}^i...z_{N-1}^i)}\right) = 1\,, \tag{3.6}$$

and sampling is reduced to performing Monte Carlo steps on the conditional probabilities alone. This procedure is also known as Gibbs sampling (*Geman and Geman*, 1984). In case of a Boltzmann distribution as defined in Eq. 3.4, the conditional probability is given by a logistic function

$$p(z_k = 1|\boldsymbol{z}_{\setminus k}) = \frac{1}{1 + e^{-\sum_i W_{ki} z_i - b_k}}\,. \tag{3.7}$$

The activation function of an LIF neuron in the HCS can also be interpreted as a conditional probability when adapting the following coding scheme (Fig. 3.4A,B): a neuron is in the binary state $z_k = 1$ while being refractory, i.e., after eliciting a spike, when it affects its postsynaptic partners. Otherwise, if the neuron is not refractory, it is considered to be in the $z_k = 0$ state where it does not interact with its partners. With this coding scheme, the neuron's activation function (Eq. 3.3) corresponds to the probability of finding the neuron in the binary state $z_k = 1$ given the states of all remaining neurons $\boldsymbol{z}_{\setminus k}$

$$p(z_k = 1|\boldsymbol{z}_{\setminus k})_{\mathrm{LIF}} = \frac{1}{1 + \exp\left(-(\mu(\boldsymbol{z}_{\setminus k}) - u_0)/\alpha\right)}\,. \tag{3.8}$$

If we compare the LIF neuron's activation function with the transition operator of Gibbs sampling (Eq. 3.7), we find that both take approximately the same form, i.e., the transition

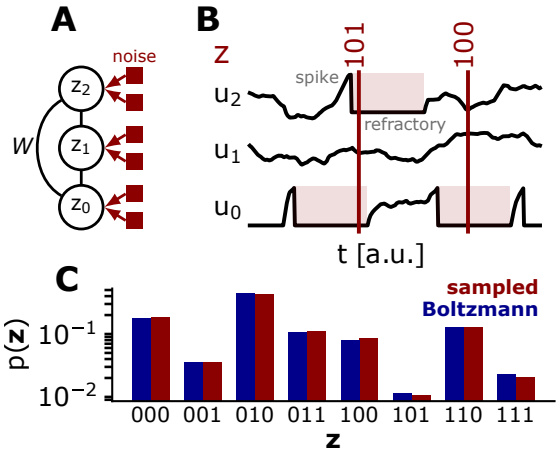

Figure 3.4.: **(A)** Schematic of a sampling spiking network, where each neuron (circles) encodes a binary random variable $z_i \in \{0, 1\}$. Neurons are rendered stochastic by adding external Poisson sources of high-frequency balanced noise (red boxes). **(B)** A neuron represents the state $z_k = 1$ when refractory and $z_k = 0$ otherwise. **(C)** The dynamics of neurons in stochastic spiking network can be described as sampling (red bars) from a target Boltzmann distribution (blue bars). Figures and caption adjusted from *Dold et al.* (2019a).

operator is a logistic function, while the LIF neuron's activation function can be fitted by one. Hence, we expect both algorithms to sample from similar distributions as long as the conditional probabilities match. By mapping Boltzmann parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ to synaptic weights $\boldsymbol{w}$ and leak potentials $\boldsymbol{E}_\mathrm{l}$ (or external currents) such that the neuron's activation function (Eq. 3.3) and the transition operator of Gibbs sampling (Eq. 3.7) agree (*Petrovici, Bill, Bytschok, Schemmel, and Meier*, 2016)

$$w_{kj} = \frac{\alpha W_{kj} C_\mathrm{m} \frac{\tau_\mathrm{ref}}{\tau^\mathrm{syn}} \left(1 - \frac{\tau^\mathrm{syn}}{\tau_\mathrm{eff}}\right) \left(E^\mathrm{rev}_{kj} - \mu\right)^{-1}}{\left[\tau^\mathrm{syn}\left(e^{-\frac{\tau_\mathrm{ref}}{\tau^\mathrm{syn}}} - 1\right) - \tau_\mathrm{eff}\left(e^{-\frac{\tau_\mathrm{ref}}{\tau_\mathrm{eff}}} - 1\right)\right]}\,, \tag{3.9}$$

$$(E_\mathrm{l})_k = \frac{\tau_\mathrm{m}}{\tau_\mathrm{eff}}(\alpha b_k + u_0) - \sum_{x \in \{\mathrm{e,i}\}} \frac{\langle g^\mathrm{syn}_x \rangle}{g_\mathrm{l}} E^\mathrm{rev}_x\,, \tag{3.10}$$

the spike dynamics of a network of LIF neurons therefore approximately[4] represents sampling from a binary Boltzmann distribution (Eq. 3.4), see Fig. 3.4C. Here, w_{kj} is the synaptic weight from neuron j to neuron k, $\boldsymbol{E}_\mathrm{l}$ a vector containing the leak potentials of all neurons, $\boldsymbol{b}$ the corresponding bias vector, $E^\mathrm{rev}_{kj} \in \{E^\mathrm{rev}_\mathrm{e}, E^\mathrm{rev}_\mathrm{i}\}$, depending on the nature of the respective synapse, and $\tau_\mathrm{m} = \frac{C_\mathrm{m}}{g_\mathrm{l}}$ (see Appendix A.1.7 for a derivation). If every neuron receives different background noise, the fit parameters α and u_0 as well as the effective time constant τ_eff and mean synaptic conductances $\langle g^\mathrm{syn}_x \rangle$ are neuron specific (not shown here to ease notation). To summarize, due to the almost logistic shape of the activation function of LIF neurons in the HCS, spikes can be interpreted as samples from a Boltzmann-type probability distribution with binary random variables, characterized by weight and bias parameters.

For a mathematically more rigorous derivation and analysis of sampling in spiking neural networks, see *Buesing, Bill, Nessler, and Maass* (2011); *Petrovici, Bill, Bytschok, Schemmel, and Meier* (2016); *Gürtler* (2018).

[4]For several reasons the sampling is only approximate. First, the activation function is only approximately logistic. Second, the interaction between neurons happens via postsynaptic potentials (PSP), which decay exponentially and thus last longer than a single 1-state. Third, LIF neurons tend to produce bursts, i.e., if the free membrane potential is above threshold after a spike, the probability to elicit a spike directly after the refractory period is increased compared to Gibbs sampling (*Gürtler*, 2018).

3.2.3 Training spike-based sampling networks

In general, Boltzmann machines can be trained via maximum likelihood learning to reproduce a given target distribution or to learn from samples originating from an unknown data distribution. The resulting learning algorithm increases the probability to produce desired states and reduces the probability of producing non-desired states. Mathematically, this takes the following form:

$$\Delta W_{ij} \propto \langle \nabla_{W_{ij}} \ln p(\boldsymbol{z}) \rangle_{\text{data}} \,, \tag{3.11}$$

$$\Delta b_i \propto \langle \nabla_{b_i} \ln p(\boldsymbol{z}) \rangle_{\text{data}} \,, \tag{3.12}$$

where we average over samples from the data distribution we want to model. Since the logarithm is monotonically increasing, maximizing the log probability is equivalent to maximizing the probability itself. For a fully visible[5] Boltzmann distribution with unconstrained connectivity matrix, this becomes

$$\Delta W_{ij} = \langle z_i z_j \rangle_{\text{data}} - \langle z_i z_j \rangle_{\text{free}} \,, \tag{3.13}$$

$$\Delta b_i = \langle z_i \rangle_{\text{data}} - \langle z_i \rangle_{\text{free}} \,, \tag{3.14}$$

using

$$\nabla_{W_{ij}} \ln p(\boldsymbol{z}) = \nabla_{W_{ij}} \left(\frac{1}{2} \boldsymbol{z}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{z} + \boldsymbol{z}^{\mathrm{T}} \boldsymbol{b} \right) - \nabla_{W_{ij}} \ln \sum_{\boldsymbol{z}} \exp \left(\frac{1}{2} \boldsymbol{z}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{z} + \boldsymbol{z}^{\mathrm{T}} \boldsymbol{b} \right) \,, \tag{3.15}$$

$$= z_i z_j - \frac{\sum_{\boldsymbol{z}} z_i z_j \exp\left(\frac{1}{2} \boldsymbol{z}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{z} + \boldsymbol{z}^{\mathrm{T}} \boldsymbol{b} \right)}{\sum_{\boldsymbol{z}} \exp\left(\frac{1}{2} \boldsymbol{z}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{z} + \boldsymbol{z}^{\mathrm{T}} \boldsymbol{b} \right)} \,, \tag{3.16}$$

$$= z_i z_j - \sum_{\boldsymbol{z}} z_i z_j \, p(\boldsymbol{z}) \,, \tag{3.17}$$

$$= z_i z_j - \langle z_i z_j \rangle_{\text{free}} \,, \tag{3.18}$$

where the second term originates from the distribution's partition function. The learning rule is separated into two phases: (i) a "free" phase, where the network receives no input and its network activity is self-generated, i.e., it freely generates samples (also called "dreaming phase"), and (ii) a "data" phase, where data samples are imposed on the network. Such data samples are supposed to be generated from an underlying data distribution we want to model, e.g., a distribution that generates samples resembling images of cats and dogs. The term obtained from the first phase reduces the probability of self-generated samples, and the term from the second phase increases the probability of generating samples that resemble the training data. If the network generates samples resembling the training data, learning stops as the contributions of both phases cancel each other. In general, the learning rule models the data distribution by matching both mean values as well as relevant pairwise correlations in data samples. This type of learning rule is generally known as "wake-sleep learning" (*Ackley, Hinton, and Sejnowski*, 1985; *Hinton*, 2002), alluding to the "dreaming" and "wake" (or

[5] I.e., no hidden neurons.

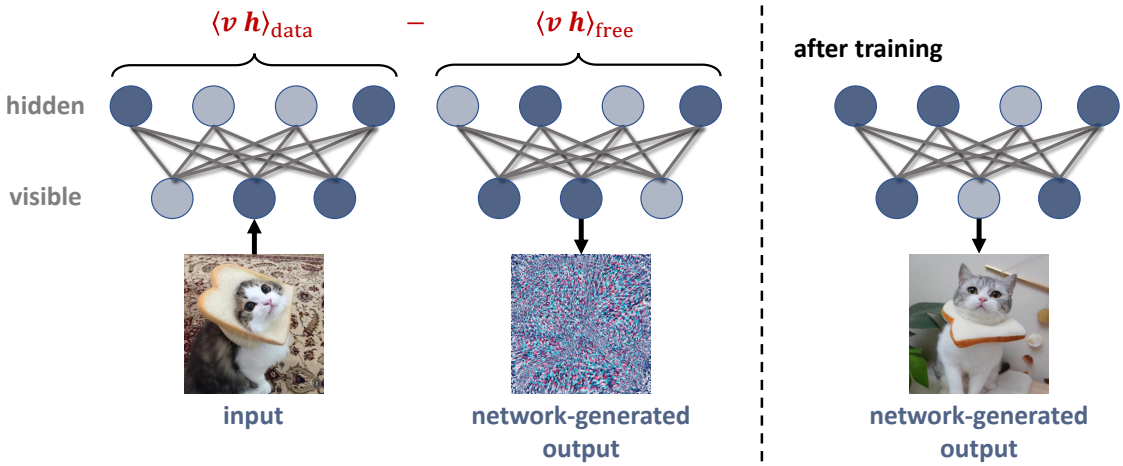

Figure 3.5.: Schematic illustration of the two learning phases for a restricted Boltzmann machine. Input is only presented to the network during the data phase. After training, the visible activity generated by the network resembles the learned data. The model can be used for classification by adding a label area on top. For the special case of only one hidden area, these label neurons are actually part of the visible area because of the symmetric connections (not shown here). Color intensity corresponds to neuronal activity. Photographs taken from (left to right) favim.com, 123rf.com and de.newchic.com, 18.02.2020.

data-driven) phase. For the special case that expectation values are approximated using single samples, the learning rule is called "contrastive divergence". If the underlying data distribution is known, this can also be rewritten in terms of probabilities:

$$\Delta W_{ij} = p^{\text{data}}(z_i = 1, z_j = 1) - \langle z_i z_j \rangle_{\text{free}} \ , \tag{3.19}$$

$$\Delta b_i = p^{\text{data}}(z_i = 1) - \langle z_i \rangle_{\text{free}} \ , \tag{3.20}$$

For representing high-dimensional data, it is useful to pre-impose a hierarchical structure in the connectivity matrix by separating the neurons into visible $\boldsymbol{v}$ and hidden $\boldsymbol{h}$ units, without connections between neurons of the same type (*Hinton and Salakhutdinov*, 2006). In this case, the Boltzmann distribution looks like

$$p(\boldsymbol{z}) \propto \exp\left(\boldsymbol{v}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{h} + \boldsymbol{v}^{\mathrm{T}} \boldsymbol{b} + \boldsymbol{h}^{\mathrm{T}} \boldsymbol{a}\right) \ , \tag{3.21}$$

with biases $\boldsymbol{a}$ for hidden neurons. This is generally known as a restricted Boltzmann machine (*Hinton*, 2002), where the hidden neurons take on the role of modeling the statistical dependencies between visible neurons. Without hidden neurons, these dependencies can only be captured using the visible-to-visible connections, restricting the shape of our model distribution to the form of a Boltzmann distribution (Eq. 3.4). But by adding hidden neurons, the posterior distribution of the visible states is obtained by marginalising over the hidden neurons, which is not limited to the form of Eq. 3.4 and can be used to model arbitrary probability distributions given a sufficient number of hidden neurons. For maximum

likelihood learning, we then obtain (see Fig. 3.5 for an illustration)

$$\Delta W_{ij} = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{free}} \,, \tag{3.22}$$

$$\Delta b_i = \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{free}} \,, \tag{3.23}$$

$$\Delta a_i = \langle h_i \rangle_{\text{data}} - \langle h_i \rangle_{\text{free}} \,. \tag{3.24}$$

With this, given a data set, we can learn the parameters of a Boltzmann distribution approximating the underlying distribution the data was generated from[6]. This gives us two possibilities to set up LIF networks to perform Bayesian inference on a given data set:

1. First train a Boltzmann machine on the data and use Eqs. 3.9 and 3.10 to translate the parameters to a network of LIF neurons.

2. Train the LIF network directly by translating the wake-sleep learning rule updates with Eqs. 3.9 and 3.10 to updates of synaptic and neuronal parameters.

3.3 Spiking networks for spatio-temporal predictions

Spike-based sampling networks, as described in the previous section, have so far been used to demonstrate both discriminative and generative properties on static input patterns, like image classification and pattern completion (*Petrovici, Bill, Bytschok, Schemmel, and Meier*, 2016; *Leng, Martel, Breitwieser, Bytschok, Senn, Schemmel, Meier, and Petrovici*, 2018). However, the introduced model can also be used on spatio-temporal inference problems, as for instance the prediction of a particle trajectory that is slowly revealed to the network.

3.3.1 From static patterns to time-dependent input

When training sampling networks on high-dimensional data, we typically clamp the visible area to "static" data values for each update step, i.e, the data samples have no temporal component and remain unchanged while being presented to the network. This remains true for testing the network's training progress, where the network is challenged to classify or fill in missing pixels from static data samples.

To move from static to time-continuous inputs, we can simply rephrase the task a network has to solve: instead of seeing the whole or partial version of the input, the network is exposed to a time-continuous input $\boldsymbol{x}(t)$ that is slowly revealed. Hence, at time t_0, the sampling network receives no input at all, and its best guess or prediction is based on the prior $p(\boldsymbol{x})$ encoded by spontaneous activity in the visible area. However, for $t' > t_0$, more and more of the input is revealed, constraining $p\big(\boldsymbol{x}(t > t') \,|\, \boldsymbol{x}(t \leq t')\big)$ in the Bayesian sense as explained in Section 3.1.2. Hence, during presentation, the internal model learned by the network $p(\boldsymbol{x})$ is consistently updated by new observations, leading to updated predictions. For instance, the process of drawing a letter by hand can be rephrased in this way, and the sampling

[6] Even though it is not guaranteed that such a distribution exists, it is a reasonable assumption as we are interested in capturing characteristic structures hidden in the data. If such structures do not exist, we are not able to reasonably model the data to begin with.

network has to guess (i) the complete shape of the letter, based on current observations and (ii) the letter being drawn (*Roth*, 2014). Another example is the prediction of trajectories, where a particle moves on a two-dimensional plane from left to right, and the network has to predict the currently unobserved parts of the trajectory including its end-position.

Of course, this approach harbors a major drawback: Even though we can simply train the network on trajectories $\boldsymbol{x}(t)$ by showing it the whole input $\lim_{t\to\infty} \boldsymbol{x}(t)$, all dimensions (spatial and temporal) have to be represented by individual neurons, and hence for longer time series, the amount of neurons required to solve the task grows linearly. This is not the case for recurrent neural networks like Elman networks (*Elman*, 1990) with context areas (*Mikolov, Joulin, Chopra, Mathieu, and Ranzato*, 2014) or long short-term memory networks (LSTM, *Hochreiter and Schmidhuber* 1997) that explicitly use their recurrent architecture to represent time and are commonly used for sequential tasks like language and text processing.

However, encoding both spatial and temporal dimensions with visible neurons enables a clear generalization of spike-based Bayesian inference to time-continuous inputs. Furthermore, it allows an extension of the spike-based sampling framework to scenarios where the network takes control of an agent that acts in an environment. In this case, the input to the network $\boldsymbol{x}(t)$ represents the state of the environment that can either change due to external factors or due to actions of the agent. For instance, the environment could be the game Pong, and the agent is the network controlling a paddle at the right side of the play field. The state of the environment is the currently uncovered ball trajectory $x(t)$, and the agent's goal is to hit the ball with the paddle, which changes the ball's trajectory (Fig. 3.6A). In the following sections, we will study the feasibility of time-continuous Bayesian inference with spiking neural networks.

3.3.2 Pattern completion of line segments

To investigate how well spike-based sampling networks perform when faced with time-continuous problems, we test them on the task of (i) predicting the remaining parts of a gradually uncovered trajectory and (ii) using this prediction to control an agent that can move a paddle up or down to position it at the predicted end-position of the trajectory. For instance, this might be the trajectory of a particle (or ball) moving from the left to the right side of a two-dimensional plane, with a random initial starting angle and reflecting boundaries on the top and bottom of the plane. Moving a paddle to the predicted end-position of the trajectory is then akin to catching a ball in the game of Pong (Fig. 3.6A). To create a data set for training, such trajectories are calculated using the equation of motion for a particle $\ddot{\boldsymbol{x}} = -\nabla V(\boldsymbol{x})$ in a potential $V(\boldsymbol{x})$ that depends on the particle's position $\boldsymbol{x}$. To simulate a trajectory, a particle is initialized at position $\boldsymbol{x}_0 = (-\frac{w}{2}, y_0)$ with velocity $\boldsymbol{v}_0 = (\cos\alpha, \sin\alpha)$, where w is the width of the plane, $(0, 0)$ the center of the plane, $\alpha \in (-\pi, \pi)$ the starting angle and $y_0 \in [-\frac{h}{2}, \frac{h}{2}]$ the starting height (with the plane's height being h). Furthermore, the starting angle is restricted such that in the case of flat potentials $V(\boldsymbol{x}) = \text{const.}$, at most

one boundary reflection occurs, leading to maximum and minimum angles (*Zenk*, 2018)

$$\tan \alpha_{\max}(y_0) = \frac{\frac{3}{2}h - y_0}{w}\,, \tag{3.25}$$

$$\tan \alpha_{\min}(y_0) = -\frac{\frac{3}{2}h + y_0}{w}\,. \tag{3.26}$$

After simulating a trajectory, it is down-sampled on a discretized grid

$$c_{ij} = \frac{\Delta}{2} + \begin{pmatrix} \Delta \cdot (j-1) - \frac{w}{2} \\ \Delta \cdot (i-1) - \frac{h}{2} \end{pmatrix}\,, \qquad i,j \in \mathbb{N}\,, \tag{3.27}$$

with Δ being the grid-spacing. The trajectory is widened by assigning a value to each grid entry depending on its distance to the trajectory, i.e.,

$$d_{ij} = \min_t \|c_{ij} - \boldsymbol{x}(t)\|\,, \tag{3.28}$$

$$P_{ij} = \max\left(0, \left(1 - \frac{d_{ij}}{\delta}\right)^{\lambda}\right)\,, \tag{3.29}$$

where P_{ij} is the trajectory strength at position c_{ij}, and the values decay with increasing distance to the trajectory according to a power law with exponent λ and cutoff δ (Fig. 3.6B). The motivation for this step is twofold: (i) increase the input to the neural network and (ii) smear out the ball position, i.e., add noise to it. By varying the starting position $\boldsymbol{x}_0$ and angle α, we generated a data set for a flat potential, consisting of $4 \cdot 10^4$ images split into training, validation and test set in the ratio 2:1:1, with $w = 48$, $h = 40$, $\delta = \frac{5}{2}$ and $\lambda = \frac{1}{2}$ (see Fig. 3.6C for example trajectories).

To reduce simulation time, we first trained a restricted Boltzmann machine with $40 \cdot 48$ visible, 500 hidden and 10 label units on the data set before translating the learned parameters

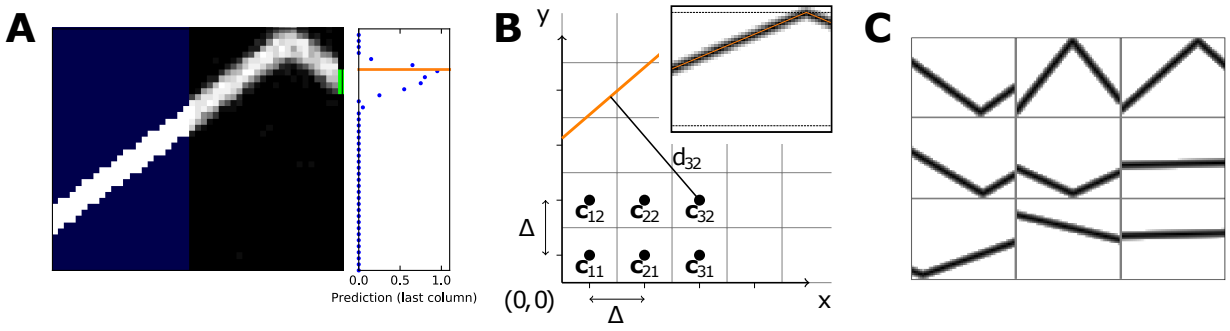

Figure 3.6.: **(A)** Illustration of the experimental setup. The input to the network is the trajectory of a ball that flies from left to right (blue background). Remaining parts of the trajectory that have not been revealed yet have to be inferred by the network (black background). In addition, a paddle at the right-most end of the play field is controlled by the network to catch the ball, i.e., predict the end point of the trajectory. **(B)** Trajectories are first created by solving the equation of motion of a particle moving through a potential (here: flat), with according boundary conditions. The obtained trajectory is down-sampled to the resolution of the playing field and widened. **(C)** Example trajectories for the case of a flat potential. Figures adjusted from *Zenk* (2018).

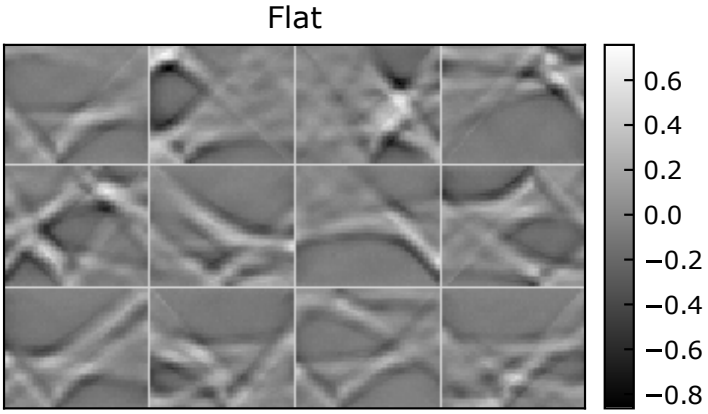

Figure 3.7.: Exemplary receptive fields of hidden neurons after training. Individual neurons encode features like flocks of trajectories, reflection points and crossing points. Figure adjusted from *Zenk* (2018).

to neurosynaptic parameters in an LIF network, using Eqs. 3.9 and 3.10. The label units are used to steer a 4 pixel wide paddle vertically[7], as sketched in Fig. 3.6A. Instead of training with pure contrastive divergence, we used persistent contrastive divergence (*Tieleman*, 2008). In persistent contrastive divergence, the data terms in Eqs. 3.22 to 3.24 are approximated by performing one Gibbs sampling step conditioned on the data as visible input ($p(\boldsymbol{h}|\boldsymbol{v}_{\text{data}})$) and the free terms are approximated through a MCMC that is consistently run during training, i.e., without resetting the chain after parameter updates. This has the benefit that, as long as the network parameters change slowly, the consistent chain produces valid samples of the Boltzmann distribution and it is not necessary to run individual MCMCs in every update step to approximate the negative term in the learning rule. Furthermore, we used L2 regularization[8] (*Ng*, 2004) and momentum[9] (*Rumelhart, Hinton, and Williams*, 1986) to improve the training quality, such that the weight update of the k'th training step takes the form

$$\Delta W_{ij}^{k} = \epsilon \Delta W_{ij}^{k-1} + \eta(k)\Big(\langle \nabla_{W_{ij}} \ln p(\boldsymbol{z}) \rangle_{\text{data}} - \kappa W_{ij} \Big) , \tag{3.30}$$

where ϵ determines how quickly previous values of the gradient decay (momentum), κ regulates how strongly weights are pushed to small values (L2 regularization) and $\eta(k) = \frac{\eta_0}{1+\frac{k}{2000}}$ is a decaying learning rate with initial value η_0. Biases follow a similar update scheme, but without regularization, as recommended in *Hinton* (2012). In addition, a batch size of 50 was used during training, meaning that we presented 50 patterns to the network and averaged the weight updates. The used parameters for training are listed in Appendix A.3.1. To get a feeling about what the hidden neurons learned, we can look at the weights projecting

[7] The paddle moves, with finite velocity, to a vertical target position obtained by averaging over all possible vertical positions weighted by the network's conditional probabilities.

[8] Also known as weight decay, since the regularization term pulls the weights κW_{ij} to zero.

[9] I.e., the weight update contains a memory of previous gradients, $\epsilon \Delta W_{ij}^{k-1}$. Momentum helps to overcome saddle points of the optimization landscape faster, as weight updates do not vanish when crossing a saddle point (where the gradient vanishes).

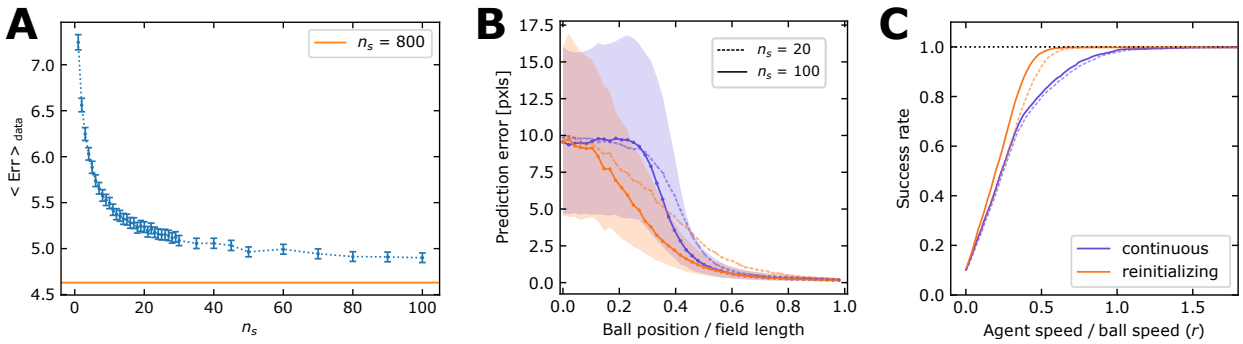

Figure 3.8.: **(A)** Time-averaged prediction error of the test data set as a function of the number of sampling steps the network is allowed to do to reach a decision. After approximately $n_s = 20$, the performance only slightly improves further. **(B)** Prediction error as a function of the uncovered trajectory fraction, from no input (0) to a fully revealed trajectory (1). If the network is reinitialized after each time step, the predictions improve much faster. Colors as in (C). **(C)** Success rate of the experiment in (B). Again, with reinitialization the success rate increases much faster to its maximum value. The reason for this is that, in case of Gibbs sampling without reinitialization, the samples at different time steps are correlated and depend on previous network states. Thus, if the network jumps into a wrong mode in the beginning (or is strongly biased towards a certain mode), it is harder for the network to escape this mode initially. Line styles as in (B). Figures adjusted from *Zenk* (2018).

from the visible neurons to individual hidden neurons – also called "receptive fields". As shown in Fig. 3.7, hidden neurons mostly react to bundles of similar trajectories, reflection points and points where many trajectories cross.

We can derive four measures of performance from the task of predicting the point of impact on the right-hand side of the plane:

- the prediction error $e(t) = \|x_w - z_w(t)\|$, where x_w is the end-point of the trajectory and $z_w(t)$ the network estimate at time t. Alternatively, we could also use the sum of differences over the whole trajectory.
- the time-averaged prediction error $E = \frac{1}{T}\int_0^T e(t)\,\mathrm{d}t \approx \frac{1}{w}\sum_{k=0}^{w} e(k)$.
- the success rate $S(r)$ that measures how often the network is able to "catch" the ball with the paddle, depending on the ratio r of vertical paddle speed to horizontal ball speed.
- the success rate for infinite agent speed S_∞, i.e., where the paddle jumps instantaneously to the predicted end-point of the network in each time step.

We first evaluated the trained model with Gibbs sampling by looking at these measures. To evaluate, each image in the test set was clamped column by column in w steps, where after each step the model prediction was altered by performing n_s sampling steps. This way, in a single trial, the trajectory is slowly uncovered column by column, and between frame updates, the network has time to perform n_s sampling steps to update its prediction based on the new evidence, i.e., the newly uncovered column. Of course, by increasing the number of sampling steps, the performance of the network increases as the samples better represent

the posterior distribution (Fig. 3.8A,B), with the drawback that either the network has to produce samples faster or the trajectory has to be uncovered slower, leading to slower ball speeds. Moreover, we compared the performance of the network with a baseline, where the prediction of the remaining trajectory is simply the current ball position, and a non-continuous version of Gibbs sampling where after each clamping step (uncovering of new column), the visible area is first reinitialized before performing Gibbs updates (Fig. 3.8B,C).

As expected, for all investigated values of n_s, the Boltzmann machine performed better than the baseline predictor. However, especially when only a small fraction of the trajectory is clamped, the prediction error as well as the success rate decrease/increase much faster when the visible area is reinitialized during clamping steps. This is probably due to the model falling into a mode initially (i.e., one trajectory that is, for instance, heavily biased by the prior), where the clamped input is not strong enough yet to induce a mode switch. To circumvent this problem, tempering methods that enhance exploratory behavior can be used. One particularly natural method to improve exploration of the underlying probability landscape is available for spike-based sampling networks, as demonstrated in the next section.

3.3.3 Short-term plasticity improves exploration

Generative models getting stuck in attractor modes is a well-known problem called the "mixing problem" (*Bengio, Mesnil, Dauphin, and Rifai*, 2013). For instance, if we want our model to clearly separate different image classes, as in our case two strongly different trajectories, the best way to do so is to encode both trajectories as well-separated, high probability modes in the modeled probability distribution. This automatically leads to the problem that, by simply using random noise to explore the probability landscape, it is hard for a network to escape one of these strong attractor modes. In case of Gibbs sampling, the escape of strong modes can be facilitated by using tempering (*Salakhutdinov*, 2010), i.e., by globally "flattening" the probability landscape so the MCMC can escape from high probability regions more easily. For LIF networks, recently it has been observed that a local mechanism can achieve a similar effect (*Leng, Martel, Breitwieser, Bytschok, Senn, Schemmel, Meier, and Petrovici*, 2018), simply by equipping each synapse with a limited pool of neurotransmitters that is depleted when presynaptic spikes arrive and regenerates otherwise. In the following, we demonstrate that spike-based sampling networks equipped with such a short-term plasticity mechanism show improved performance in the spatio-temporal trajectory prediction task of the previous section.

Short-term plasticity describes the temporary strengthening and weakening of synaptic weights over short time scales. Contrary to long-term plasticity, which is used to memorize new information or learn new skills, short-term plasticity is mostly a consequence of the limited amount of resources available to a synapse to transmit information. An intuitive phenomenological model of short-term plasticity has been introduced by *Tsodyks and Markram* (1997), commonly known as the Tsodyks-Markram (TSO) model. Here, we use a slightly simplified version of TSO proposed in *Fuhrmann, Segev, Markram, and Tsodyks* (2002) and only model short-term depression, i.e., the weakening of synaptic strengths due to depletion of neurotransmitters available to the synapse. To do so, each synapse is equipped with a reservoir of neurotransmitter $R \in [0, 1]$, from which a fraction U_{SE} is used up whenever a

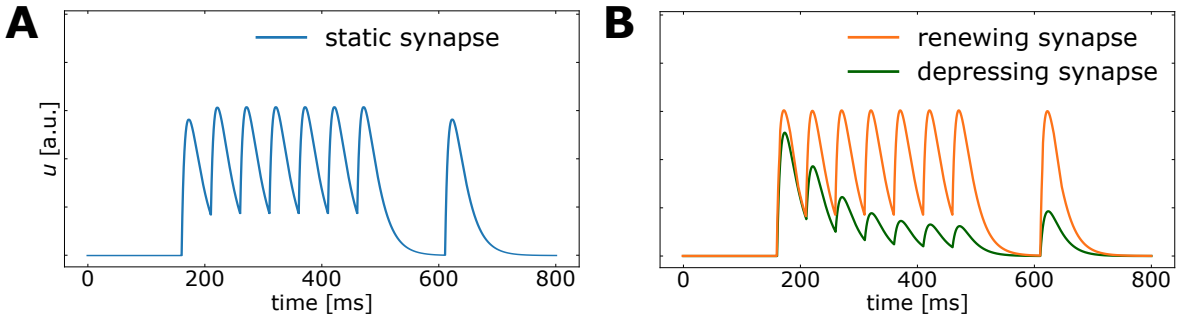

Figure 3.9.: Illustration of short-term plasticity. **(A)** Without short-term plasticity, also called "static synapse", the postsynaptic potential (PSP) envelope resulting from presynaptic spikes initially builds-up (due to overlap with exponential tails) and then keeps a constant height. **(B)** With short-term plasticity ($U = 0.5$), the depletion of neurotransmitters leads to a decrease in the postsynaptic response for strong presynaptic activity (green). The neurotransmitters recover on the time scale of the recovery time constant τ_{rec} if no presynaptic activity is present, resulting in a recovered postsynaptic response height. For the special case of $U = 1$ and $\tau_{\text{rec}} = \tau^{\text{syn}}$, called renewing synapses, all PSPs have identical height (orange). Figures adjusted from *Zenk* (2018).

presynaptic spike occurs,

$$\dot{R}(t) = \frac{1 - R}{\tau_{\text{rec}}} - \sum_{\text{spikes s}} U_{\text{SE}}\, R\, \delta(t - t_s)\,, \tag{3.31}$$

modifying the synaptic weight w to $w(t) = U_{\text{SE}} \cdot R(t) \cdot w$. Thus, if several presynaptic spikes occur in sequence, the effect of each spike on the PSP will be increasingly diminished. While no presynaptic spikes occur, the reservoir recovers on the time scale τ_{rec} (see Fig. 3.9).

For spike-based sampling, we commonly choose $U_{\text{SE}} = 1$ and $\tau_{\text{rec}} = \tau^{\text{syn}}$, as proposed in *Petrovici, Bill, Bytschok, Schemmel, and Meier* (2016), which we call "renewing synapses" for the remainder of the document. Choosing the parameters this way compensates for an initial build-up of the PSP height if the neuron is exposed to tonic presynaptic spiking (Fig. 3.9A,B), ensuring that always only spike times transmit information. The build-up happens since the exponential tail of a PSP is not yet decayed to zero when the next PSP occurs[10]. In *Leng, Martel, Breitwieser, Bytschok, Senn, Schemmel, Meier, and Petrovici* (2018), it was further shown that certain choices of parameters different from the renewing case (Fig. 3.9B, green) can be used to destabilize local attractors in the network, without changing the probability landscape globally, and thus enforce an exploration of the probability landscape.

For these experiments, we use LIF neurons with CuBa synapses, as described in Section 2.1.2. To improve the mixing property of our networks, we performed a parameter scan in U_{SE} and τ_{rec} based on the time-averaged prediction error E. We found a valley of parameter combinations that are in a reasonable dynamical regime and lead to improved time-averaged

[10] If we consider two consecutive presynaptic spikes, the resulting PSP of the first spike will have decayed by a factor of $\exp\left(-\tau_{\text{ref}}/\tau^{\text{syn}}\right) > 0$ when the second one arrives. Thus, without renewing synapses, the height of the second PSP would be increased due to the non-zero remnant of the first one (Fig. 3.9A). But with renewing synapses, the first spike depleted the neurotransmitter pool R and the effect of the second spike is diminished by a factor $1 - \exp\left(-\tau_{\text{ref}}/\tau^{\text{syn}}\right)$, which is the fraction of regenerated neurotransmitters between consecutive spikes. Thus, adding the remaining PSP of the first spike to the diminished effect of

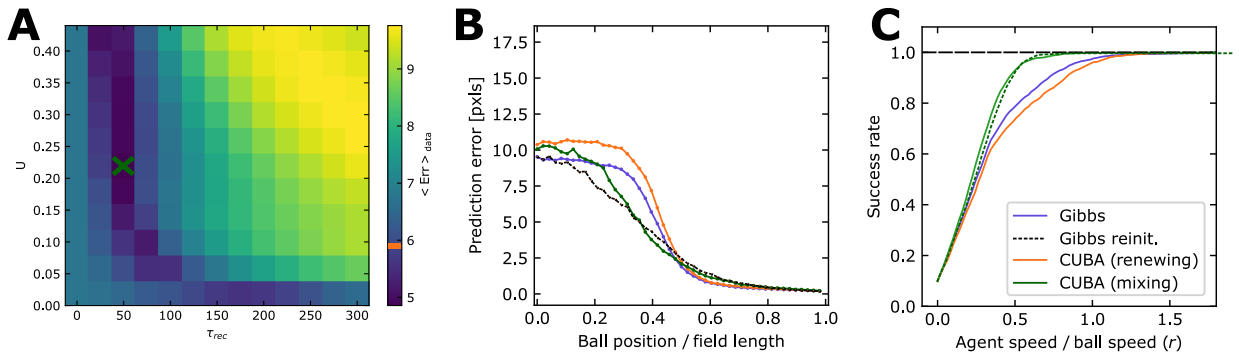

Figure 3.10.: Spatio-temporal prediction task with spike-based networks and short-term plasticity. **(A)** Time-average prediction error evaluated on the test set for different short-term plasticity parameters. We observe a valley of best parameters which clearly improve the performance compared to renewing synapses (orange bar). **(B)** Prediction error for CuBa LIF neurons with renewing synapses (orange), CuBa with the best short-term plasticity parameters of (A), $U = 0.22$ and $\tau_{\text{rec}} = 50\text{ms}$ (green cross, hereafter called "mixing synapses"), Gibbs sampling (violet) and Gibbs sampling with reinitialization (black, dotted). To increase visibility, only for the mixing case the first and third quartile are shown as well (green shade). **(C)** Respective success rates of the experiments shown in (B). CuBa LIF neurons with mixing parameters improve the generative properties of the network, allowing it to escape local attractors much faster than CuBa with renewing synapses or Gibbs sampling. This way, the biological inspired network reaches the performance of the Gibbs sampler with reinitialization between uncovered ball positions. Figures adjusted from *Zenk* (2018).

prediction errors compared to the obtained performance with renewing synapses (Fig. 3.10A). As expected, in these cases, the prediction error decreases much earlier than with renewing synapses (Fig. 3.10B), and the success rate converges faster to one (Fig. 3.10C). The best performance was reached for $U = 0.22$ and $\tau_{\text{rec}} = 50\text{ms}$, which we call "mixing synapses". By visually expecting single trials, one can see how, with a set of short-term depression parameters that facilitates mixing, the network explores several possible trajectories, see Fig. 3.11. In contrast, the network stays mostly in the same prediction when using renewing synapses (or, equivalently, Gibbs sampling is used to evaluate the model).

Since the task to predict linear trajectories is rather simple, we repeated the experiment for a Boltzmann machine trained on curved trajectories resulting from a Gaussian hill potential centered in the middle of the plane

$$V(x, y) = A \exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right) \right], \tag{3.32}$$

with standard deviations $\sigma_x = 0.32 \cdot w$, $\sigma_y = 0.25 \cdot h$ and amplitude $A = 0.4$. By using the same short-term depression parameters as for the flat potential, we again observe a strong performance increase compared to renewing synapses (Fig. 3.12). However, in this case, spike-based sampling with renewing synapses seems to perform much worse than Gibbs sampling. It is likely that this is due to the heuristic translation from Boltzmann parameters

the second spike yields $\exp\left(-\tau_{\text{ref}}/\tau^{\text{syn}}\right) + 1 - \exp\left(-\tau_{\text{ref}}/\tau^{\text{syn}}\right) = 1$, and both the first and second PSP have identical height (Fig. 3.9B, orange).

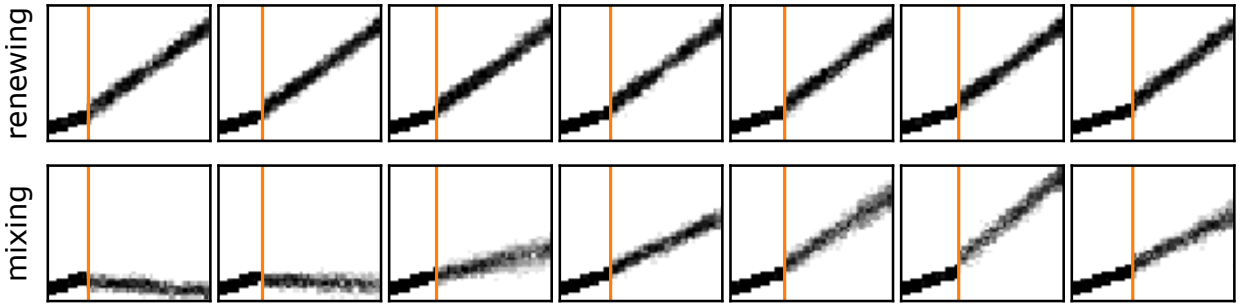

Figure 3.11.: Prediction of the network at different time steps, shown for CuBa LIF neurons with renewing (top) and mixing (bottom) synapses. With renewing synapses, the network stays stuck in its initial prediction. However, using mixing synapses the network explores different feasible solutions while the trajectory is uncovered. Figure adjusted from *Zenk* (2018).

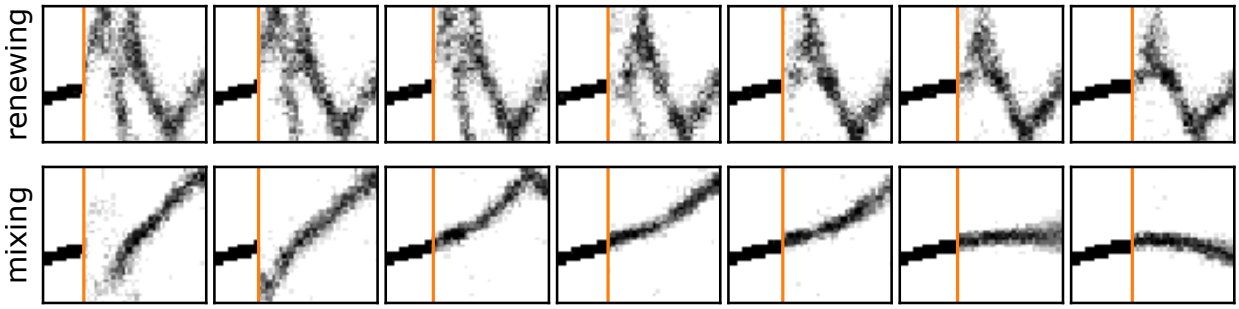

Figure 3.12.: Same as in Fig. 3.11, but for a ball moving through a Gaussian potential landscape. The network has been trained on the new data set, but we used the ideal TSO parameters found in Fig. 3.10 for flat potentials. Figure adjusted from *Zenk* (2018).

to neurosynaptic LIF parameters, and by training the LIF network directly, the difference in performance might be further reduced.

Finally, we found that network performance is hardly impaired when only the current ball position and a small part of the preceding trajectory is shown, i.e., when the network "forgets" parts of the trajectory that lie far enough in the past (or, similarly, it only "remembers" the recent past). In this case, the network has to both predict the future trajectory as well as reconstruct the past trajectory. This is especially helpful when the ball is under the influence of external forces, e.g., random kicks not present during training that alter the trajectory (see *Zenk* (2018), Section 4.3.4 and 4.3.5, for details). Furthermore, this increases the biological plausibility of the implementation, as a reset of the input after each trial is not required anymore.

In summary, these results show that spike-based sampling networks equipped with short-term depression are capable of reaching good performance in the time-continuous task of predicting a ball's trajectory. Especially the short-term depression reveals a natural way for neural networks to promote exploration of the input-constrained solution space without getting stuck in preliminary predictions. The proposed model might be used to implement an agent playing Pong on neuromorphic hardware; or by using the predicted trajectory as the state vector for an agent trained via reinforcement learning techniques (*Mnih, Kavukcuoglu,*

Silver, Graves, Antonoglou, Wierstra, and Riedmiller, 2013) to catch the ball as fast as possible by moving into the plane (e.g., playing tennis), making full usage of the trajectory prediction. This would be a natural extension of recent work on reinforcement learning (*Wunderlich et al.*, 2019) and spike-based sampling (*Kungl et al.*, 2019b) on neuromorphic hardware.

3.4 Sampling in deterministic ensembles

So far, sampling with spike-based neural networks required external sources of well-behaved noise. In the following, we will show that sampling also works under in-vivo-like conditions. This allows a self-consistent and self-contained realization of Bayesian computing in spiking neural networks, without any external noise sources.

3.4.1 The self-noising brain

A major ingredient of spike-based sampling is high-frequency Poisson noise that elevates neurons in a stochastic high-conductance state. However, this is quite problematic for physical realizations of spike-based sampling, as every neuron needs a perfect and private source of well-behaved and high-frequent noise – something that is obviously not the case in the brain and a strong constraint for realizations in physical devices. For instance, in case of neuromorphic hardware, this means that noise has to either be provided externally, which is restricted by bandwidth limitations (*Bytschok*, 2017), or has to be generated on-chip, for instance, via linear-feedback shift registers (*Murase*, 1992; *Großkinsky*, 2016), which are costly in terms of surface area and achievable noise quality. However, as discussed in Section 3.1.1, the brain mostly attains its stochastic behavior through synaptic noise. Spike-based sampling with synaptic transmission noise has been demonstrated in simulations in *Neftci, Pedroni, Joshi, Al-Shedivat, and Cauwenberghs* (2016), although a fundamental theoretical treatment is still lacking. Furthermore, these simulations still assume a well-behaved implementation of stochasticity to evaluate synaptic transmission probabilities, which they model by drawing pseudo-randomly generated numbers.

Arguably the biggest source of irregularity in the brain, as well as the actual motivation behind Poisson noise, is synaptic background noise. Thus, inspired by the response variability and the modular structure of the brain (*Chen, He, Rosa-Neto, Germann, and Evans*, 2008; *Bullmore and Sporns*, 2009; *Meunier, Lambiotte, and Bullmore*, 2010; *Bertolero, Yeo, and D'Esposito*, 2015; *Song, Sjöström, Reigl, Nelson, and Chklovskii*, 2005), i.e., strongly connected functional clusters that are weakly interconnected (see, e.g., Fig. 3.13A), we propose a similar architecture for spiking sampling networks: a weakly interconnected ensemble of functional subnetworks, where the interconnections are used to provide background noise to every subnetwork (Fig. 3.13B) – a neuronal heat bath so-to-speak. This enables the implementation of completely deterministic ensembles of functionally independent neural networks, where every network uses the spikes of other networks as a noise source to perform probabilistic inference (*Dold, Bytschok, Kungl, Baumbach, Breitwieser, Senn, Schemmel, Meier, and Petrovici*, 2019a). Thus, spikes take on a dual role: as samples from a posterior

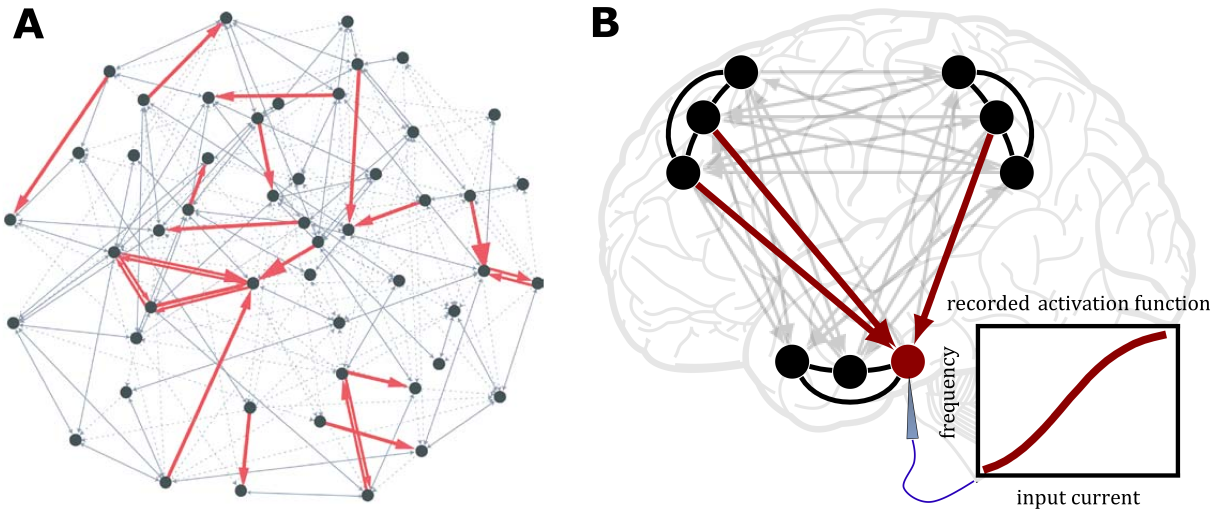

Figure 3.13.: **(A)** In the neocortex, strongly and often bidirectionally connected neurons are embedded in a sea of weak, unidirectional connections. The shown network was reconstructed from 50 layer 5 pyramidal neurons. Figure taken from *Song et al.* (2005). **(B)** Instead of using explicit Poisson noise to elevate neurons into a stochastic firing regime, inspired by the brain, we embed networks into a network of networks. Each network performs its own distinct functional task (black) while providing other neurons in the network with noise (gray). This is illustrated here for a single neuron: by receiving background noise from other sampling networks (red arrows), the red neuron becomes stochastic and expresses the logistic activation function needed for spike-based sampling. Image adjusted from https://commons.wikimedia.org/wiki/File:Brain_-Surface_Gyri.SVG (version: 11:12, 30 March 2010) and *Dold et al.* (2019a).

probability distribution, allowing each network to perform Bayesian inference tasks, but also as the source of background noise for other networks in the ensemble. This has the benefit that no resources are wasted on generating noise – different from previous approaches where networks are used solely to generate noise (*Jordan, Petrovici, Breitwieser, Schemmel, Meier, Diesmann, and Tetzlaff*, 2019a) or pseudo-random number generators are required to enable stochastic computations (*Petrovici, Bill, Bytschok, Schemmel, and Meier*, 2016; *Neftci, Pedroni, Joshi, Al-Shedivat, and Cauwenberghs*, 2016).

In the following sections, we approach the described ensemble model incrementally. First, we investigate the effect of auto- and cross-correlated background noise on the functional performance of spike-based sampling networks. This will enable us to set up ensembles of networks, without any external noise sources, as described above. Finally, we demonstrate that this approach can be scaled to networks trained on higher-dimensional visual data and is compatible with the constraints of physical model systems like BrainScaleS–1.

3.4.2 Characterizing ensemble-generated noise

Network-generated background activity, unlike Poisson noise, is both auto- and cross-correlated, impeding the performance of spike-based sampling networks receiving such noise. To investigate the effect of correlated noise on the performance of a sampling network, we replace Poisson noise with spikes coming from other (background) sampling networks –

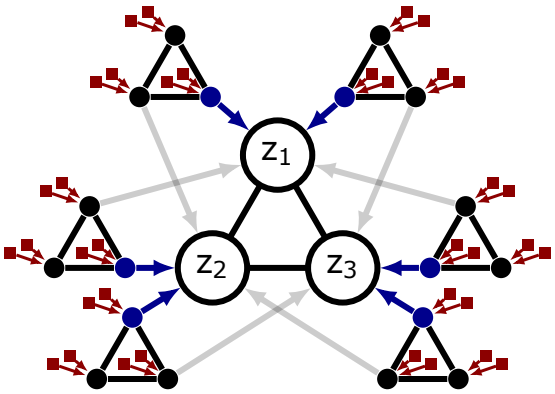

Figure 3.14.: Replacement of Poisson noise by spiking activity from other spike-based sampling networks. In this illustration, the principal network consists of three neurons receiving background input only from other functional networks that sample from their own predetermined target distribution. For clarity, only two out of a total of [260, 50, 34] (top to bottom in Fig. 3.15) background networks per neuron are shown here. By modifying the background connectivity (gray and blue arrows) the amount of cross-correlation in the background input can be controlled (Fig. 3.16). At this stage, the background networks still receive Poisson input (red boxes). Figure and caption adjusted from *Dold et al.* (2019a).

although at this stage, the background networks still receive Poisson input, see Fig. 3.14.

First, spike trains generated by LIF neurons feature an autocorrelation function (ACF) that depends both on the refractory time τ_{ref} and mean spike frequency $\bar{r} = p(z = 1)\tau_{\text{ref}}{}^{-1}$. For low firing rates, a neuron spikes only rarely and generated spikes are far enough apart to be completely uncorrelated, yielding a Dirac delta distribution as the ACF. If we pool the spike trains of many low-frequency neurons together, the generated spike train approaches a Poisson distribution: in every time step, many neurons can spike (independently), but only with a very low probability. However, for high firing rates, regular structures, so-called bursts, i.e., sequences of equidistant spikes with an interspike interval (ISI) of ISI $\approx \tau_{\text{ref}}$, start dominating the spike trains[11]. Since the ACF measures the amount of self-similarity of the spike train, i.e., how much does the spike train resemble itself when shifted by a fixed time offset, such structures lead to regular side-peaks at multiples of τ_{ref} in the ACF, $\mathcal{C}(S_x, S_x, \Delta) = \frac{\langle S_x(t)S_x(t+\Delta)\rangle - \langle S_x\rangle^2}{\text{Var}(S_x)}$, of the network-generated (excitatory or inhibitory) background S_x, $x \in \{\text{e}, \text{i}\}$. With increasing firing rates, here controlled by the leak potentials E_{l} of the background networks (blue neurons in Fig. 3.14), these side-peaks become more pronounced (Fig. 3.15A)

$$\mathcal{C}(S_x, S_x, n\tau_{\text{ref}}) \approx \sum_{k=1}^{\infty} e^{k \ln \bar{p}} \delta\big([n-k]\tau_{\text{ref}}\big)\,, \tag{3.33}$$

where $\bar{p}$ is the probability for a burst to start (see Appendix A.1.4 for a derivation). If a LIF neuron is exposed to such autocorrelated noise instead of Poisson noise, this results in

[11] By merging spike trains of several neurons, the ACF of the resulting spike train is given by the average ACF of the individual spike trains. Hence, even when forming a population average, these fixed structures remain.

a reduced width σ' of the FMP distribution (Fig. 3.15B)

$$f(u^{\text{free}}) \sim \mathcal{N}(\mu' = \mu, \sigma' = \sqrt{\beta}\sigma) \tag{3.34}$$

with scaling factor $\sqrt{\beta}$ that depends on the ACF (see Appendix A.1.2 for a derivation). This can be explained rather intuitively, as autocorrelations in the background noise reduce the variance of the noise frequency, and hence also of the FMP distribution of the receiving neuron. For instance, in case of extremely autocorrelated background noise, i.e., tonic bursting, the frequency distribution of the noise would be a Dirac delta peak. This narrowing of the FMP distribution width translates, by virtue of the threshold spike mechanism, into a modified inverse slope of the neuron's activation function

$$p(z = 1) \approx \int_{\vartheta}^{\infty} f(u)\mathrm{d}u \approx \varphi(\mu)\Big|_{u_0'=u_0, \alpha'=\sqrt{\beta}\alpha}, \tag{3.35}$$

with inflection point u_0' and inverse slope α' (see Appendix A.1.3 for details). Thus, autocorrelations in the background input lead to a reduced width of the FMP distribution and hence to a steeper activation function compared to the one obtained using uncorrelated Poisson input. For a better intuition, we used an approximation of the activation function of LIF neurons here, but the argument also holds for the exact expression derived in *Petrovici, Bill, Bytschok, Schemmel, and Meier* (2016), as verified by simulations (Fig. 3.15C).

Apart from the change in FMP width and inverse slope of the activation function, the background autocorrelations do not affect neuron properties that depend linearly on the synaptic noise input, as for instance the mean of the FMP or the inflection point $u_0' = u_0$ of the activation function (Fig. 3.15B,C). Thus, according to Eqs. 3.9 and 3.10, exchanging Poisson noise with autocorrelated background noise is equivalent to a rescaling of the Boltzmann parameters (weights W_{kj} and biases $\boldsymbol{b}$) by a factor equal to the ratio between the new and the original slope α'/α. Consequently, the network is still guaranteed to sample from a Boltzmann distribution as long as the noise characteristics are taken into account when setting synaptic connections w_{kj} and leak potentials $\boldsymbol{E}_{\text{l}}$, see Fig. 3.15D.

In addition to being autocorrelated, network-generated noise is cross-correlated, mostly due to synaptic connections between noise-providing neurons (Fig. 3.16A) or due to neurons of the principal network sharing some of their noise sources (Fig. 3.17A). Such correlations in the background translate to additional correlations between neurons in the principal network, distorting the distribution the network is sampling from.

Different from autocorrelations, background cross-correlations can be averaged out to a significant degree. If two neurons receive cross-correlated background noise, their dynamics are correlated as well, which can be measured by the correlation coefficient (CC) ρ between the FMPs of the two neurons (see Appendix A.1.5):

$$\begin{aligned}\rho(u_i^{\text{free}}, u_j^{\text{free}}) \propto \sum_{l,m} w_{il} w_{jm} \left(E_{il}^{\text{rev}} - \mu_i\right)\left(E_{jm}^{\text{rev}} - \mu_j\right) \\ \cdot \int \mathrm{d}\Delta\ \lambda_{li,mj}\ \mathcal{C}\left(S_{l,\text{i}}, S_{m,\text{j}}, \Delta\right) \widetilde{\mathcal{C}}\left(\kappa, \kappa, \Delta\right),\end{aligned} \tag{3.36}$$

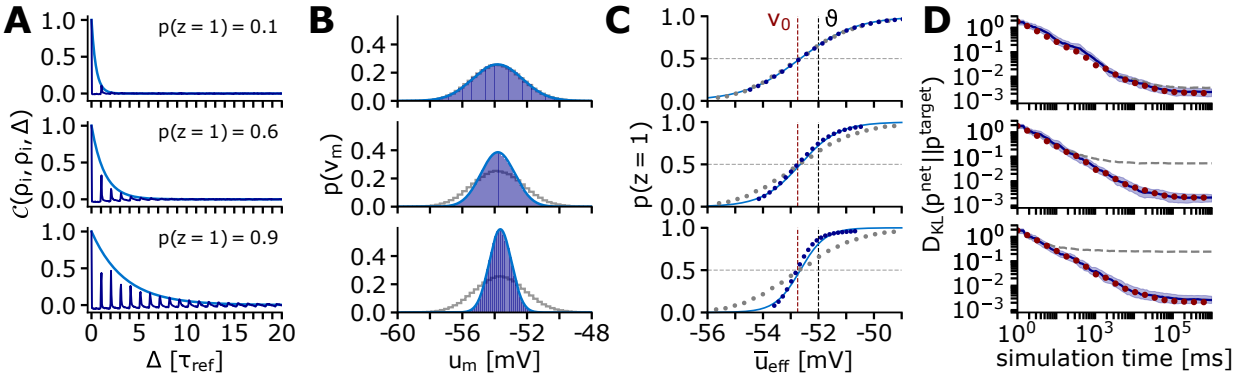

Figure 3.15.: **(A)** By appropriate parametrization of the background networks, we adjust the mean spike frequency of the background neurons (blue in Fig. 3.14) to study the effect of background autocorrelations $\mathcal{C}(S_x, S_x, \Delta)$. Higher firing probabilities increase the chance of evoking bursts, which induce background autocorrelations for the neurons in the principal network at multiples of τ_{ref} (dark blue: simulation results; light blue: $e^{k \ln \bar{p}}$ with $k = \frac{\Delta}{\tau_{\text{ref}}}$, see Eq. 3.33). **(B)** Background autocorrelation narrows the FMP distribution of neurons in the principal network: simulation (blue) and the theoretical prediction (Eq. 3.34, light blue) vs. background Poisson noise of the same rate (gray). Background intensities correspond to (A). **(C)** Single-neuron activation functions corresponding to (A,B) and the theoretical prediction (Eq. 3.35, light blue line). For autocorrelated noise, the slope of the response curve changes, but the inflection point (with $p(z = 1) = 0.5$) is conserved. **(D)** Kullback-Leibler divergence $D_{\text{KL}}\left(p^{\text{net}} \parallel p^{\text{target}}\right)$ (median and range between the first and third quartile) for the three cases shown in (A)-(C) after sampling from 50 different target distributions with 10 different random seeds for the 3-neuron network depicted in Fig. 3.14. Appropriate reparametrization can fully cancel out the effect of background autocorrelations (blue). The according results without reparametrization (gray) and with Poisson input (red) are also shown. Figures and caption adjusted from *Dold et al.* (2019a).

where l sums over all background spike trains $S_{l,i}$ projecting to neuron i and m sums over all background spike trains $S_{m,j}$ projecting to neuron j. $\widetilde{\mathcal{C}}\left(\kappa, \kappa, \Delta\right)$ is the unnormalized ACF of the PSP kernel κ, i.e., $\widetilde{\mathcal{C}}\left(\kappa, \kappa, \Delta\right) = \langle \kappa(t)\kappa(t + \Delta) \rangle$, and $\mathcal{C}\left(S_{l,\text{i}}, S_{m,\text{j}}, \Delta\right)$ the cross-correlation function of the background inputs. $\lambda_{li,mj}$ is given by $\lambda_{li,mj} = \sqrt{\text{Var}\left(S_{l,\text{i}}\right) \text{Var}\left(S_{m,\text{j}}\right)}$.

Background cross-correlations couple to the membrane potentials via the synaptic background weights $w_{il} w_{jm}$ (Fig. 3.16A). Thus, since we can assume that background correlations are identically distributed over an ensemble, even exclusively positive (or negative) cross-correlations can average out as long as the signs and strengths of the synaptic weights are drawn from a symmetric probability distribution[12] (Fig. 3.16A,B). However, the correlation coefficient also depends on the distance of the mean FMP to the reversal potentials, $\left(E_{il}^{\text{rev}} - \mu_i\right)\left(E_{jm}^{\text{rev}} - \mu_j\right)$, and therefore a complete cancelation to zero correlation between FMPs is not guaranteed. Even though this can be accounted for by changing the distribution synaptic background connections are generated from (see Appendix A.1.5), we found that for biological plausible reversal potentials, a very simple cross-wiring rule, i.e., independently and randomly determined connections, already suffices to accomplish low background

[12]This way of compensating background correlations is independent of the weight distribution of the noise-generating networks.

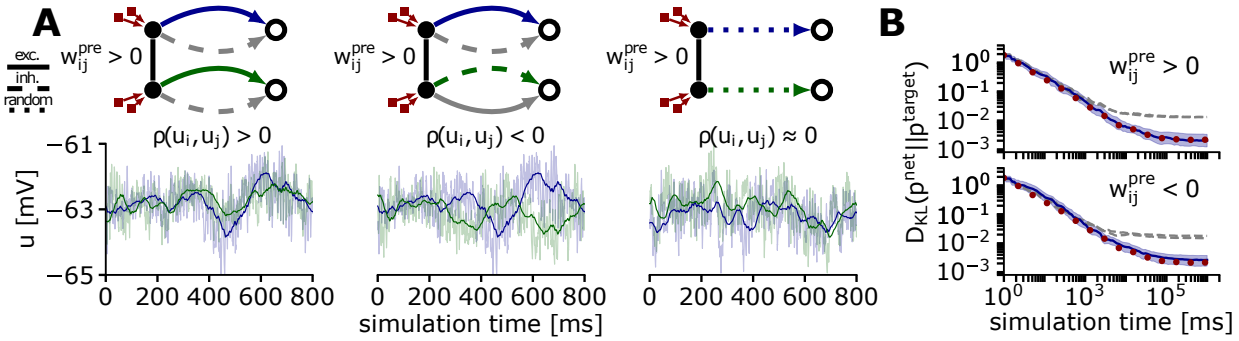

Figure 3.16.: **(A)** A pair of interconnected neurons in a background network generates correlated noise, as given by Eq. 3.36. The effect of cross-correlated background on a pair of target neurons depends on the nature of synaptic projections from the background to the principal network. Here, we depict the case where their interaction w_{ij}^{pre} is excitatory; the inhibitory case is a mirror image thereof. Left: If forward projections are of the same type, PSPs will be positively correlated. Middle: Different synapse types in the forward projection only change the sign of the PSP correlations. Right: For many background inputs with mixed connectivity patterns, correlations can average out to zero even when all input correlations have the same sign. **(B)** Same experiment as in Fig. 3.15D, with background connection statistics adjusted to compensate for input cross-correlations. The uncompensated cases from (A, left) and (A, middle) are shown in gray. Figures and caption adjusted from *Dold et al.* (2019a).

cross-correlations and therefore reach a good sampling performance (Fig. 3.16B).

The effect of cross-correlations can also be treated similarly to autocorrelations by directly looking at the influence on the sampling dynamics. For simplicity, we restrict the discussion here to the case of shared input correlations, but the observations are true for all kinds of background cross-correlations, independent of their origin. To illustrate the effect of cross-correlated background noise on the sampled distribution, we look at a network consisting of two neurons that sample from a binary Boltzmann distribution with states $(1,1)$, $(1,0)$, $(0,1)$ and $(1,1)$:

- If we introduce shared noise, this leads to a synchronization of neuronal activity, shifting probability mass from the asynchronous $(1,0)$ and $(0,1)$ states to the synchronous $(0,0)$ and $(1,1)$ states (Fig. 3.17A).
- If, instead of introducing shared noise, we change the correlation between neurons by changing the synaptic strength, this only leads to a shift of probability mass into the $(1,1)$ state, because only the $(1,1)$ state has a non-zero contribution to the energy that depends on the weights, see Fig. 3.17B.

Therefore, shared noise and synaptic weights – even though both increase the correlation between neurons – have a different effect on the shape of the encoded probability distribution. This changes when we switch from the natural[13] state space $\boldsymbol{z} \in \{0,1\}^N$ to the more

[13] The $z = 0$ state for a silent neuron is arguably more natural, because it has no effect on its postsynaptic partners during this state. In contrast, $z \in \{-1,1\}$ would, for example, imply efferent excitation upon spiking and constant efferent inhibition otherwise.

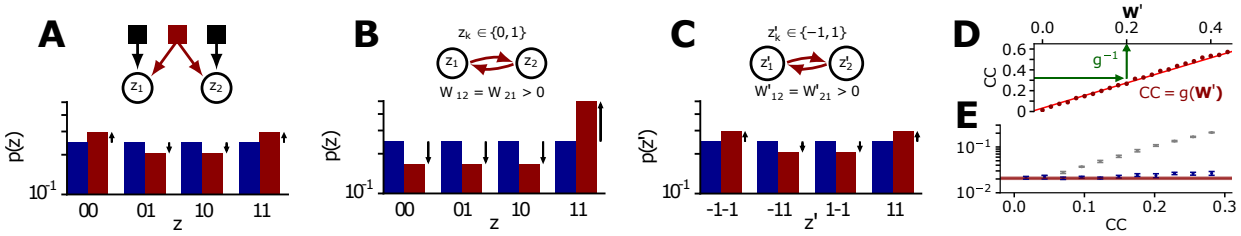

Figure 3.17.: **(A)**–**(C)** Exemplary sampled distributions for a network of two neurons. The "default" case is the one where all weights and biases are set to zero (uniform distribution, blue bars). **(A)** Shared noise sources have a correlating effect, shifting probability mass into the (1,1) and (0,0) states (red bars). **(B)** In the $\{0,1\}^2$ space, increased weights introduce a (positive) shift of probability mass from all other states towards the (1,1) state (red bars), which is markedly different from the effect of correlated noise. **(C)** In the $\{-1,1\}^2$ space, increased weights have the same effect as correlated noise (red bars). **(D)** Correlation-canceling reparametrization in the principal network. By transforming the state space from $\boldsymbol{z} \in \{0,1\}^n$ to $\boldsymbol{z}' \in \{-1,1\}$, input correlations attain the same functional effect as synaptic weights (Eq. 3.38); simulation results given as red dots, linear fit as red line. Weight rescaling (green) followed by a transformation back into the $\boldsymbol{z} \in \{0,1\}^n$ state space (which affects both weights and biases) can therefore alleviate the effects of correlated background. **(E)** Similar experiment as in Fig. 3.15D for a network with ten neurons, with parameters adjusted to compensate for input cross-correlations. As in the case of autocorrelated background, cross-correlations can be canceled out by appropriate reparametrization. Figures and caption adjusted from *Dold et al.* (2019a).

symmetric space $\boldsymbol{z}' \in \{-1,1\}^N$ (i.e., $\boldsymbol{z}' = 2\boldsymbol{z} - \mathbf{1}$). By requiring $p(\boldsymbol{z}') \overset{!}{=} p(\boldsymbol{z})$, a transformation of weights and biases between the two state spaces can be found that conserves state probabilities (i.e., conserve the energy function up to constant terms, see Appendix A.1.6):

$$\boldsymbol{W}' = \frac{1}{4}\boldsymbol{W} \text{ and } \boldsymbol{b}' = \frac{1}{2}\boldsymbol{b} + \frac{1}{4}\sum_i \mathrm{col}_i \boldsymbol{W} \,. \tag{3.37}$$

In the $\{-1,1\}$ state space, the energy of all four states depends on the weights, and increasing weights leads to a shift in probability mass from asymmetric states $(-1,1)$, $(1,-1)$ to symmetric states $(-1,-1)$, $(1,1)$ (Fig. 3.17C); as is the case for shared noise. Thus, in the $\{-1,1\}$ space, shared noise and synaptic connections have the same effect on the shape of the sampled probability distribution. Consequently, weights in the $\{-1,1\}$ space can be used to mitigate the effect of shared noise correlations. Therefore, for spiking networks, we can use the following heuristic to account for background cross-correlations: first, by changing to the $\{-1,1\}$ state space, weights w'_{ij} can be found that precisely conserves the desired correlation structure between neurons (Fig. 3.17D):

$$w'_{ij} = g^{-1}[\rho(S_i, S_j)] \approx \frac{\rho(S_i, S_j) - g_0}{g_1} \,, \tag{3.38}$$

with constants g_0 and g_1. Afterwards, this weight is mapped to Boltzmann parameters via the inverse of Eq. 3.37 and subsequently synaptic weights and leak potentials of the spiking neural network using Eqs. 3.9 and 3.10, see Fig. 3.17E.

To summarize, we found that background noise correlations (both auto and cross) are equivalent to offsets in the parameters constituting the distribution a network is sampling

from (e.g., synaptic weights); i.e., the distribution the network samples from remains in the well-defined Boltzmann domain. Thus, the effect of correlated background noise is rendered controllable through synaptic weights and biases, either (i) when using Eqs. 3.9 and 3.10 to set up spike-based sampling networks or (ii) when training networks on, e.g., visual sensory streams, as such shifts represent offsets in the network initialization and are automatically compensated for during training.

3.4.3 Deterministic probabilistic inference with spikes

In the last section, we investigated the effect of correlated background noise on the dynamics and sampling performance of spiking neural networks. We concluded that the activity of spike-based sampling networks is a sufficient source of stochasticity for other spike-based sampling networks, since all effects that follow from replacing Poisson noise in a network with functional output from other networks can be compensated by appropriate parameter adjustments. However, until now, only one (principal) network received noise from other sampling networks, whilst the others were still fed by Poisson input. To set up an ensemble of functional spiking networks that all perform different sampling tasks without explicit Poisson input, the ensemble has to fulfill a self-consistency condition: the spike statistics of the generated output has to equal the spike statistics of the expected input, i.e., the assumed noise input – or more casually phrased: ensemble output and ensemble input have to match. Inspired by the findings of the previous chapter, we found two ways to initialize ensembles in a self-consistent state while controlling the distributions each network is sampling from, which we call "calibration scheme" (Fig. 3.18) and "plasticity scheme" (Fig. 3.19) for convenience.

The idea behind the calibration scheme is based on the following observation: in the self-consistent state, each network of the ensemble samples from its target distribution given by synaptic weights and leak potentials (weights and biases). Consequently, the generated output statistics are indistinguishable from an ensemble without interconnections, where each network still receives private Poisson noise to attain stochastic firing (Fig. 3.18A). Hence, to characterize the expected background activity under in-vivo-like conditions, we simulate each network with Poisson noise first. The collected background spikes are then used to determine the activation function of each neuron, enabling us to use the translation rules Eqs. 3.9 and 3.10 to calculate the correct synaptic weights and leak potentials under ensemble background noise. This way, the ensemble is initialized in a self-consistent state, where it uses its own activity as noise instead of external Poisson source. The approach is verified for a simple test setup in Fig. 3.18B and can be used to set up arbitrary ensembles rather quickly, as will be shown in the next section.

As simple as the calibration scheme might be, it is rather inconvenient (and artificial) to record the sampling dynamics under Poisson noise in advance, before setting up the ensemble without Poisson input. Since background correlations are effectively just offsets in parameters like synaptic weights, such ensembles can be trained from scratch – which we call the "plasticity scheme". Thus, the self-consistent state can be found automatically via synaptic plasticity, independent of the underlying functionality each network has to implement. In such an approach, the whole ensemble is trained in parallel using wake-sleep learning (Eqs. 3.19 and 3.20) and each network learns to (i) perform its function and (ii)

use the available background noise to achieve the desired functionality simultaneously. That is, the ensemble is automatically shaped by synaptic plasticity to reach a functional self-consistent state where the dynamics of individual networks in the ensemble can be described via the theory of spike-based sampling (*Petrovici, Bill, Bytschok, Schemmel, and Meier*, 2016; *Dold, Bytschok, Kungl, Baumbach, Breitwieser, Senn, Schemmel, Meier, and Petrovici*, 2019a). We demonstrate this for an ensemble of 100 6-neuron sampling networks with an inter-network connectivity of $\epsilon = 0.1$ and random synaptic weights. During training, no external input is needed to kick-start neuronal activity, as some neurons spike spontaneously due to random parameter initialization. After an initially strongly synchronized phase, ensemble activity becomes asynchronous and irregular due to strong inhibitory background noise weights, starting the sampling process (Fig. 3.19A). Post-training, the ensemble reaches a sampling performance (median D_{KL}) of $\mathbf{1.06}^{+\mathbf{0.27}}_{-\mathbf{0.40}} \times \mathbf{10^{-3}}$, which is similar to the median performance of an idealized setup (independent, Poisson-driven networks as in *Petrovici, Bill, Bytschok, Schemmel, and Meier* 2016) of $\mathbf{1.05}^{+\mathbf{0.15}}_{-\mathbf{0.35}} \times \mathbf{10^{-3}}$ (errors are given by the first and third quartile), see Fig. 3.19B. To put the above D_{KL} values in perspective, we compare the sampled and target distributions of one of the networks in the ensemble at various stages of learning (Fig. 3.19C). Thus, despite the fully deterministic nature of the system, the network dynamics and achieved performance after training are essentially indistinguishable from that of networks harnessing explicit noise for the representation of probability.

In conclusion, a self-consistent ensemble configuration where output and input statistics

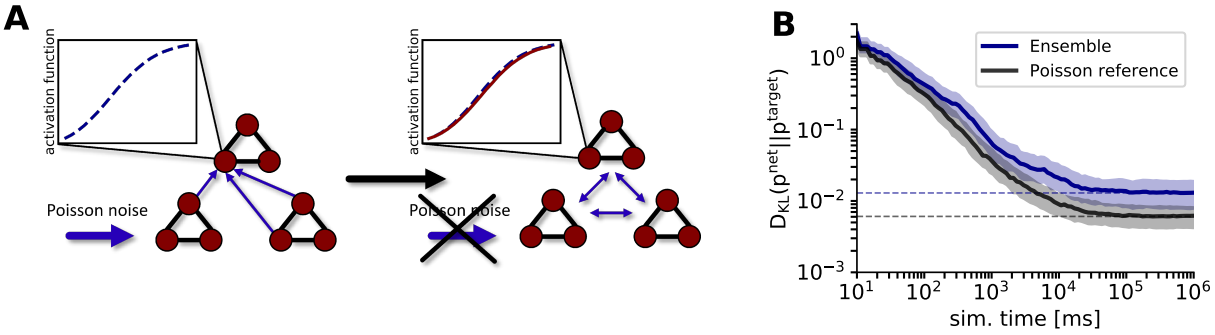

Figure 3.18.: Illustration and application of the calibration scheme. **(A)** A straightforward way to set up the parameters of each network (w_{ij} and E_l) is to use the parameter translation Eqs. 3.9 and 3.10, i.e., use the corresponding activation function of each neuron to correctly account for the background noise statistics. This is demonstrated here for the case of (left) 399 networks (only two shown) receiving Poisson noise and one network only receiving ensemble input and (right) all networks only receiving ensemble input. In both cases, the resulting activation function is the same and we can indeed use it to translate the parameters of the target distribution to neurosynaptic parameters. **(B)** Using the corresponding activation functions to set up the ensemble (but no training), each network in the ensemble is able to accurately sample from its target distribution without explicit noise, as expected from our considerations in (A) and the previous section. This is shown here for an ensemble of 400 3-neuron sampling networks with an interconnection probability of 0.05, reaching a median D_{KL} of $\mathbf{12.8}^{+\mathbf{6.4}}_{-\mathbf{5.0}} \times \mathbf{10^{-3}}$ (blue), which is close to the ideal result with Poisson noise of $\mathbf{6.2}^{+\mathbf{2.0}}_{-\mathbf{2.0}} \times \mathbf{10^{-3}}$ (black, errors given as the first and third quartile). Figures and caption adjusted from *Dold et al.* (2019a).

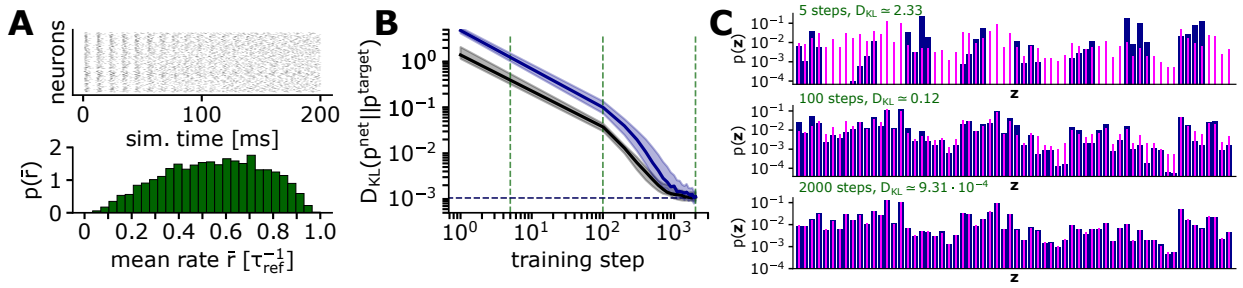

Figure 3.19.: Sampling without explicit noise from a set of predefined target distributions. **(A)** (top) Temporal evolution of spiking activity in an ensemble of 100 interconnected 6-neuron networks with no source of explicit noise. An initial burst of regular activity caused by neurons with a strong enough positive bias, i.e., leak potential above threshold, quickly transitions to asynchronous irregular activity due to inhibitory synapses. (bottom) Distribution of mean neuronal firing rates of the ensemble after training, covering the range of firing rates investigated in Fig. 3.15. **(B)** Median sampling quality of the ensemble during learning, for a test sampling period of 10^6ms. At the end of the learning phase, the sampling quality of individual networks in the ensemble (blue) is on par with the one obtained in the theoretically ideal case of independent networks with Poisson background (black). Error bars given over 5 simulation runs with different random seeds. **(C)** Illustration of a single target distribution (magenta) and corresponding sampled distribution (blue) of a network in the ensemble at several stages of the learning process (dashed green lines in (B)). Figures and caption adjusted from *Dold et al.* (2019a).

agree can be reached by virtue of biophysical features like weak inter-connectivity between modules as well as ordinary (Hebbian-like) synaptic plasticity[14]. This allows a self-contained, self-consistent and deterministic implementation of Bayesian inference in spiking neuronal systems, strengthening the biological plausibility of the original theory (*Petrovici, Bill, Bytschok, Schemmel, and Meier*, 2016) while decreasing the structural constraints for physical realizations like neuromorphic hardware.

3.4.4 Deterministic inference in hierarchical networks

Recently, spiking neural networks have been shown to yield powerful discriminative and generative capabilities comparable to abstract neural networks (*Leng, Martel, Breitwieser, Bytschok, Senn, Schemmel, Meier, and Petrovici*, 2018; *Lee, Delbruck, and Pfeiffer*, 2016; *Zenke and Ganguli*, 2018; *Kheradpisheh, Ganjtabesh, Thorpe, and Masquelier*, 2018; *Bellec, Salaj, Subramoney, Legenstein, and Maass*, 2018; *Bellec, Scherr, Hajek, Salaj, Legenstein, and Maass*, 2019; *Pfeiffer and Pfeil*, 2018). In this spirit, we extend the study of deterministic ensembles to networks trained on higher-dimensional visual input, i.e., the EMNIST data set (*Cohen, Afshar, Tapson, and van Schaik*, 2017), an extended version of the widely used MNIST data set (*LeCun, Bottou, Bengio, and Haffner*, 1998) that includes black and white images of digits as well as capital and lower-case letters with a size of 28×28 pixels. We use a hierarchical architecture (Fig. 3.20A), where inference is Bayesian in both top-down and bottom-up direction. The bottom-up pathway is used to calculate the conditional probability

[14] In Hebbian learning, weight updates are a function of the pre- and postsynaptic activity only, see Section 5.1.

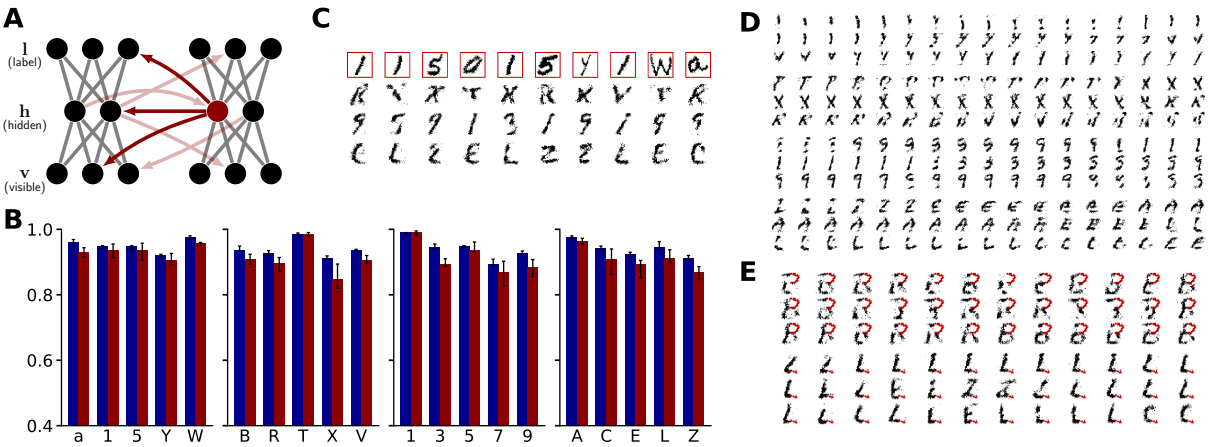

Figure 3.20.: Bayesian inference on visual input. **(A)** Illustration of the connectivity between two hierarchical networks in the simulated ensemble. Each spiking network has a visible area **v**, a hidden **h** and a label area **l** with 784, 200 and 5 neurons, respectively. Neurons in the same area of a network are not interconnected. Each neuron in a network receives only activity from the hidden areas of other networks as background (no sources of explicit noise). **(B)** An ensemble of four such spike-based sampling networks (red) is trained to perform generative and discriminative tasks on visual data from the EMNIST data set. We use the classification rate of restricted Boltzmann machines trained with the same hyperparameters as a benchmark (blue). Error bars are given (on blue) over 10 test runs and (on red) over 10 ensemble realizations with different random seeds. **(C)** Illustration of a scenario where one of the four networks (red boxes) receives visual input for classification (B). At the same time, the other networks continuously generate images from their respective learned distributions (column). The shown examples (left to right) were picked to cover a representative range of network activity. **(D)** Pattern generation and mixing during unconstrained dreaming. Here, we show the activity of the visible area of all four networks from (B), each spanning three rows. Time evolves from left to right. **(E)** Pattern completion and rivalry for two instances of incomplete visual stimuli. The stimulus consists of the top right and bottom right quadrant of the visible area, respectively. In the first run, we clamp the top arc of a 'B' compatible with either a 'B' or an 'R' (top three rows, red), in the second run we choose the bottom line of an 'L' compatible with an 'L', an 'E', a 'Z' or a 'C' (bottom three rows, red). An ensemble of networks performs Bayesian inference by implicitly evaluating the conditional distribution of the unstimulated visible neurons, which manifests itself here as sampling from all image classes compatible with the ambiguous stimulus. Time evolves from left to right. Figures and captain adjusted from *Dold et al.* (2019a).

distribution over the labels given a certain input pattern, i.e., perform classification. The top-down pathway is used to generate patterns or explain the observed input. If an occluded image is presented to the network, it is thus capable of filling in the missing details in the Bayesian sense ("pattern completion") and explore different high-probability interpretations ("pattern rivalry"). Further, if no input is given to the network, it freely samples from the prior distribution and generates patterns similar to those included in the data set used during training ("dreaming").

To set up ensembles of data-driven networks, we first pre-train restricted Boltzmann machines using the CAST algorithm (*Salakhutdinov*, 2010) and use the calibration scheme introduced in the previous section to set up a noise-free ensemble. We create an ensemble of four spike-based sampling networks with 784 visible, 200 hidden and 5 label neurons each,

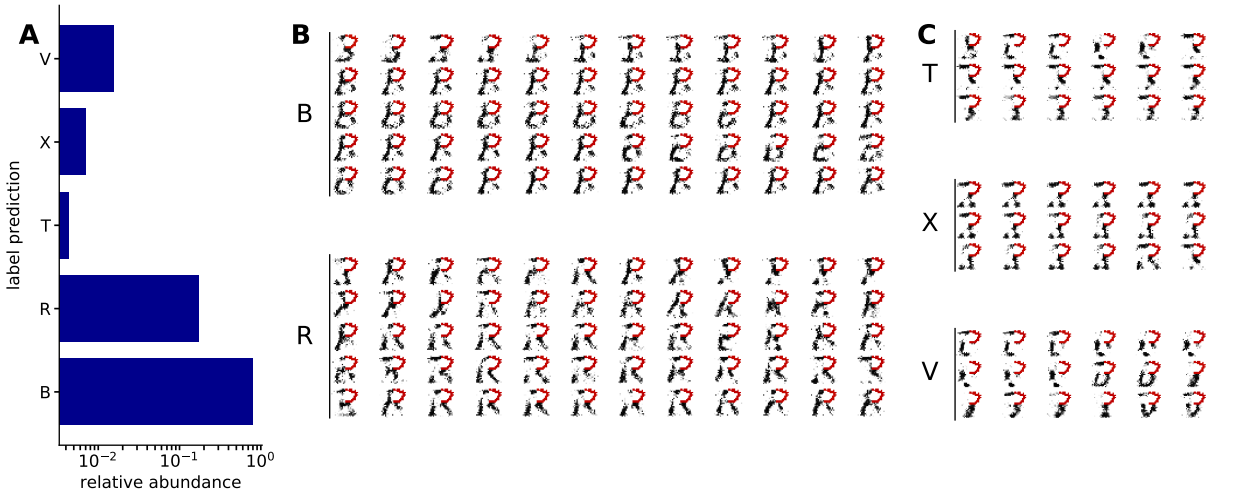

Figure 3.21.: Pattern completion in an ensemble trained on EMNIST. **(A)** Relative abundance of the label output while clamping parts of a 'B'. Most of the time (**79.85%**), the image is correctly classified as a 'B'. The closest alternative explanation, an 'R', is generated second most (**17.45%**). The remaining classes are explored significantly less often by the network (**0.43%, 0.70%, 1.57%**). **(B)** Examples of the visible area activity while the label area classifies the partially clamped images either as a 'B' (top) or an 'R' (bottom). **(C)** Examples of the visible area activity while classifying the image as a 'T', 'X' or 'V'. In these cases, the images generated by the visible neurons show prominent features of these letters. Figures and caption adjusted from *Dold et al.* (2019a).

where each network was trained on a different subset of the EMNIST data set: ({a, 1, 5, Y, W}, {B, R, T, X, V}, {1, 3, 5, 7, 9} and {A, C, E, L, Z}). This is done to imitate the modularity of cortical areas, i.e., every subnetwork performs a different task. But the approach also works if all networks are trained on the same data distribution. To validate the classification performance, we test each network on a test set consisting of 200 images per class from a separate data set, reaching a total median classification rate of $\mathbf{91.5^{+3.6}_{-3.0}\%}$ compared to $\mathbf{94.0^{+2.1}_{-1.5}\%}$ of the initial restricted Boltzmann machines Fig. 3.20B. While one of the networks receives visual input for classification, the others still freely sample from their prior distribution and produce recognizable images (Fig. 3.20C). If no input is presented to the networks, they all sample from their underlying prior distribution and generate recognizable images (Fig. 3.20D). It is important to note that without any source of external noise, the networks are capable to mix between the relevant modes (images belonging to all classes) of their respective underlying distributions, which is a hallmark of a good generative model. Moreover, when presenting a single network with ambiguous or incomplete data, for instance a partly occluded image, it is able to infer different interpretations that are compatible with the presented visual stimulus (Fig. 3.20E and Fig. 3.21). We further extend these results to an ensemble trained on the full MNIST data set, reaching a similar generative performance for all networks Fig. 3.22.

For such generative tasks like pattern completion, the key mechanism facilitating this form of stochastic exploration is noise, which in our case is provided by the functional

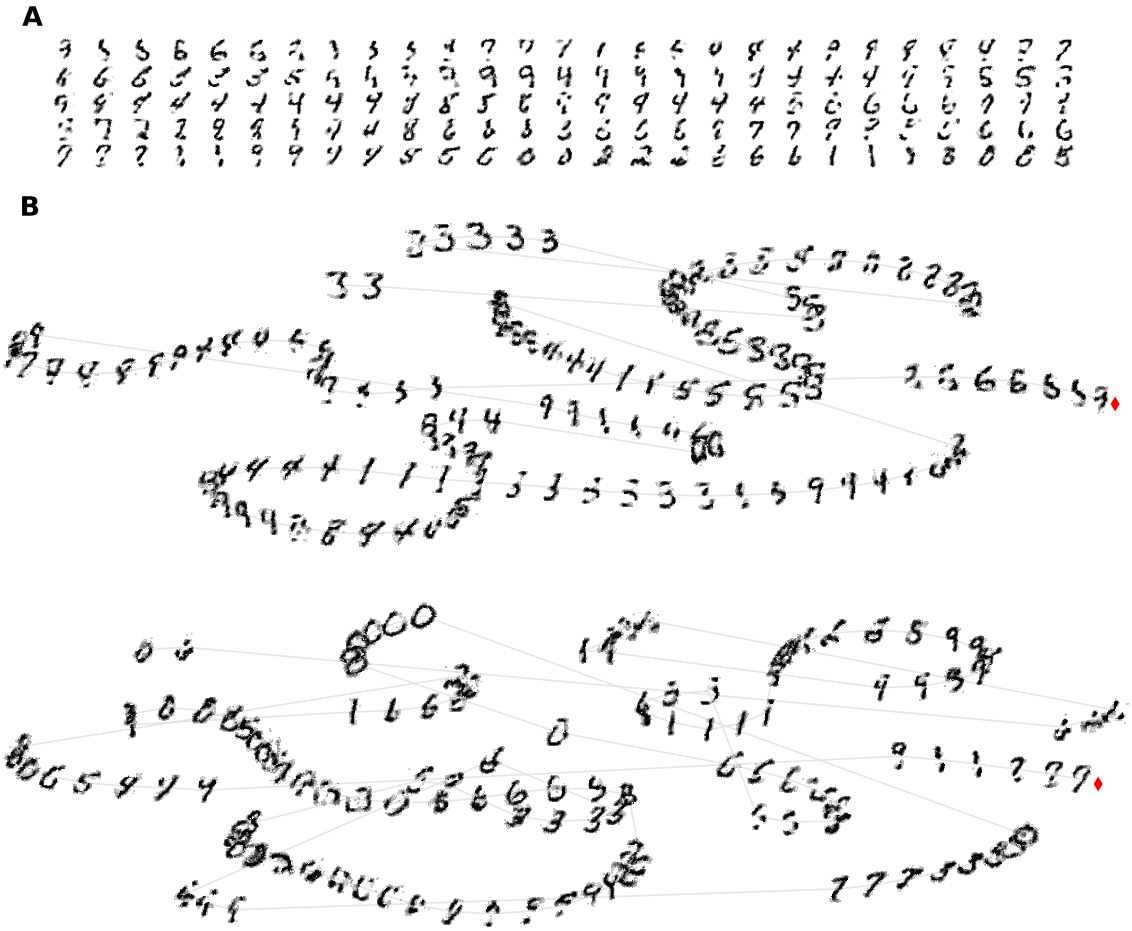

Figure 3.22.: Dreaming of MNIST in an ensemble of networks. **(A)** Dreaming ensemble of five hierarchical spiking networks with 784 visible, 500 hidden and 10 label neurons (without explicit noise). Each row represents samples from a single network of the ensembles, with samples being 375ms apart. To set up the ensemble, a restricted Boltzmann machine was trained on the MNIST data set and the resulting parameters translated to corresponding neurosynaptic parameters of the ensemble using the calibration scheme. Here, to facilitate mixing, we used short-term depression to modulate synaptic interactions and weaken attractor states that would be otherwise difficult to escape (*Leng et al.*, 2018). **(B)** t-distributed stochastic neighbor embedding (t-SNE) representation (*Maaten and Hinton*, 2008) of consecutively generated images of two of the five networks trained on MNIST digits. t-SNE is a technique for dimensionality reduction, where high-dimensional data is mapped to lower-dimensional representations by trying to preserve distances between data points locally. Both networks are able to generate and mix between diverse images of different digit classes while dreaming. The red diamond marks the first image in the sequence. Consecutive images are 400ms apart and connected by gray lines. See Appendix A.2 for a link to an online video. Figures and caption adjusted from *Dold et al.* (2019a).

activity of other neurons in the ensemble. In the presented case, the ensemble as well as the networks are rather small. We expect that such ensembles scale well to larger networks (or number of networks), where the influence of external sensory input on the ensemble

activity becomes smaller. In the limit of large ensembles, input would only locally constrain the posterior distribution of the receiving network, while the ensemble itself is still mainly driven by self-generated activity – similar to how the early visual pathway is mostly driven by recurrent circuitry and only modulated by visual input (*Fiser et al.*, 2004; *Leinweber et al.*, 2017).

3.4.5 From many to one: ensembles with only one network

Until now, we regarded an ensemble as a collection of weakly connected individual networks. However, the ensemble itself is a big, functional network, so one obvious question is whether the introduced framework also scales to single networks. This can be shown to be true for the special case of hierarchical networks, where lateral connections are introduced to act as carriers of irregularity between neurons of an area – which would otherwise not be connected, as functional connections are only between areas.

The connectivity structure of the ensembles presented in the previous sections can be described in the following form:

- Neurons are labeled in increasing order. For instance, if the ensemble consists of N networks with M neurons each, the activity vector has dimension $N \times M$, where the first M entries belong to the first network, the next M entries to the second, etc.
- With such a sorting, the functional connectivity matrix has dimension $(N \cdot M) \times (N \cdot M)$, with non-zero values on the block-diagonal (N blocks with size $M \times M$) and off-block-diagonal values as well as the diagonal being zero.
- Connections used for background noise are then added in the off-block-diagonal (i.e., connections that are not used functionally).
- In case of networks with different sizes, the same applies, just with differently sized blocks on the diagonal.

Hence, the connectivity matrix of the whole ensemble is dominated by strong symmetric connections in the block-diagonal embedded in weak unidirectional connections in the off-block-diagonal. If we look at the connectivity structure of a single restricted, hierarchical network, it takes the inverse shape of such an ensemble:

- Each area represents a sub-network in the above sense, i.e., lateral connections are on the block-diagonal again. Since in the hierarchical structure, lateral connections are suppressed, the block-diagonal is zero.
- Functional connections between areas are on the off-block-diagonal.

Therefore, the inverse logic of the ensembles can be applied here: instead of utilizing the off-block-diagonal connections as background noise connections, we can use the lateral (block-diagonal) connections. This is demonstrated in Fig. 3.23 for a single hierarchical network with 784 visible and 200 hidden units. To set up the network, we use a combination of

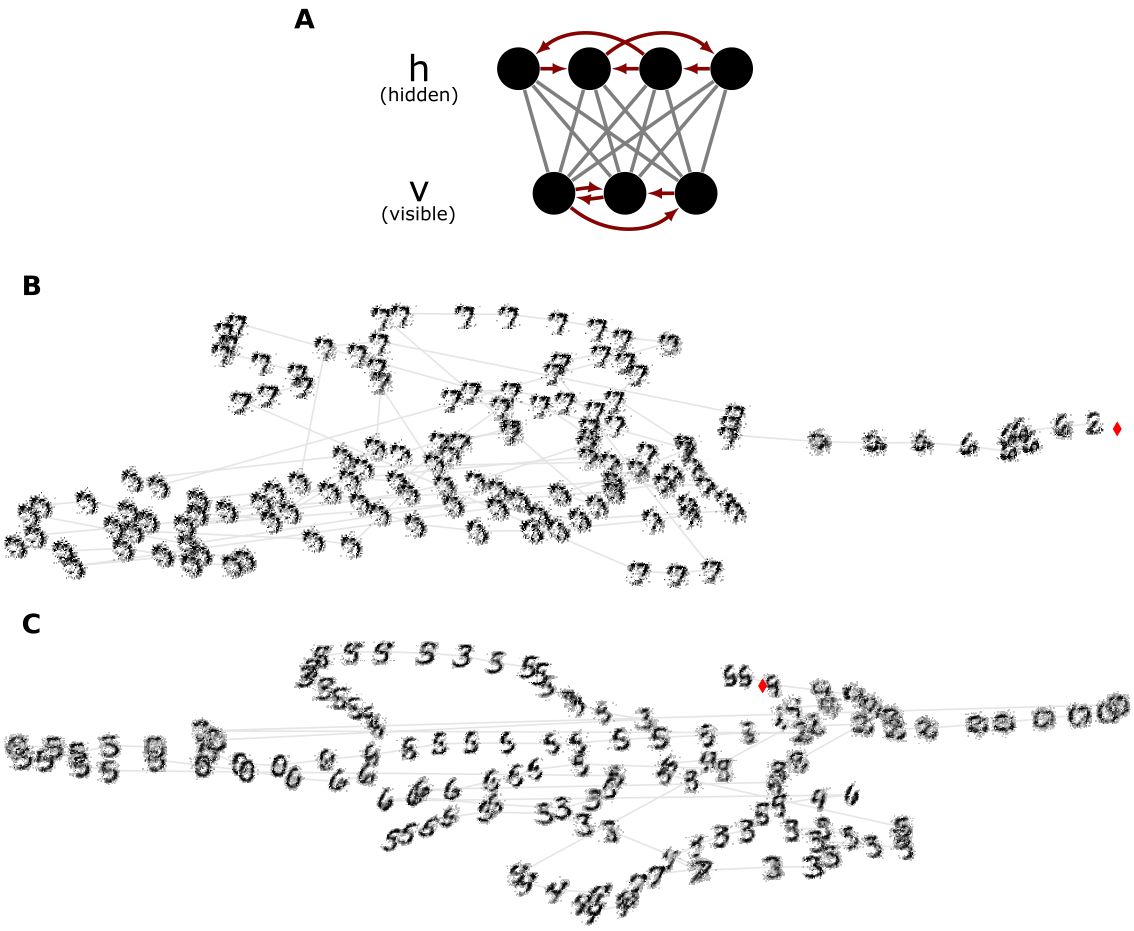

Figure 3.23.: A single hierarchical network with 784 visible and 200 hidden neurons generating samples of the MNIST handwritten digits data set without explicit noise sources, represented via t-SNE. **(A)** Illustration of the used network architecture. Lateral (non-plastic) connections in each area were utilized as a noise source (red), with an inter-connectivity of $\epsilon = 0.2$. **(B,C)** Averaged activity (average window ±90ms) of the visible area (B) after initializing the network and (C) after further training the network. After initialization, the network is able to generate recognizable images but does not mix well between different digit classes since the network is not able to correctly utilize its own background activity as noise yet (B). Further training the network on images of the MNIST training set improves both image quality and mixing (C). During the second training phase, neurosynaptic parameters are adjusted such that every neuron is able to perform its task with the available background activity it receives. See Appendix A.2 for a link to an online video. Figures and caption adjusted from *Dold et al.* (2019a).

the calibration scheme and plasticity scheme: first, to reduce simulation time, a restricted Boltzmann machine is trained on the data and translated to neurosynaptic parameters using the calibration scheme. However, since lateral noise connections have a strong influence on the correlations encoded in the hidden and visible areas, an additional training phase in the biological domain is needed (see Section 3.2.3), such that the network successfully learns to

utilize the available background activity as noise, e.g., by accounting for background noise correlations. After the training phase, the network is able to generate recognizable images of all ten digit classes it was trained on, purely generated by its own dynamics without any external or explicit source of noise. This way, a strong separation between modular clusters, as discussed in previous sections, is not needed to apply the presented principle of deterministic sampling in spiking neural networks. These results demonstrate that local and sparse circuitry within an area provides a sufficient source of neuronal noise without interfering with the functional performance of long-range connections between areas.

3.4.6 Functional ensembles on a physical neuronal substrate

The ensemble-based framework of spike-based sampling is an ideal candidate for neuromorphic implementations, allowing a self-sustained and closed physical realization of neural sampling. We prototype the approach on the BrainScaleS–1 wafer scale system (*Schemmel, Fieres, and Meier*, 2008; *Schmitt et al.*, 2017; *Kungl et al.*, 2019b), which is a mixed-signal analog device implementing neurons and synapses in analog circuits while spike transmission is digital (Fig. 3.24A, for details, see Section 2.2). As mentioned before, BrainScaleS–1 is a low-energy system and, by virtue of the analog implementation, features an acceleration of $10^3 - 10^5$ compared to real-time of the emulated biological system. However, apart from these advantages, analog systems also introduce new challenges like fixed-pattern noise (originating from the manufacturing process), spike loss due to congestion, finite resolution of parameters (e.g., 4 bit weight resolution) and limited control over adjustable neuron parameters (e.g., time constants like the refractory period). Therefore, different from a simulation where all neurons are completely identical, analog neurons are by nature of the substrate all unique in their realization. The capability of our model to work on such an imperfect substrate is important for biological plausibility, where similar mechanistic restrictions are present.

Implementing deterministic sampling ensembles on BrainScaleS–1 has three main benefits:

- it serves as an independent cross-check of the previously discussed theoretical and numerical results,
- it tests the robustness of the ensemble-framework when facing challenges like limited control over parameters and parameter precision and
- it represents a physical realization of neural sampling in a closed system (i.e., the differential equations governing the electronic circuits are similar to those of the LIF neurons), rendered controllable through synaptic plasticity.

We implemented a small ensemble of 15 4-neuron networks, with an inter-connectivity of $\epsilon = 0.2$ and a random weight and bias initialization on hardware. All networks of the ensemble were simultaneously trained in-the-loop (*Schmitt et al.*, 2017; *Kungl et al.*, 2019b; *Wu, Chua, Zhang, Yang, Li, and Li*, 2019a), i.e., spike statistics were collected from the hardware emulation and used to calculate weight and bias updates on a host computer via Eqs. 3.19 and 3.20. After training, the ensemble reaches a median sampling performance that is comparable to the ideally attainable performance on hardware (Fig. 3.24B-D) –

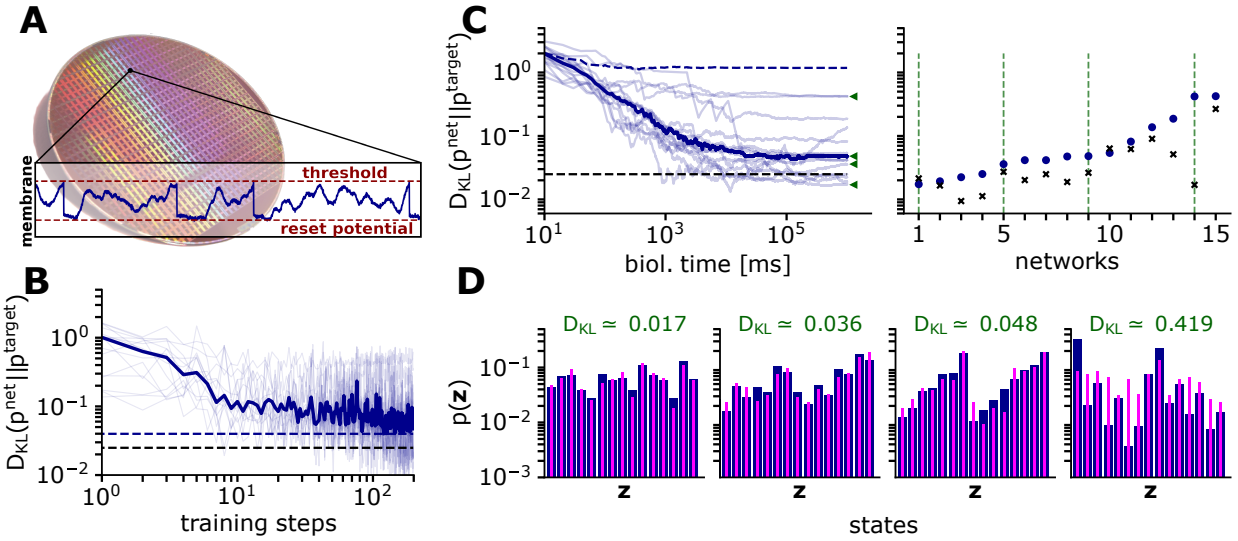

Figure 3.24.: Sampling without explicit noise from a set of predefined target distributions on a neuromorphic substrate. **(A)** Photograph of a wafer from the BrainScaleS-1 neuromorphic system used in (B), (C) and (D) before post-processing (i.e., adding additional structures like buses on top), which would mask the underlying modular structure. Blue: exemplary membrane trace of an analog neuron receiving Poisson noise. **(B)** Performance of an ensemble consisting of 15 4-neuron spike-based sampling networks with no external noise during learning on the neuromorphic substrate, shown in light blue for each network and with the median shown in dark blue. The large fluctuations compared to software simulations (Fig. 3.19B) are a signature of the natural variability of the substrate's analog components. The dashed blue line represents the best achieved median performance at $D_{\mathrm{KL}}\left(p^{\mathrm{net}} \parallel p^{\mathrm{target}}\right) = \mathbf{3.99}^{+\mathbf{1.27}}_{-\mathbf{1.15}} \cdot \mathbf{10^{-2}}$ (errors given by the distance to the first and third quartile). For comparison, we also plot the optimal median performance for the case of independent, Poisson-driven networks emulated on the same substrate, which lies at $D_{\mathrm{KL}}\left(p^{\mathrm{net}} \parallel p^{\mathrm{target}}\right) = \mathbf{2.49}^{+\mathbf{3.18}}_{-\mathbf{0.71}} \cdot \mathbf{10^{-2}}$ (dashed black line). **(C)** Left: Demonstration of sampling in the neuromorphic ensemble of spiking networks after 200 training steps. Individual networks in light blue, median performance in dark blue. Dashed blue line: median performance before training (D_{KL} of $\mathbf{1.18}^{+\mathbf{0.47}}_{-\mathbf{0.55}}$). Dashed black line: median performance of ideal networks, as in (B). Right: Best achieved performance, after 100s of biological time (10ms of hardware time) for all networks in the ensemble depicted as blue dots (sorted from lowest to highest D_{KL}). For comparison, the same is plotted as black crosses for their ideal counterparts. **(D)** Sampled (blue) and target (magenta) distributions of four of the 15 networks. The selection is marked in (C) with green triangles (left) and vertical green dashed lines (right). Since we made no particular selection of hardware neurons according to their behavior, hardware defects have a significant impact on a small subset of the networks. Despite these imperfections, a majority of networks perform close to the best value permitted by the limited weight resolution (4 bits) of the substrate. See Appendix A.2 for a link to an online video. Figures and caption adjusted from *Dold et al.* (2019a).

considering limitations like 4 bit weight resolution. As an "ideal" reference, we repeated the experiment with external Poisson noise and no interconnections between networks. Even though a few networks perform rather poorly in the "noiseless" ensemble, mostly due to misbehaving neurons (e.g., permanently bursting), the majority of networks in the ensemble is able to utilize the ensemble's background activity to approximately sample from their respective target distributions. We expect that such ensembles become more robust against

individual neuron defects when scaled up to larger ensembles and networks trained on data.

These results highlight the feasibility of noiseless sampling even under extreme conditions like small ensembles (here: in total 60 neurons) and an unreliable computing substrate.

3.5 Discussion

Starting with the trial-to-trial variability of cortical neurons as well as behavioral studies suggesting that the brain operates in a near Bayes optimal way when faced with uncertainty (Section 3.1), we presented a spike-based model that implements Bayesian computing in a robust and self-contained manner (Sections 3.2 and 3.4). The presented model acts both as a generative and discriminative model of its learned environment and can be used to perform stochastic computations on static as well as spatio-temporal tasks. Especially for the latter, we found that synaptic short-term plasticity yields an ideal mechanism to facilitate exploration of the posterior probability distribution, allowing the network to arrive at plausible solutions much faster given its current information status (Section 3.3).

The presented results propose both (i) a self-consistent and parsimonious (or in-vivo-like) implementation of spike-based sampling in biological neural networks and (ii) a possible functional purpose – giving neurons the flexibility to implement Bayesian computing – for the stereotypical architectural blueprint of strongly connected clusters embedded in a sea of weak interconnections found in the neocortex (Section 3.4).

Different to previous suggestions where the required stochasticity needed for neural sampling might originate from in the brain (*Neftci, Pedroni, Joshi, Al-Shedivat, and Cauwenberghs*, 2016), the presented framework has a sound underpinning in the theory of sampling with LIF neurons: although the presented ensembles are completely deterministic, their dynamics can not only be described stochastically as approximate sampling from Boltzmann distributions, but also be controlled and shaped to arbitrary functionality. We further expect these results to translate to other spiking neuron models, as implied by our neuromorphic implementation (Section 3.4.6), and scale to larger ensembles and networks, where the effects of input and cross-wiring correlations become less restrictive. Moreover, the results presented here reduce the architectural constraints imposed on physical neural substrates required to perform spike-based probabilistic computations both in biology and neuromorphic hardware, as illustrated in Section 3.4. It is particularly interesting that in the ensemble-based sampling networks, the biological weights $\boldsymbol{w}$ of the subnetworks are not necessarily symmetric anymore. This is because every neuron receives slightly different background noise, and hence, even though the underlying Boltzmann weight $\boldsymbol{W}$ is symmetric, the translation formula Eq. 3.9 can lead to asymmetric weights w_{ij} and w_{ji}, depending on the characteristics of the postsynaptic neuron's background noise.

We have made several assumptions to ease analysis of the proposed model, for instance, we neglected both synaptic plasticity and short term plasticity for background connections. However, if spike activity is not too high, i.e., tonic bursting only happens rarely, the effect of short term plasticity on background noise connections are negligible. In fact, we saw that tonic bursting has a negative effect on background noise quality due to its strong autocorrelation, and short term plasticity can be used to decrease the overall effect of such

strongly active neurons on noise quality. Furthermore, in future models, the clear separation into plastic and non-plastic connections might be lifted to allow functional wiring between modular areas, further narrowing the gap between noise and functionality in spiking neural networks. Plastic background connections might also be used to further shape the noise characteristics, as, e.g., proposed in Section 5.3.

Finally, the introduced framework allows a clear separation between the performed computation and its physical realization – both in terms of biology as well as analog neuromorphic hardware. Thus, the main results of this chapter can be summarized with Marr's three levels of analysis (*Marr*, 1982):

1. **Computational level:** Processing of sensory stimuli using probabilistic inference.
2. **Algorithmic level:** Markov Chain Monte Carlo sampling.
3. **Implementation level:** Ensembles of LIF neurons – both as mathematical model or physical realization – shapeable by sensory stimuli using a Hebbian-like plasticity rule. Stochasticity is provided by the ensemble's own spiking activity, while spikes represent samples.

The presented results demonstrate that rich network dynamics arise in a weakly coupled network of networks, describable as a stochastic system that satisfies a self-consistency condition: noise input equaling the generated functional output – thus being completely self-sufficient.

Details about simulations, calculations and implementations in this chapter can be found in Appendix A.

3.6 Contributions

The results and figures of Section 3.3 are taken from *Zenk* (2018) and were produced by Maximilian Zenk, whom I supervised during his Master's thesis. The work in Section 3.4 is a direct continuation of my Master's thesis (*Dold*, 2016; *Bytschok*, 2017). Compared to the preceding work, a rigorous mathematical analysis has been added as well as conceptually important experiments showing that ensembles of deterministic networks (i) can be trained from a random initial state to perform sampling of their respective target distributions, (ii) are considerably robust and allow an implementation on analog neuromorphic hardware, (iii) can be scaled up to hierarchical networks performing inference in high-dimensional spaces and (iv) can be reduced to single, self-noising networks. Experimental results concerning the characterization of correlated background noise that also appear in *Dold* (2016); *Bytschok* (2017) have been reproduced and extended, representing original work. The results have been published in *Dold, Bytschok, Kungl, Baumbach, Breitwieser, Senn, Schemmel, Meier, and Petrovici* (2019a).

4 | Intermezzo

In the previous chapter, we started by observing that neurons behave stochastically in vivo, largely due to a strong background bombardment from 10^3 to 10^4 adjacent neurons. Functionally, we showed that this background activity might be a key enabler of a probabilistic computing scheme in the brain – or at least in brain-like devices – where spikes take on the role of probabilistic samples from a posterior distribution as well as the role of background noise ("temperature" in physics, see *Korcsak-Gorzo* 2017; *Baumbach, Schemmel, and Petrovici* 2019) to drive exploration of a posterior probability landscape in the first place (Fig. 4.1, left). If trained on sensory stimuli, a spike-based sampling network approximates the environment in terms of a generative probability distribution. This allows the network to

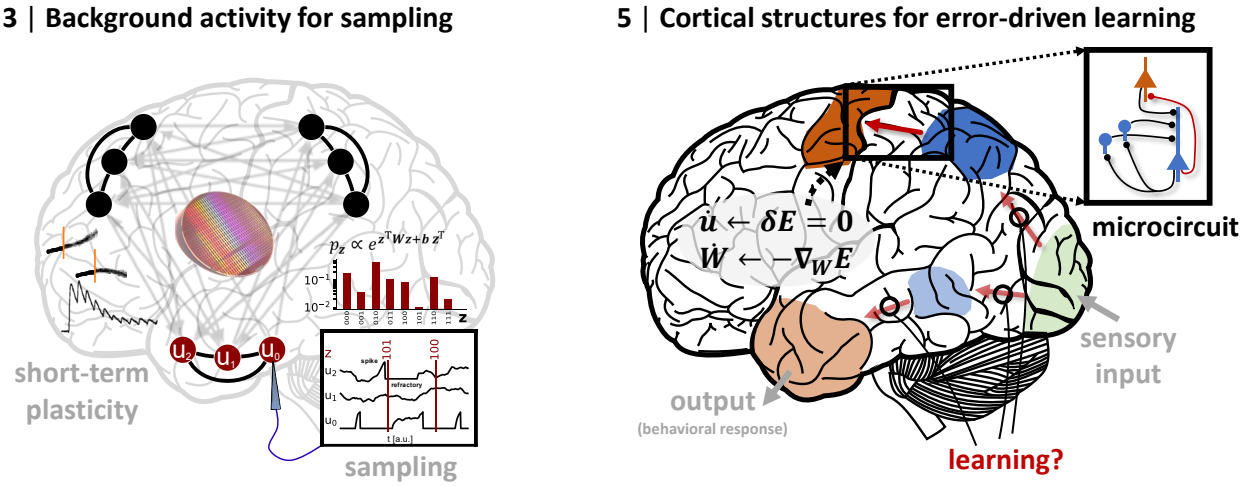

Figure 4.1.: **3** | Schematic summary of the previous chapter. An ensemble of functional spiking neural networks, each performing stochastic inference, provides itself with the required noise for probabilistic computations – in this case sampling from a learned probability distribution. Thus, stochasticity is attained from a modular and weakly interconnected network structure, a blueprint also found in the neocortex. Especially for spatio-temporal prediction tasks, the performance of such networks can be improved with a simple mechanism of real synapses: short-term plasticity. **5** | In the next chapter, we explore how learning of hierarchical cortical areas (colored areas) can be phrased as an optimization procedure, solvable via gradient-based methods. We propose a top-down approach, deriving neuronal and synaptic dynamics from first principles (formulas), from which we deduce both a link to the well-known error backpropagation algorithm as well as a biological implementation of the algorithm using local microcircuits and advanced neuronal responses. Figures adjusted from https://commons.wikimedia.org/wiki/File:Brain_Surface_-Gyri.SVG (version: 11:12, 30 March 2010).

make predictions about the environment and modulate its own perception when presented with insufficient or unreliable inputs – which is mathematically described by sampling from the conditional probability distribution.

Although Boltzmann machines have great representational power, they fell out of favor in the AI field because of the hardships in training them: different from supervised models trained end-to-end using error backpropagation (*Linnainmaa*, 1970; *Werbos*, 1982; *Rumelhart, Hinton, and Williams*, 1986), stochastic models like deep Boltzmann machines have to be trained areawise in a greedy way[1], i.e., never as a whole (*Hinton and Salakhutdinov*, 2006; *Salakhutdinov and Hinton*, 2009). Furthermore, the wake-sleep algorithm is evaluated using sampling, which requires that the MCMC has sufficiently converged before calculating parameter updates, different from error backpropagation where only a single forward and backward evaluation of the network is needed. In the next chapter, we explore a biologically plausible learning framework that approximates the error backpropagation algorithm and hence allows end-to-end learning of cortex-like networks (Fig. 4.1, right). To have a close analogy to deep learning, we first investigate non-probabilistic supervised learning models; though at the end, we also propose how the presented model could be extended to probabilistic, generative models. Furthermore, for mathematical convenience we drop the assumption of spike-based neurons and work with rate-based models (Eq. 2.2) instead – setting the focus more on the actual implementation of error backpropagation in cortical tissue and less on the neuronal coding scheme.

In the following chapter, we first introduce the error backpropagation algorithm and discuss why it is, at least in its original form, not a suitable model for learning in the brain (Section 5.1). Afterwards, we introduce a novel framework where coupled neuronal and synaptic dynamics implement approximate real-time error backpropagation in a biologically plausible way, i.e., the derived dynamics are directly mappable to components of cortical networks (Section 5.2). Finally, we propose an extension of the derived model to unsupervised learning as well as generative models (Section 5.3).

[1] The idea behind areawise training is to slowly build up a hierarchical network area by area. Thus, a first area is trained unsupervised on data to learn useful representations. Then, the output of this area is used to train another area, etc. This way, each area learns features useable by subsequent areas, which are added one by one during the training process. In contrast, if the whole network is trained using wake-sleep, weight updates in higher areas will not harmonize with updates in lower areas, as lower-area updates affect higher-area activities.

5 | Deep learning in mechanistic models of cortical networks

(...) and this is the miraculous fact on which the rest of AI stands. It is the fact that when you have a circuit, and you impose constraints on your circuit using data, you can find a way to satisfy these constraints using backprop; by iteratively making small changes to the weights of your neural network until its predictions satisfy the data.

Ilya Sutskever, MIT Artificial General Intelligence (AGI)
"Meta-Learning and Self-Play", 1.02.2018

5.1 Deep learning in the brain?

In the past years, deep learning methods achieved impressive results previously thought to still remain elusive for a long time (*LeCun et al.*, 2015; *Ciregan et al.*, 2012; *Krizhevsky et al.*, 2012; *Goodfellow et al.*, 2014; *Silver et al.*, 2017; *Vaswani et al.*, 2017), reforming our understanding and the future role of AI (*Brooks et al.*, 2012; *Ng*, 2016; *Hassabis et al.*, 2017; *Sejnowski*, 2018; *Richards et al.*, 2019; *Sejnowski*, 2020). However, compared to abstract neural networks used in deep learning, their biological archetypes studied in computational neuroscience still lag behind in performance and scalability (*Pfeiffer and Pfeil*, 2018; *Davies*, 2019).

One of the crucial ingredients of deep learning is the backpropagation-of-errors algorithm (*Linnainmaa*, 1970; *Werbos*, 1982; *Rumelhart, Hinton, and Williams*, 1986; *LeCun, Bengio, and Hinton*, 2015), which enables end-to-end learning of hierarchical neural networks, driven by an output error formed through the network-generated output (given an input) and a target output (the teaching signal). In contrast, learning in the cortex is believed to be largely driven by correlation-based rules – devoid of any output errors; a type of learning that can be summarized by Hebb's rule: *what fires together wires together* (*Hebb*, 1949). One realization of this paradigm observed in biology is spike time dependent plasticity (STDP), where synaptic plasticity is modulated by the spike time difference of pre- and postsynaptic neurons (e.g., if pre- and postsynaptic spike times are causal, the synapse is strengthened, and weakened otherwise), see *Bi and Poo* (2001); *Sjöström and Gerstner* (2010). Such Hebbian rules do not scale well to deeper networks and are mostly confined to shallow learning (no hidden areas),

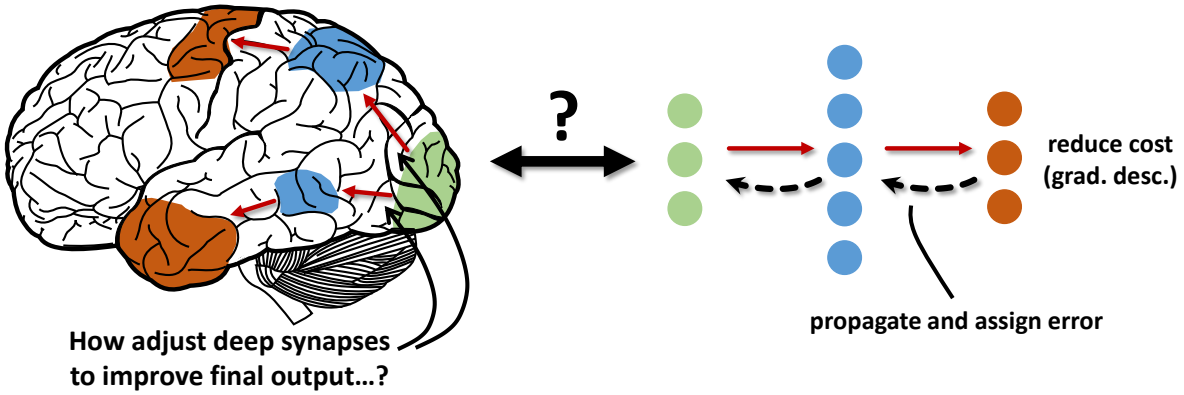

Figure 5.1.: How are deep synapses in the brain adjusted such that neuronal activity in lower areas is more useful for higher cortical areas (left)? One possible solution to this problem is given by the error backpropagation algorithm (right): if we assume that the highest area encodes the behavioral response, we can define an output cost measuring the network's performance in a given task. Synaptic weights are then adjusted via gradient descent on the cost function. In case of the output area, learning is driven by an output error derived from the cost. This error is backpropagated through the network to adjust deep synapses, increasing the network's overall performance on the task to be learnt. Figure adjusted from https://commons.wikimedia.org/wiki/File:Brain_-Surface_Gyri.SVG (version: 11:12, 30 March 2010).

being quite inferior in their learning capabilities compared to deep learning methods (*Zhong, Ling, and Wang*, 2019). For linear networks and under specific conditions, one can show that both learning paradigms – correlation-based learning as well as error-based learning – become equivalent (*Xie and Seung*, 2003). Recent results for non-linear networks also show that at least with one hidden area, Hebbian-based learning can achieve competitive results on some machine learning benchmarks (*Kheradpisheh, Ganjtabesh, Thorpe, and Masquelier*, 2018; *Illing, Gerstner, and Brea*, 2019; *Krotov and Hopfield*, 2019), although these approaches lack the mathematical underpinning (i.e., performing gradient descent on an arbitrary cost function) and scalability of the backpropagation algorithm (being effortlessly extendable to deeper architectures).

The main challenge of training deep neural networks is the following: if we have a network with, e.g., three areas, how should the synaptic weights from the first to second area be changed such that the neurons in the last area benefit from this change – i.e., such that the change leads to an improved network performance on a given task we want to learn? This is generally known as the "credit assignment problem", meaning: if the output of the network is erroneous, how do we assign credit (or blame) to the deeper synapses, far away from the output error, such that changing their weight improves the outcome (Fig. 5.1). Error backpropagation solves this problem by assigning each synapse an error signal, iteratively derived from the output error. However, in its classical form, error backpropagation requires information that is not locally available to neurons (different from correlation-based learning rules that only access pre-and postsynaptic quantities), and hence, for a long time, it was believed to be biologically implausible and not relevant for understanding learning in the brain (*Crick*, 1989). But to cite Geoffrey Hinton (*Hinton*, 2007):

> *Do you really believe that evolution could not find an efficient way to adapt a*

> *feature so that it is more useful to higher-level features in the same sensory pathway? (have faith!)*

And in fact, in the light of recent evidence, the debate whether learning in the brain can be described as gradient-based optimization has been rekindled again (*Richards et al.*, 2019). In particular, abstract neural networks trained with gradient-based methods – mostly error backpropagation – on sensory inputs have been found to reproduce activity patterns found in deeper cortical areas surprisingly well – even better than biophysical models constructed bottom-up to model cortical structures and dynamics (*Khaligh-Razavi and Kriegeskorte*, 2014; *Yamins, Hong, Cadieu, Solomon, Seibert, and DiCarlo*, 2014; *Yamins and DiCarlo*, 2016; *Schrimpf et al.*, 2018). Additionally, it has been found that cortical structures like feedback connections can be used in combination with local Hebbian plasticity rules to approximate error backpropagation in a biological plausible way (*Whittington and Bogacz*, 2019). However, these models often still come with drawbacks of their own, like separation of neuronal and synaptic dynamics or learning algorithms with several disjunct phases, contradicting the dynamic and time-continuous nature of the brain.

Here, we propose a novel model that allows an implementation of error backpropagation in time-continuous systems. Starting from an abstract energy function, we derive equations of motion that can be mapped to features found in the neocortex, like pyramidal neurons and local microcircuits (Section 2.1.3) as well as local plasticity rules. The top-down approach allows us to tackle the problem of error backpropagation on all three of Marr's levels of analysis (*Marr*, 1982), with a stacked implementation level:

1. **Computational level:** learning of input-output relations in time-continuous neuronal systems.

2. **Algorithmic level:** gradient-based optimization with error backpropagation.

3.1 **1st implementation level:** synaptic dynamics minimizing an energy function E and neuronal dynamics minimizing the corresponding action $A = \int E \mathrm{d}t$.

3.2 **2nd implementation level:** the derived dynamics can be mapped to cortical structures like pyramidal neurons with feedback connections and local interneuron microcircuits, again describable by dynamics as in 3.1.

In the following, we will first discuss the error backpropagation algorithm and its challenges concerning biological plausibility before introducing the proposed model.

5.1.1 The error backpropagation algorithm

The backpropagation-of-errors algorithm (backprop) is an elegant and simple approach to train a hierarchical network of neurons, extendable to recurrent networks (*Rumelhart, Hinton, and Williams*, 1986; *Williams and Zipser*, 1989; *Werbos et al.*, 1990). In general, abstract neural networks equipped with non-polynomial activation functions φ are universal function approximators (*Leshno, Lin, Pinkus, and Schocken*, 1993), i.e., the neural network itself represents a flexible model parameterized by weights which can be learned end-to-end

from data[1] by only providing inputs and target outputs (e.g., images and labels, or when thinking of function fitting, $\boldsymbol{x}_i$ and $f(\boldsymbol{x_i})$ for $i = 1...N$ observations). An illustrative example can be given for image recognition techniques: instead of engineering features that allow classification of images into several categories (e.g., edge and corner detectors), a neural network trained with error backpropagation learns useful features automatically (*LeCun, Bengio, and Hinton*, 2015). This way, the network can adequately transform the input into a linearly separable representation, making classification thereafter a simple task.

To derive the error backpropagation algorithm, we first define a hierarchical neural network with N areas (called "layers" in deep learning) according to Eq. 2.1

$$u_l = W_l \, \bar{r}_{l-1} \quad \text{with} \quad \bar{r}_{l-1} = \varphi(u_{l-1}) \quad \text{and} \quad l = 1...N \,, \tag{5.1}$$

with u_l being the vector of membrane potentials in area l and weights W_l projecting from neurons in area $l-1$ to neurons in area l (Fig. 5.2). We dropped the bold notation for vectors and matrices here. To optimize the network parameters, a cost function C (also called loss or error function) is introduced that measures how well the network performs, e.g., by taking the Euclidean norm of the difference between the prediction[2] of the last area N and the actual target output u_N^{target}

$$C = \frac{1}{2} \| u_N^{\text{target}} - u_N \|^2 \,. \tag{5.2}$$

Training the network now consists of reducing the average cost on the given training data set, which is achieved via gradient descent optimization

$$\Delta W_l = -\eta \left\langle \frac{\partial C}{\partial W_l} \right\rangle_{\text{data}} \,, \tag{5.3}$$

yielding for the last area, assuming a single data sample,

$$\Delta W_N = -\eta \frac{\partial C}{\partial W_N} = -\eta \frac{\partial C}{\partial u_N} \frac{\partial u_N}{\partial W_N} = \eta \, (u_N^{\text{target}} - W_N \bar{r}_{N-1}) \, \bar{r}_{N-1}^{\mathrm{T}} = \eta \, \bar{e}_N \, \bar{r}_{N-1}^{\mathrm{T}} \,, \tag{5.4}$$

where we used $u_N = W_N \bar{r}_{N-1}$ and introduced the scalar learning rate η and output error $\bar{e}_N = -\frac{\partial C}{\partial u_N}$. To derive the updates of the weights for the next area, we use the chain rule

$$\Delta W_{N-1} \propto -\frac{\partial C}{\partial W_{N-1}} = -\frac{\partial C}{\partial u_{N-1}} \frac{\partial u_{N-1}}{\partial W_{N-1}} = \bar{e}_{N-1} \bar{r}_{N-2}^{\mathrm{T}} \,, \tag{5.5}$$

with the backpropagated error

$$\bar{e}_{N-1} = -\frac{\partial C}{\partial u_{N-1}} = -\frac{\partial C}{\partial u_N} \frac{\partial u_N}{\partial u_{N-1}} = \bar{r}'_{N-1} \odot W_N^{\mathrm{T}} \bar{e}_N \,, \tag{5.6}$$

[1] To cite Yann Lecun: "DL [Deep Learning] is constructing networks of parameterized functional modules & training them from examples using gradient-based optimization", https://twitter.com/ylecun/status/1215744205919670272, 10.01.2020.

[2] In deep learning, the last area is normally linear, normalized with a softmax function. To ease the comparison with theory introduced later, we define the cost function over the membrane potentials instead.

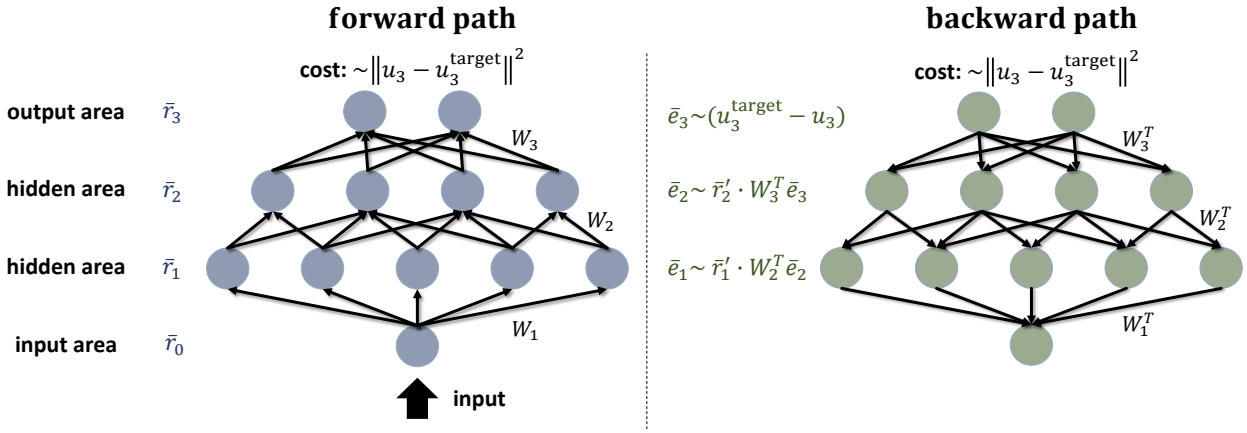

Figure 5.2.: Illustration of the error backpropagation algorithm, which is separated into two distinct phases. (left) An input is presented to the network, which leads to a forward propagation of activity, resulting in an output. The cost function measures how well the network output matches the target output. (right) Weight updates are defined as an optimization problem: minimizing the cost function. The key property of error backpropagation is that weight updates (or rather errors used for weight updates) are calculated iteratively in a mirror network, given as the transposed of the forward network.

where $\odot$ is element-wise multiplication. Notice that, due to the feedforward structure of the network, we can pretend that the network only consists of $N-1$ areas with output error $\bar{e}_{N-1}$ (since area N does not influence area $N-1$). In this case, we can again apply the chain rule as in Eqs. 5.5 and 5.6, from which we find an iterative formula for the errors:

$$\bar{e}_l = \bar{r}_l' \odot W_{l+1}^{\mathrm{T}} \bar{e}_{l+1} \,, \tag{5.7}$$

$$\Delta W_l = \eta \, \bar{e}_l \, \bar{r}_{l-1}^T \,. \tag{5.8}$$

This is commonly known as the error backpropagation algorithm: in order to reduce the cost, first an input is given to the network and its output is calculated ("forward path"). From this, the output error $\bar{e}_N$ is calculated, which is propagated through a mirror network (with transposed weights) to obtain the weight updates for every area iteratively ("backward path"), see Fig. 5.2.

Although the error backpropagation algorithm is rather simple and has become the gold standard for training abstract neural networks (*LeCun, Bengio, and Hinton*, 2015), it turns out to be hard to find ways how the brain might implement an optimization scheme like backprop – or task-dependent gradient-based optimization in general.

5.1.2 The challenges of backprop in the brain

Several problems arise when exploring how the error backpropagation algorithm might be physically realized, e.g., in the brain. First, the weights of the forward and backward path are tied, i.e., the weights used for error backpropagation perfectly track the forward weights, making the algorithm non-local. Furthermore, the algorithm is made up of distinct phases, meaning that forward pass and backward pass happen separately, and each network area

has to wait until the error signal arrives and plasticity can be applied – the areas are "locked" together. Hence, weight updates cannot be done in parallel, but are strictly limited by the sequential roll out of the two mentioned phases. In addition, backprop does not provide the means how the error signal is calculated, stored, propagated through the network and accessed by synapses in a biologically feasible manner.

Recently, several solutions have been proposed for these problems. A major breakthrough was the realization that random feedback weights are sufficient to reach competitive performance in classification and regression tasks, with the effect that forward weights align with the feedback weights during training to approximate error backpropagation (*Lillicrap, Cownden, Tweed, and Akerman*, 2016; *Nøkland*, 2016; *Moskovitz, Litwin-Kumar, and Abbott*, 2018; *Frenkel, Lefebvre, and Bol*, 2019). In *Jaderberg, Czarnecki, Osindero, Vinyals, Graves, Silver, and Kavukcuoglu* (2017), a model was proposed solving the areawise locking during learning: instead of waiting for the backpropagated error, additional networks learn to approximate the errors instead (so-called "synthetic gradients"), only requiring the activity of the corresponding area. In an effort to push error backpropagation closer to neuroscience, *Scellier and Bengio* (2017, 2019) demonstrated how neuronal dynamics and synaptic plasticity rules can be derived from a single scalar function[3], implementing an approximate version of error backpropagation. *Whittington and Bogacz* (2017) connected the widely known predictive coding framework (*Rao and Ballard*, 1999) to the error backpropagation algorithm, and in *Sacramento, Costa, Bengio, and Senn* (2018); *Guerguiev, Lillicrap, and Richards* (2017), a microcircuit of pyramidal and interneurons (see Section 2.1.3) was used to calculate and propagate errors in simulated cortical tissue, demonstrating how such error signals might become available to a local plasticity rule to implement approximate error backpropagation[4].

However, in these models, neuronal and synaptic dynamics either require a separation of time scales, with neuronal dynamics occurring much faster than or separately from synaptic weight changes, or the learning rule consists of two distinct phases like in wake-sleep (see Section 3.2.3). Therefore, the studied systems are not time-continuous during learning, unlike real neuronal and synaptic systems. In the following, we introduce a novel model that extends the previous ideas and derives a real-time version of error backpropagation from a least-action principle. More specifically, we show how learning can happen without requiring a separation of time scales or distinct learning phases. The derived model is compatible with cortical structure and dynamics, suggesting that it might be portable to brain-inspired neuromorphic systems, which often inherit many physical constraints from their biological archetype.

[3] *LeCun, Touresky, Hinton, and Sejnowski* (1988) presented a similar framework to derive error backpropagation using Lagrange multipliers.

[4] For the specific case of deep reinforcement learning, an implementation of backprop using error neurons was further proposed in *Pozzi, Bohté, and Roelfsema* (2018).

5.2 Real-time error backpropagation in cortical circuits

5.2.1 A principled approach to learning in dynamical systems

To tackle the question of deep learning in dynamical systems like the brain, we choose a top-down approach: starting with an abstract, scalar representation of cortical networks (a so-called "energy function"), we derive equations of motion for a dynamical system capable of learning. From the shape of the derived dynamical equations, we are then able to directly map these dynamics to biological neural systems. Such an approach allows a smooth transition from high-level, abstract theory, to possible biophysical implementations of the derived dynamics as well as its algorithmic functionality[5].

The dynamics and structure of the neural network (the "neural code") is compressed in the energy function E (*Rao and Ballard*, 1999; *Whittington and Bogacz*, 2017)

$$E = V + \beta C = \sum_{l=1}^{N} \frac{1}{2}\|u_l - W_l \bar{r}_{l-1}\|^2 + \frac{\beta}{2}\|u_N^{\text{target}} - u_N\|^2 , \tag{5.9}$$

composed of a "prediction error" (or "mismatch energy"[6]) V and a cost function C, weighted by a scalar prefactor β. For simplicity[7], we choose a multi-area network (or "layered" network in deep learning) with N areas here. The prediction error is given by the difference of two components: bottom-up input $W_l \bar{r}_{l-1}$ entering area l and the membrane potentials u_l of the neurons in area l. W_l are the weights of neurons projecting into area l and $\bar{r}_{l-1} = \varphi(u_{l-1})$ is the vector of stationary rates of neurons in the previous area $l-1$, given by an activation function φ. We can interpret the bottom-up input from area $l-1$ as a predictor of the subsequent area's membrane potentials. Hence, the prediction error tells us how much of the activity in area l is due to the input coming from area $l-1$, i.e., how well can the network explain its own network state. The cost function C can be seen as a prediction error as well, but to an imposed target u_N^{target} that is not generated by the network's own activity, but externally provided. In case of a hierarchical feedforward network, the last area could, for instance, encode possible classification labels of visual inputs – with u_N^{target} being the correct labels. The cost function then simply measures how well the network performs in its classification task.

The goal of learning now is to adjust the model parameters – the weights – such that the average cost over a given data set we want to model is minimized. The following weight

[5] This is similar to how we model neuronal systems and their algorithmic functionality in the previous chapter: from top-down, we described neuronal dynamics as sampling from a Boltzmann distribution characterized by an energy function E which encodes the network structure and interaction strengths between neurons. These dynamics are implemented by a dynamical system using spikes (LIF neurons), which are a mathematical abstraction of real biological neurons.

[6] Later, we propose a physical realization of this term, from which V can be interpreted as a mismatch energy.

[7] The presented theory trivially generalizes to non-layered networks (e.g., recurrent networks).

dynamics minimize the cost[8] :

$$\frac{1}{\eta}\dot{W}_l = -\frac{\mathrm{d}C}{\mathrm{d}W_l} = -\lim_{\beta\to 0}\frac{1}{\beta}\frac{\partial E}{\partial W_l} = \lim_{\beta\to 0}\frac{1}{\beta}\big(u_l - W_l\,\bar{r}_{l-1}\big)\,\bar{r}_{l-1}^{\mathrm{T}}\,, \tag{5.10}$$

with learning rate $\eta \geq 0$ and the condition that the neuronal dynamics satisfy $\frac{\partial E}{\partial u_l} = 0$ for all areas as well as $E = 0$ for $\beta = 0$, that is, when no target signal is imposed, the prediction errors vanish and synaptic plasticity stops automatically.

The derived plasticity rule has an immediate biological interpretation: if we assume pyramidal neurons (Section 2.1.3), then $W_l\bar{r}_{l-1}$ are the inputs to the basal compartments and u_l the somatic potentials. The plasticity rule can then be phrased as the basal prediction of the somatic potential (*Urbanczik and Senn*, 2014). Thus, if no apical inputs arrive at the pyramidal neuron, the somatic potential simply follows the basal input and Eq. 5.10 vanishes. Otherwise, plasticity will increase (or decrease) the basal input until it matches (or predicts) the somatic potential correctly. With this interpretation, the synaptic plasticity is completely local and has the attractive property that learning stops automatically, different from purely Hebbian learning rules.

To make the connection to *learning from the dendritic prediction of somatic spiking* (*Urbanczik and Senn*, 2014) more explicit, we can approximate Eq. 5.10 as

$$\dot{W}_l = \eta\big(u_l - W_l\,\bar{r}_{l-1}\big)\,\bar{r}_{l-1}^{\mathrm{T}} \approx \eta\big(\varphi(u_l) - \varphi(W_l\,\bar{r}_{l-1})\big)\frac{1}{\varphi'}\,\bar{r}_{l-1}^{\mathrm{T}}\,, \tag{5.11}$$

for small prediction errors $u_l - W_l\,\bar{r}_{l-1}$. To recover the original predictive plasticity rule proposed in *Urbanczik and Senn* (2014), $\varphi(u_l)$ has to be sampled, representing backpropagating action potentials that penetrate deep into the basal tree (Section 2.1.3). This way, synapses projecting to the basal compartment can attain all the information required for plasticity updates[9].

5.2.2 Gradient-based neuron dynamics

What remains to be found are neuronal dynamics that are compatible with the derived plasticity rule by obeying the two constraints $\frac{\partial E}{\partial u_l} = 0$ and $E = 0$ for $\beta = 0$. Since, according to Eq. 5.10, energy minima correspond to learned patterns, a standard method of deriving neuronal dynamics is to assume that neural networks strive towards energy minima (*Hopfield*, 1982; *Rao and Ballard*, 1999; *Scellier and Bengio*, 2017; *Whittington and Bogacz*, 2017),

[8] This can be derived following *Scellier and Bengio* (2017): given the condition $\frac{\partial E}{\partial u_l} = 0$, the total derivatives of the energy with respect to the weights are the same as the partial derivatives, $\frac{\mathrm{d}E}{\mathrm{d}W_l} = \sum_l \frac{\partial E}{\partial u_l}\frac{du_l}{dW_l} + \frac{\partial E}{\partial W_l} = \frac{\partial E}{\partial W_l}$. The same is also true for β, i.e., $\mathrm{d}E/\mathrm{d}\beta = \partial E/\partial\beta$. Since the cost C can be extracted from the energy E by taking the partial derivative with respect to β, $C = \partial E/\partial\beta$, we can apply the derivative identities (for W_l and β) and interchange the total derivatives to arrive at $-\frac{\mathrm{d}C}{\mathrm{d}W_l} = -\frac{\mathrm{d}}{\mathrm{d}W_l}\frac{\partial E}{\partial\beta} = -\frac{\mathrm{d}}{\mathrm{d}\beta}\frac{\partial E}{\partial W_l} = \lim_{\beta\to 0}\frac{1}{\beta}(u_l - W_l\,\bar{r}_{l-1})\,\bar{r}_{l-1}^{\mathrm{T}}$. For the second last equality we exploited that for $\beta = 0$, the prediction error vanishes and hence $E = 0$ and $\frac{\partial E}{\partial W_l} = 0$.

[9] STDP requires such backpropagating action potentials as well, as it is purely driven by pre- and postsynaptic spike times.

effectively performing maximum likelihood estimates (MLE) if we interpret Eq. 5.9 as a negative log-likelihood function[10]. In this spirit, we can define neuronal dynamics as performing gradient descent on the energy function (Fig. 5.3A-C and Fig. 5.4D,E):

$$\tau\dot{u}_l = -\frac{\partial E}{\partial u_l} \iff \tau\dot{u}_l = -u_l + W_l\bar{r}_{l-1} + \bar{e}_l\,, \quad \text{for } l = 1\dots N\,, \tag{5.12}$$
$$\text{with} \quad \bar{e}_l = \bar{r}'_l \odot [W_{l+1}^{\mathrm{T}}(u_{l+1} - W_{l+1}\bar{r}_l)] \text{ for } l = 1\dots N-1 \text{ and } \bar{e}_N = \beta\big(u_N^{\text{target}} - u_N\big)\,,$$

resulting in leaky membrane dynamics as discussed for Eq. 2.2. Apart from the forward input $W_l\bar{r}_{l-1}$ discussed previously, an error term $\bar{e}_l$ enters the membrane dynamics, originating from "nudging" at the output area $\bar{e}_N$: to train the network, the output membrane potentials u_N are slightly pulled towards the target values u_N^{target}. How strongly the last area is pulled to the target is controlled by the nudging strength β. As in other models of backpropagation in the brain (*Scellier and Bengio*, 2017; *Whittington and Bogacz*, 2017; *Guerguiev, Lillicrap, and Richards*, 2017; *Sacramento, Costa, Bengio, and Senn*, 2018), the constraints set by the plasticity rule Eq. 5.10 are only satisfied if the network resides in a stationary state $\frac{\partial E}{\partial u_l} \propto \dot{u}_l = 0$. If this is the case, we can identify the error entering the neuronal dynamics as the prediction error from the energy function E

$$\bar{e}_l = u_l - W_l\bar{r}_{l-1}\,, \tag{5.13}$$

which then also enters the plasticity rule Eq. 5.10

$$\frac{1}{\eta}\dot{W}_l = \lim_{\beta\to 0}\frac{1}{\beta}\big(u_l - W_l\,\bar{r}_{l-1}\big)\,\bar{r}_{l-1}^{\mathrm{T}} = \lim_{\beta\to 0}\frac{1}{\beta}\bar{e}_l\,\bar{r}_{l-1}^{\mathrm{T}}\,, \tag{5.14}$$

and the error representation itself

$$\bar{e}_l = \bar{r}'_l \odot [W_{l+1}^{\mathrm{T}}(u_{l+1} - W_{l+1}\bar{r}_l)] = \bar{r}'_l \odot W_{l+1}^{\mathrm{T}}\bar{e}_{l+1}\,, \tag{5.15}$$

taking the iterative form of the error backpropagation algorithm (Eq. 5.7). Thus, as long as the neuronal dynamics are stationary, weak nudging[11] of the last area introduces an error signal which then propagates backward through the network via the membrane potentials. Without nudging, this output error is zero, and hence all areawise prediction errors vanish, leading to a vanishing energy function E as required by the plasticity rule. Therefore, as long as the network is stationary, the local plasticity rule Eq. 5.14 minimizes a local error signal, and by doing so performs gradient descent on a (global) cost function C from which the output error is derived[12].

[10] Different from the stochastic networks of the previous chapter that sampled the whole energy landscape.

[11] Weak nudging is required such that with and without nudging, the forward activity of the network is approximately equal, i.e., learning (or nudging) does not strongly disturb the inference pathway.

[12] Although errors are propagated through the transpose of the forward weights, which is non-local. This problem will be discussed in Section 5.2.5.

5.2.3 From gradients to lookahead dynamics

In the gradient-based model, learning is separated into two phases with plasticity either turned on or off. While a new and static input stimulus u_0^{input} is presented to the network, plasticity is turned off until neuronal dynamics reach a stationary state (Fig. 5.3A,B). If the output area is not nudged towards a target u_N^{target} (i.e., $\beta = 0$), the network settles in an energy minimum with zero value (Fig. 5.3A). Otherwise it settles in a non-zero energy minimum (Fig. 5.3B), as the output error propagates areawise through the network via feedback connections W^{T}, leading to non-zero contributions when summing up the squared prediction errors in the energy function Eq. 5.9. After convergence, plasticity can be turned on to decrease the energy back towards zero (Fig. 5.3C) and hence forcing the network to produce the output u_N^{target} when presenting the corresponding input u_0^{input} without nudging, whereas synaptic plasticity has to be much slower than neuronal dynamics in order to remain approximately in the steady state.

This partition into phases is indispensable for plasticity to reduce the cost function and has been used in previous energy-based approaches (*Pineda*, 1987; *Almeida*, 1987; *LeCun, Touresky, Hinton, and Sejnowski*, 1988; *Scellier and Bengio*, 2017; *Whittington and Bogacz*, 2017). It originates from the fact that, because of the introduced neuronal dynamics, the feedback from higher[13] areas will always be delayed due to somatic (leaky) integration. Hence, if the activity in a lower area causes an error in a higher area, and this error is fed back to the lower area, the correcting error signal will arrive too late (see Fig. 5.4E). In fact, the error signal will be mixed with bottom-up activity corresponding to new input patterns, thus being disturbed and rendered unusable for learning the presented patterns. This effect becomes worse when increasing the number of areas, leading to larger waiting times until the steady state is reached and plasticity can be turned on, thus making it impossible to learn input-output relations which are not presented in a static way.

In the following, we propose an extension of the gradient-based model which allows neuronal and synaptic dynamics to be on all the time, allowing plasticity to act at every moment in time when presented with a time-continuous input-output stream $(u_0^{\text{input}}(t), u_N^{\text{target}}(t))$, without any phases or a strong separation of time scales. This is achieved by replacing the firing rate $\bar{r}_{l-1}(t)$ and error $\bar{e}_l(t)$ with a lookahead (or advanced/prospective) version $\bar{r}_{l-1}(t+\tau)$ and $\bar{e}_l(t+\tau)$ thereof:

$$\tau \dot{u}_l(t) \approx -u_l(t) + W_l \bar{r}_{l-1}(t+\tau) + \bar{e}_l(t+\tau)\,. \tag{5.16}$$

In case of rates, the advanced version contains the information which rate (or membrane potential) the neuron will have in the future (i.e., τ ms later when the stimulus actually arrives), hence bridging the temporal delay introduced by somatic and dendritic filtering. The upper area will integrate this rate, and its instantaneous voltage $u_{l+1}(t)$ can now be seen as directly caused by $u_l(t)$ at the same time step t. Similarly, an advanced error in the upper area will be fed back and, when being integrated with the lower-area voltage $u_l(t)$, adequately correct the lower-area membrane potential in order to undo the upper-area error caused by $u_l(t)$. This way, we establish in Eq. 5.16 the correct causal structure such

[13] Higher in the sense of closer to the output area, and lower in the sense of closer to the input.

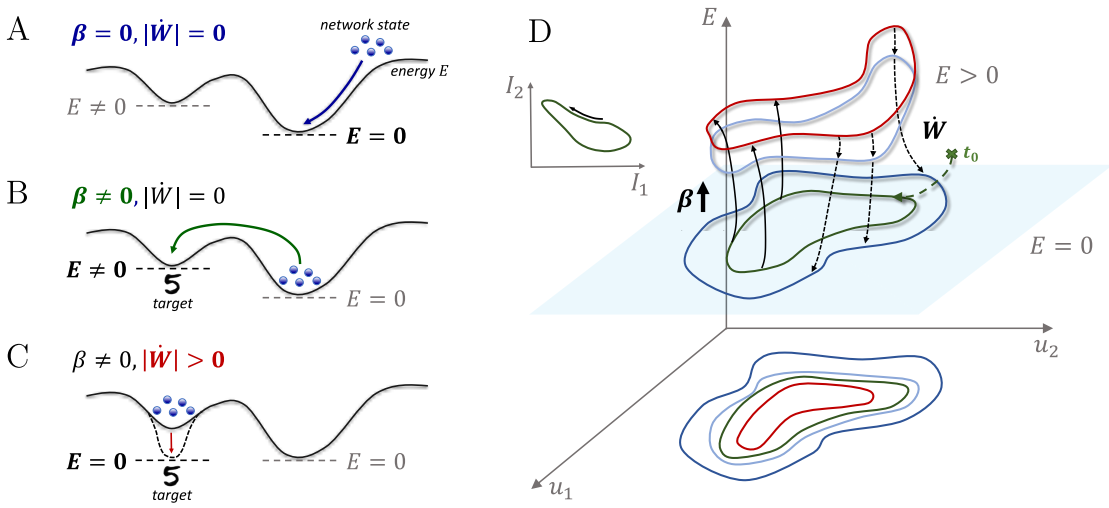

Figure 5.3.: Illustration of the network dynamics for energy-based models without (A-C) and with (D) lookaheads. **(A-C)** For purely gradient-based neuronal dynamics, the network always settles in a stationary state given by an energy minimum. Without teacher signal ($\beta = 0$), this minimum corresponds to a global minimum with zero energy (A). With teaching ($\beta \neq 0$), the energy minimum has a value $E \neq 0$ (B). After network dynamics are stationary, plasticity can be turned on to reduce the energy back to zero, effectively learning the input-output relation (C). **(D)** With lookaheads, the network moves along a zero-energy trajectory (green), here shown for two neurons with membrane potentials u_1 and u_2. After initialization (t_0), the two neurons decay exponentially fast towards the inputs I_1 and I_2 and follow them consistently. Nudging ($\beta \neq 0$) with an unlearned input-output stimulus pushes the network to non-zero energy regions (black arrows), while plasticity continuously pulls the trajectory back (dashed black arrows) to a zero-energy trajectory (blue). 2D projections of all trajectories are shown at the bottom.

that (locally) between areas, the temporal connection between inputs leading to an error and backpropagated errors is not lost due to filtering delays (Fig. 5.4D,E). Different from the gradient descent approach, the network now moves along a zero-energy trajectory in the absence of nudging. Whenever the output is nudged, the trajectory is pushed towards non-zero energy regions, but synaptic plasticity continuously pulls it back to the zero-energy plain, simultaneously minimizing the output cost (Fig. 5.3D).

Even though the advanced rates as introduced here are strongly coupled to the low-pass filtering mechanism, such advancements in the instantaneous response of cortical neurons have in fact been measured in response to sinusoidally modulated input currents superimposed with background noise, with a temporal advancement of 20 ms, i.e., on the same order of magnitude as the membrane time constant[14] (*Köndgen, Geisler, Fusi, Wang, Lüscher, and Giugliano* (2008), see also Fig. 5.4A). To advance its rate, a neuron only needs access to its own membrane potential u_l and time derivative $\dot{u}_l$ at time t. This can be seen when considering the first two Taylor expansion terms of the advanced rate (Fig. 5.4B)

$$\bar{r}_l(t+\tau) = \bar{r}_l(t) + \tau\,\dot{\bar{r}}_l(t) + \mathcal{O}(\tau^2) = \bar{r}_l(t) + \tau\,\bar{r}_l'(t)\,\dot{u}_l(t) + \mathcal{O}(\tau^2)\,, \tag{5.17}$$

[14] And in *Palmer, Marre, Berry, and Bialek* (2015), it is experimentally shown that *groups of cells in the retina carry information about the future state of their own activity*, further highlighting the importance of temporal predictions for neural responses.

where $\bar{r}_l'$ denotes the derivative of $\bar{r}_l$ with respect to u_l (of course for larger values of τ this is only an approximation). Such an advancement might, for instance, be realized by the mechanism to generate action potentials, allowing neurons to compensate (or even over-compensate) for delays introduced by leaky dynamics; as can be demonstrated for Hodgkin-Huxley-like activation mechanisms (Appendix B.1.1).

5.2.4 Deriving lookaheads via prospective least action

The idea of advanced responses as discussed in the previous section can be formalized by combining the least action principle (see, e.g., *Landau and Lifshitz* 1959) with a prospective coding scheme (*Brea, Gaál, Urbanczik, and Senn*, 2016). To do so, we introduce the "generalized neuronal coordinates" $\tilde{u}$, defined as the discounted future voltage

$$\tilde{u}(t) = \frac{1}{\tau}\int_t^\infty u(t')e^{-\frac{t'-t}{\tau}}\mathrm{d}t' \,. \tag{5.18}$$

The ordinary membrane potentials can be recovered from these predictive voltages by looking back in time, $u = \tilde{u} - \tau\dot{\tilde{u}}$ (Fig. 5.4C). By formulating the least action principle with respect to these generalized neuronal coordinates, the ordinary membrane potentials still have leaky dynamics, but rates and errors enter with lookaheads as described in the previous section. The least action principle requires that the trajectory $\left(\tilde{u}(t), \dot{\tilde{u}}(t)\right)$ leaves the action $A = \int E(\tilde{u}, \dot{\tilde{u}})\mathrm{d}t$ stationary, $\delta A = 0$. The solution for such an integral optimization is generally given by the Euler-Lagrange equations

$$\frac{\partial E}{\partial \tilde{u}_l} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial E}{\partial \dot{\tilde{u}}_l} = 0 \,, \tag{5.19}$$

which – in our case – can be reformulated to

$$\left(1 + \tau\frac{\mathrm{d}}{\mathrm{d}t}\right)\frac{\partial E}{\partial u_l} = 0 \,, \tag{5.20}$$

using[15] $\frac{\partial}{\partial \dot{\tilde{u}}_l} = -\tau\frac{\partial}{\partial u_l}$. $\mathcal{L} = (1 + \tau\frac{\mathrm{d}}{\mathrm{d}t})$ is the lookahead operator introduced in Eq. 5.17, i.e.,

$$\mathcal{L}(\bar{r}_l) = \bar{r}_l(t) \, + \tau\dot{\bar{r}}_l(t) \approx \bar{r}_l(t+\tau) \,. \tag{5.21}$$

Moreover, it is also the inverse operator of low-pass filtering

$$\mathcal{L}(\bar{x}) = x \,, \tag{5.22}$$

for time-dependent variables x with their low-pass filtering

$$\bar{x}(t) = \frac{1}{\tau}\int_{-\infty}^t x(t')e^{-\frac{t-t'}{\tau}}\mathrm{d}t' \,. \tag{5.23}$$

[15] This can be derived by using the identities $\frac{\partial E}{\partial \tilde{u}} = \frac{\partial E}{\partial(\tilde{u}-\tau\dot{\tilde{u}})}\frac{\partial(\tilde{u}-\tau\dot{\tilde{u}})}{\partial \tilde{u}} = -\frac{1}{\tau}\frac{\partial E}{\partial(\tilde{u}-\tau\dot{\tilde{u}})}\frac{\partial(\tilde{u}-\tau\dot{\tilde{u}})}{\partial \dot{\tilde{u}}} = -\frac{1}{\tau}\frac{\partial E}{\partial \dot{\tilde{u}}}$ and $\frac{\partial E}{\partial \tilde{u}} = \frac{\partial E}{\partial u}$.

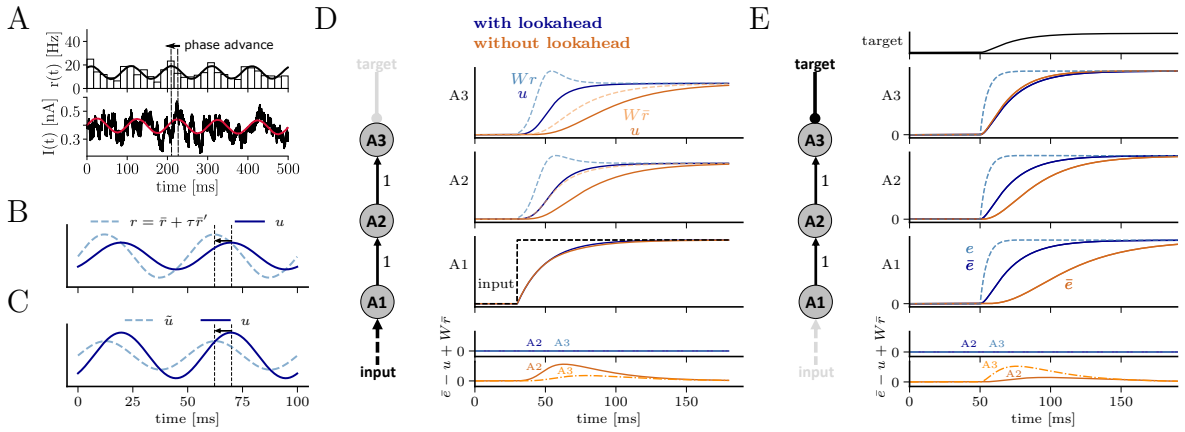

Figure 5.4.: **(A)** Instantaneous response $r(t)$ (population response averaged over trials) of pyramidal neurons (L5 pyramidal cells of the rat somatosensory cortex) to a sinusoidal input current with background noise. At low frequencies, the response is phase-advanced compared to the input. Figure adapted from *Köndgen et al.* (2008). **(B)** In the presented model, the instantaneous response is advanced with respect to the membrane potential of the neuron. **(C)** Illustration of the prospective rate $\tilde{u}$. **(D, E)** Illustration of the lookahead dynamics for the inference path (D) and error path (E) in a network consisting of three neurons (A1-A3) connected with weights of strength 1. **(D)** Given a step current as input, the membrane potential exponentially decays towards the input. For higher areas, the lookahead compensates for integration delays (blue, A1-A3), while without lookahead, the input signal gets further and further delayed (orange, A1-A3). Because of delays, the condition $\frac{\partial E}{\partial u} = \bar{e} - u + W\bar{r} = 0$ is not met during transients for the model without lookaheads (bottom). **(E)** Same as in (D), but for the propagated errors in response to target nudging (without input signal). Due to the lookahead (e), errors ($\bar{e}$) propagate undelayed through the network (blue, A3-A1), while being delayed without lookahead (orange, A3-A1). To summarize, by advancing their response, neurons are capable of perfectly tracking their input, allowing almost delay-less propagation of time-dependent signals. A similar fast tracking behavior was observed for populations of cortical neurons in response to broadband input signals (with transients much faster than the membrane time constant), see Figure 7 in *Köndgen et al.* (2008).

Inserting the predictive voltages in the energy function Eq. 5.9 and using the least action principle, we obtain the following neuronal dynamics:

$$\delta A = 0 \iff \left(1 + \tau \frac{\mathrm{d}}{\mathrm{d}t}\right) \frac{\partial E}{\partial u_l} = 0 \iff \tau \dot{u}_l = -u_l + W_l r_{l-1} + e_l \,, \tag{5.24}$$

with $r_l = \bar{r}_l + \tau \dot{\bar{r}}_l$, $e_l = \bar{e}_l + \tau \dot{\bar{e}}_l$ and[16] $\bar{e}_l = \bar{r}'_l \odot [W_{l+1}^{\mathrm{T}}(u_{l+1} - W_{l+1}\bar{r}_l)]$. We assume small learning rates η, such that terms including $\dot{W}_l$ can be neglected[17].

Several properties of these dynamics have to be discussed here: first, the rates and errors entering the equation of motion are advanced, i.e., $r \approx \bar{r}(t+\tau)$ and $e \approx \bar{e}(t+\tau)$. Second, the membrane potential of each neuron leaky integrates its input, as expected from neuronal membrane dynamics (Fig. 5.4D). However, due to lookaheads canceling low-pass filtering,

[16] The error can be written explicitly as $e_l = \bar{r}'_l \odot [W_{l+1}^{\mathrm{T}}(u_{l+1} + \tau \dot{u}_{l+1} - W_{l+1} r_l)] + \tau \dot{\bar{r}}'_l \odot [W_{l+1}^{\mathrm{T}}(u_{l+1} - W_{l+1}\bar{r}_l)]$.

[17] Slow learning is required anyway to avoid catastrophic forgetting (overwriting weights such that previously learned stimuli are unlearnt again, see, e.g., *McCloskey and Cohen* 1989).

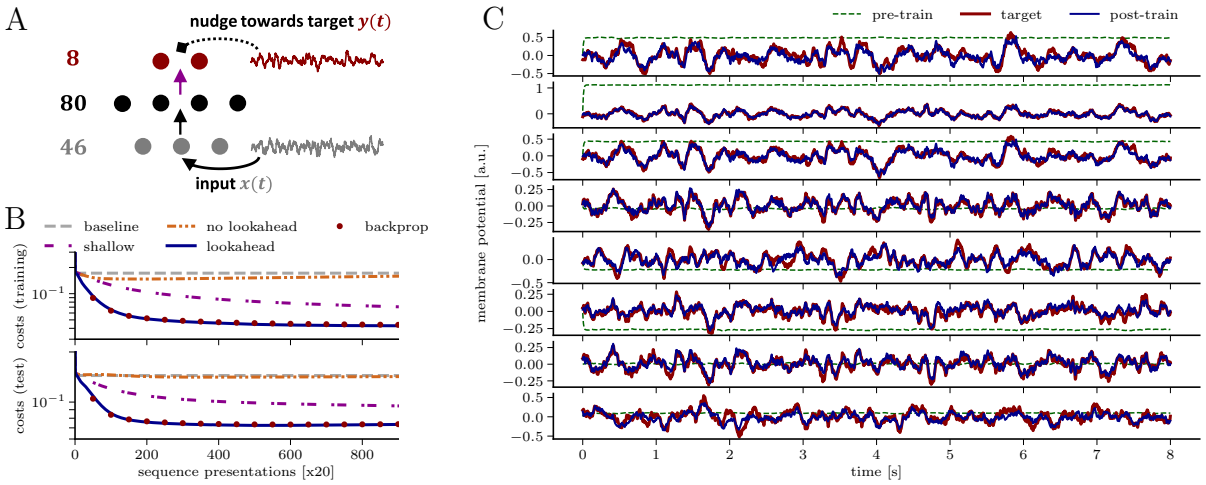

Figure 5.5.: Real-time regression learning of human intracranial electroencephalography (iEEG) data, provided by Kaspar Schindler (University Hospital Bern, *Burrello, Schindler, Benini, and Rahimi* 2018). Different from static data sets like MNIST, the iEEG data is time-continuous and consequently used here to investigate the real-time learning capability of the presented framework. **(A)** For the learning task, 54 electrode signals of cortical local field potentials are separated into 46 input and 8 target output signals. **(B)** Performance of different models, repeatedly trained on an 8s excerpt of the iEEG data (testing is done on a different 8s excerpt). For baseline performance, the mean value of each target iEEG trace is used as a predictor (gray). Learning with the presented lookahead model (blue) performs as well as error backpropagation (red), while learning without lookaheads (orange) is stuck at baseline performance. Utilizing error backpropagation significantly speeds up the training, as seen when freezing the visible-to-hidden weights, only training the output weights (magenta in A,B). **(C)** Generated output given the test data before and after training.

no delays as discussed in the gradient-based model occur (Fig. 5.4D,E). Furthermore, the equation of motion $\left(1+\tau\frac{\mathrm{d}}{\mathrm{d}t}\right)\frac{\partial E}{\partial u_l}$ pulls $\frac{\partial E}{\partial u_l}$ exponentially fast to zero (Fig. 5.4D,E, bottom), guaranteeing that, far away from initialization, $\frac{\partial E}{\partial u_l} = 0$ for all areas, as required by the plasticity rule (Eq. 5.10)[18].

In this framework, error backpropagation can be reobtained by low-pass filtering the equation of motion, yielding $\overline{\tau\dot{u}_l + u_l} = u_l = W_l\bar{r}_{l-1}+\bar{e}_l$. Similar to how we did in the gradient-based model (without lookahead), this can again be inserted into the error representation and plasticity rule, resulting in Eqs. 5.14 and 5.15. Therefore, the lookahead dynamics introduced by the least action principle enable a real-time implementation of error backpropagation while being implementable by, e.g., biological or neuromorphic neurons. Both rates and errors propagate simultaneously through the network and plasticity minimizes these local errors to reduce the cost function continuously in time. Even though lookaheads only act locally between areas, all areas are phase-locked and information spreads, independent of network depth (the number of areas), from input to output area with times on the order of the membrane time constant τ, allowing fast neuronal processing (*Thorpe, Fize, and Marlot*,

[18] A more involved proof with a slightly different perspective on learning in the neuronal least action framework can be found in Appendix B.1.2.

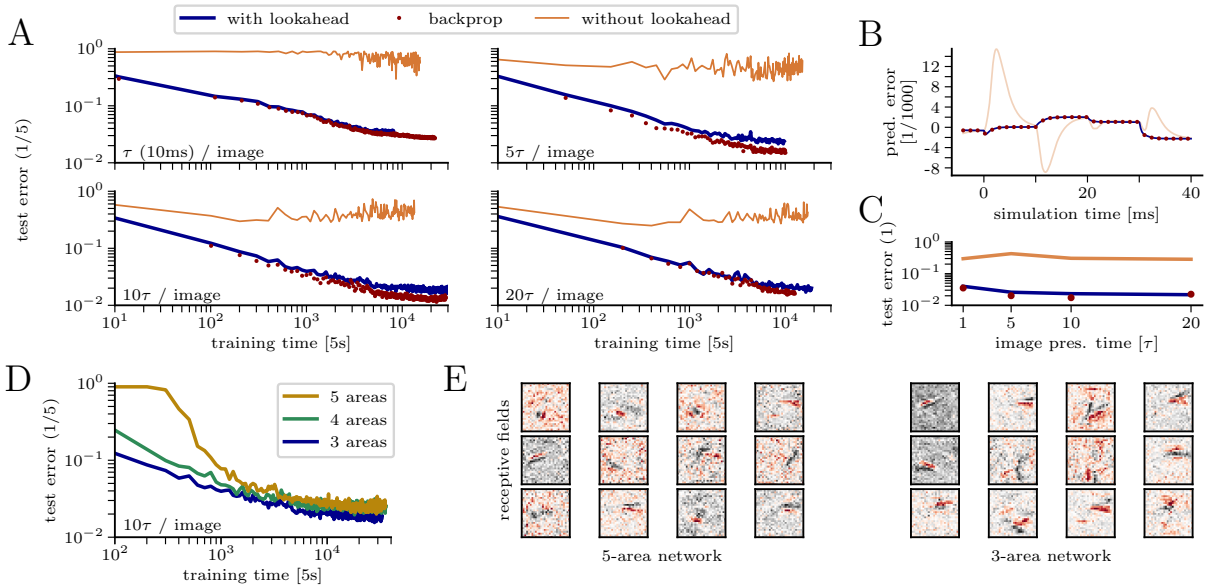

Figure 5.6.: Learning to classify handwritten digits from a continuous stream of MNIST images. **(A)** The error rate is tested during training (on 1/5 of the test data) for different models (colors) as well as different presentation times per image (panels). Even for presentation times on the order of the membrane time constant, the model with lookahead still performs well (top left panel). For all experiments, a network with $784 - 500 - 10$ neurons was used. **(B)** Prediction error used in the plasticity rule during time-continuous presentation of several images. For the model without lookaheads (orange), strong transients in the error signal arise when switching patterns, disturbing learning. With lookaheads (blue), the model is able to correctly follow the error flow of backprop (red). **(C)** Final classification performance on the whole MNIST test data set for the experiments shown in (A). The presented model reaches error rates of **3.98%** (τ), **2.59%** (5τ), **2.33%** (10τ), **2.16%** (20τ), comparable to what can be achieved when replacing synaptic dynamics with real error backpropagation, **3.52%** (τ), **2.01%** (5τ), **1.74%** (10τ), **2.24%** (20τ). **(D)** Learning of MNIST with lookahead dynamics for deeper networks with $784-300-200-10$ (4 areas) and $784-200-200-100-10$ (5 areas) neurons. The 3-area network performance is taken from (A). **(E)** Exemplary receptive fields of the 5 and 3-area network in (D), after training finished. As expected, the first area learned to recognize edges and strokes, characteristic components of handwritten digits. The shown experiments are single shots. Additional results can be found in Fig. B.2.

1996)[19]. We demonstrate real-time learning matching error backpropagation for a simple, time-continuous regression task (Fig. 5.5) as well as for learning to classify MNIST digits from a time-continuous input-output stream (Fig. 5.6). The model can also be extended to recurrent networks, implementing truncated[20] error backpropagation (see Appendix B.2.2).

[19]However, we are not considering axonal delays ($\mathcal{O}(1)$ms or less, *Wang, Shultz, Burish, Harrison, Hof, Towns, Wagers, and Wyatt* 2008), which render almost instantaneous information transmission through an infinitely deep network impossible in practice.

[20]Truncated in time.

5.2.5 Dendritic microcircuits for local error learning

As described in Section 5.2.1, the dynamics derived from the prospective least action principle can be realized by pyramidal neurons with basal and apical dendrites. The basal dendrites integrate feedforward input, whereas the soma integrates input coming from both the basal and apical compartment. What remains to be shown is how the neuron calculates, stores and accesses the prediction error $\bar{e}_l$.

Similar to *Sacramento, Costa, Bengio, and Senn* (2018), we propose that local prediction errors are stored in the apical dendrites. This way, both forward input and errors are stored separately and simultaneously in the neuron, allowing it to recover the prediction error for synapses projecting to the basal compartment via a predictive learning rule (*Urbanczik and Senn*, 2014). Errors are calculated by taking the difference of (i) feedback from a higher cortical area projecting to the apical compartment and (ii) a bottom-up prediction that tries to explain away the feedback. The bottom-up prediction is mediated via a stereotypical interneuron microcircuit, projecting from a cortical area to a pool of interneurons that project back to the apical dendrites of the pyramidal neurons of this area[21].

We can again use the least action principle to derive dynamics and plasticity for interneurons and apical dendrites from separate energy functions, ensuring time-continuous neurosynaptic dynamics:

1. The neural code of the interneurons is given by

$$E^{\mathrm{I}} = \sum_{l=1}^{n} \frac{1-\beta^{\mathrm{I}}}{2} \|u_l^{\mathrm{I}} - W_l^{\mathrm{IP}} \bar{r}_l^{\mathrm{P}}\|^2 + \frac{\beta^{\mathrm{I}}}{2} \|B_l^{\mathrm{IP}} u_{l+1}^{\mathrm{P}} - u_l^{\mathrm{I}}\|^2 , \tag{5.25}$$

   where u^{I} are the interneuron membrane potentials, W_l^{IP} the weights projecting from the pyramidal neurons of area l to the interneurons of the same area, B_l^{IP} fixed and random weights for top-down nudging (from area $l+1$ to l) and β^{I} the corresponding nudging strength[22]. To better discriminate pyramidal and interneuron membrane potentials, we further add an index for pyramidal neurons, $u_l = u_l^{\mathrm{P}}$. Introducing the predictive interneuron voltage $u^{\mathrm{I}} = \tilde{u}^{\mathrm{I}} - \tau \dot{\tilde{u}}^{\mathrm{I}}$, interneuron dynamics are obtained by requiring a stationary action $A^{\mathrm{I}} = \int E^{\mathrm{I}}(\tilde{u}^{\mathrm{I}}, \dot{\tilde{u}}^{I}) \mathrm{d}t$:

$$\delta A^{\mathrm{I}} = 0 \iff \left(1 + \tau \frac{\mathrm{d}}{\mathrm{d}t}\right) \frac{\partial E^{\mathrm{I}}}{\partial u_l^{\mathrm{I}}} = 0 \iff \tau \dot{u}_l^{\mathrm{I}} = W_l^{\mathrm{IP}} r_l^{\mathrm{P}} - u_l^{\mathrm{I}} + \frac{\beta^{\mathrm{I}}}{1-\beta^{\mathrm{I}}} e_l^{\mathrm{I}} , \tag{5.26}$$

   with top-down error $\bar{e}_l^{\mathrm{I}} = B_l^{\mathrm{IP}} u_{l+1}^{\mathrm{P}} - u_l^{\mathrm{I}}$ and lookaheads $r_l^{\mathrm{P}} = \bar{r}_l^{\mathrm{P}} + \tau \dot{\bar{r}}_l^{\mathrm{P}}$ and $e_l^{\mathrm{I}} = \bar{e}_l^{\mathrm{I}} + \tau \dot{\bar{e}}_l^{\mathrm{I}}$. Low-pass filtering this equation yields

$$u_l^{\mathrm{I}} = W_l^{\mathrm{IP}} \bar{r}_l^{\mathrm{P}} - u_l^{\mathrm{I}} + \frac{\beta^{\mathrm{I}}}{1-\beta^{\mathrm{I}}} \bar{e}_l^{\mathrm{I}} , \tag{5.27}$$

[21] In the cortex, pyramidal neurons are mostly excitatory and interneurons inhibitory (*Markram, Toledo-Rodriguez, Wang, Gupta, Silberberg, and Wu*, 2004). For simplicity, and to better compare with deep learning methods, we neglect this restriction here. However, by virtue of a more elaborate circuitry, the theory might be reorganized to suffice this restriction (e.g., by replacing an inhibitory connection from a pyramidal neuron by an additional inhibitory interneuron).

[22] Top-down nudging acts as a teacher signal for the interneurons. Hence, they learn to approximate a linear transformation of the activity in the higher cortical area.

and by writing out the top-down error, we further get

$$u_l^{\mathrm{I}} = (1-\beta^{\mathrm{I}})W_l^{\mathrm{IP}}\bar{r}_l^{\mathrm{P}} + \beta^{\mathrm{I}}B_l^{\mathrm{IP}}u_{l+1}^{\mathrm{P}}\,. \tag{5.28}$$

Thus, the interneuron membrane potential is a convex combination of the lateral (dendritic) input $W_l^{\mathrm{IP}}\bar{r}_l^{\mathrm{P}}$ and top-down (somatic) nudging $B_l^{\mathrm{IP}}u_{l+1}^{\mathrm{P}}$. The plasticity rule obtained by minimizing the energy function

$$\dot{W}_l^{\mathrm{IP}} \propto -\frac{\partial E^{\mathrm{I}}}{\partial W_l^{\mathrm{IP}}} \propto (u_l^{\mathrm{I}} - W_l^{\mathrm{IP}}\bar{r}_l^{\mathrm{P}})\,(\bar{r}_l^{\mathrm{P}})^{\mathrm{T}}\,, \tag{5.29}$$

changes the lateral weights W_l^{IP} such that the lateral input predicts the upper area activity, i.e., using $\dot{W}_l^{\mathrm{IP}} = 0$ and Eq. 5.28 gives $u_l^{\mathrm{I}} = W_l^{\mathrm{IP}}\bar{r}_l^{\mathrm{P}} = B_l^{\mathrm{IP}}u_{l+1}^{\mathrm{P}}$. The learning rule is again given by the dendritic prediction of somatic activity (*Urbanczik and Senn*, 2014). Top-down weights B_l^{IP} are randomly initialized and remain constant. To allow a loss-less representation of the upper area activity, the number of interneurons has to be at least as numerous as pyramidal neurons in the upper cortical area, but can, for instance, also be larger.

2. We propose that apical dendrites encode a local prediction error used for learning of the forward weights W_l. This error is formed by subtracting top-down feedback from the upper cortical area and input from lateral interneurons (as introduced above)[23],

$$\bar{e}_l^{\mathrm{P}} = \bar{r}_l'^{\mathrm{P}} \odot \bar{u}_l^{\mathrm{d},\infty} \quad \text{with} \quad \bar{u}_l^{\mathrm{d},\infty} = B_l^{\mathrm{PP}}u_{l+1}^{\mathrm{P}} - W_l^{\mathrm{PI}}u_l^{\mathrm{I}}\,, \tag{5.30}$$

where B_l^{PP} are the top-down weights from area $l+1$ to area l and W_l^{PI} the lateral weights from interneurons of area l to the apical compartments of pyramidal neurons of area l. Multiplication with the derivative of the postsynaptic rate could be seen as an inactivation of the apical signal through backpropagating action potentials (from soma to apical) at saturating frequencies (*Larkum, Senn, and Lüscher*, 2004). Similar to feedback alignment (*Lillicrap, Cownden, Tweed, and Akerman*, 2016), we choose the feedback weights B_l^{PP} to be randomly initialized and constant.

The original error representation $\bar{e}_l$ (Eq. 5.15) can be recovered by choosing $B_l^{\mathrm{PP}} = W_l^{\mathrm{PI}} = W_{l+1}^{\mathrm{T}}$, $W_l^{\mathrm{IP}} = W_{l+1}$ and $r_l^{\mathrm{I}} = W_l^{\mathrm{IP}}\,\bar{r}_l$ (which is, e.g., learned by the interneurons by setting $B_l^{\mathrm{IP}} = \mathbb{1}$). The energy function for the apical voltages u^{d} is given by

$$E^{\mathrm{d}} = \sum_{l=1}^{n} \frac{1-\beta^{\mathrm{d}}}{2}\|u_l^{\mathrm{d}} - \bar{u}_l^{\mathrm{d},\infty}\|^2 + \frac{\beta^{\mathrm{d}}}{2}\|u_{\mathrm{rest}} - u_l^{\mathrm{d}}\|^2\,. \tag{5.31}$$

[23] Both feedback and lateral information is given via membrane potentials instead of rates. This can be achieved by assuming (i) that interneurons basically operate in a linear activation regime (e.g., like fast spiking neurons, *La Camera, Rauch, Thurbon, Luscher, Senn, and Fusi* 2006) where rate and membrane potential are proportional to each other and (ii) that short-term plasticity for the top-down connections recovers an approximation of the upper-area membrane potential from the presynaptic rate, as proposed for spiking neurons in *Pfister, Dayan, and Lengyel* (2010).

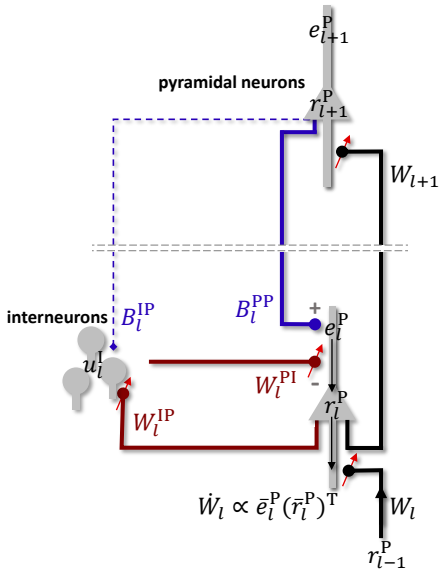

Figure 5.7.: Microcircuit of interneurons and pyramidal neurons used to learn the local error representation. Red arrows mark plastic synapses, all other synapses are static and initialized randomly. Lateral connections from pyramidal to interneuron and back (red) learn to explain away top-down feedback (blue). In order to train the pyramidal-to-interneuron weights W^{IP}_l, the interneurons also receive weak top-down nudging (blue, dashed). The original error representation for pyramidal neurons is recovered when setting $W^{\mathrm{IP}}_l = W_{l+1}$, $W^{\mathrm{PI}}_l = B^{\mathrm{PP}}_l = W^{\mathrm{T}}_{l+1}$ and $B^{\mathrm{IP}}_l = \mathbb{1}$.

and dynamics are derived by assuming a stationary action $A^{\mathrm{d}} = \int E^{\mathrm{d}}(\tilde{u}^{\mathrm{d}}, \dot{\tilde{u}}^{\mathrm{d}})\,\mathrm{d}t$:

$$\delta A^{\mathrm{d}} = 0 \iff \left(1 + \tau \frac{\mathrm{d}}{\mathrm{d}t}\right) \frac{\partial E^{\mathrm{d}}}{\partial u^{\mathrm{d}}_l} = 0 \iff \tau \dot{u}^{\mathrm{d}}_l = -u^{\mathrm{d}}_l + (1-\beta^{\mathrm{d}})\, u^{\mathrm{d},\infty}_l + \beta^{\mathrm{d}} u_{\mathrm{rest}} \,. \quad (5.32)$$

Low-pass filtering the equation of motion, we can identify $u^{\mathrm{d}}_l = (1-\beta^{\mathrm{d}}) u^{\mathrm{d},\infty}_l$. The lookahead of the dendritic input $u^{\mathrm{d},\infty}_l$ can be realized by voltage-dependent conductances in the apical dendrite, similar to how sodium conductances may give rise to advanced rates (Appendix B.1.1). By assuming energy-minimizing plasticity again,

$$\dot{W}^{\mathrm{PI}}_l \propto -\frac{\partial E^{\mathrm{d}}}{\partial W^{\mathrm{PI}}_l} \propto -(u^{\mathrm{d}}_l - \bar{u}^{\mathrm{d},\infty}_l)\,(r^{\mathrm{I}}_l)^{\mathrm{T}} = \beta^{\mathrm{d}}\,(B^{\mathrm{PP}}_l u^{\mathrm{P}}_{l+1} - W^{\mathrm{PI}}_l u^{\mathrm{I}}_l)\,(u^{\mathrm{I}}_l)^{\mathrm{T}}\,, \quad (5.33)$$

the constructed interneuron microcircuit learns to predict away top-down feedback in the absence of output nudging, i.e., the apical potential stays at baseline u_{rest} (for simplicity, we assume $u_{\mathrm{rest}} = 0$) independent of the input, representing vanishing errors $\bar{e}_l = 0$. The plasticity rule takes the Hebbian form "postsynaptic times presynaptic activity", i.e., $\dot{W}^{\mathrm{PI}}_l \propto \bar{u}^{\mathrm{d},\infty}_l\,(u^{\mathrm{I}}_l)^{\mathrm{T}}$, driving the apical voltage to a homeostatic value u_{rest}, as previously proposed for cortical inhibitory plasticity (*Vogels, Sprekeler, Zenke, Clopath, and Gerstner*, 2011).

The complete microcircuit used for error coding is illustrated in Fig. 5.7. If the microcircuit is aligned accordingly, in the absence of output nudging ($\beta = 0$), no output error is induced and hence, all errors throughout the network should vanish. Therefore, any remaining errors are due to a misalignment of the forward prediction pathway $W^{\mathrm{PI}}_l W^{\mathrm{IP}}_l$ (mediated via interneurons) and the feedback pathway $B^{\mathrm{PP}}_l W_{l+1}$ (feedforward excitation followed by a recurrent feedback connection). In the following, we show that the plasticity rules derived for the microcircuit reduce such errors back to zero. Such a network state is called "self-predictive state", as the lateral microcircuit is able to correctly predict the self-generated network activity, resulting in vanishing errors without output nudging.

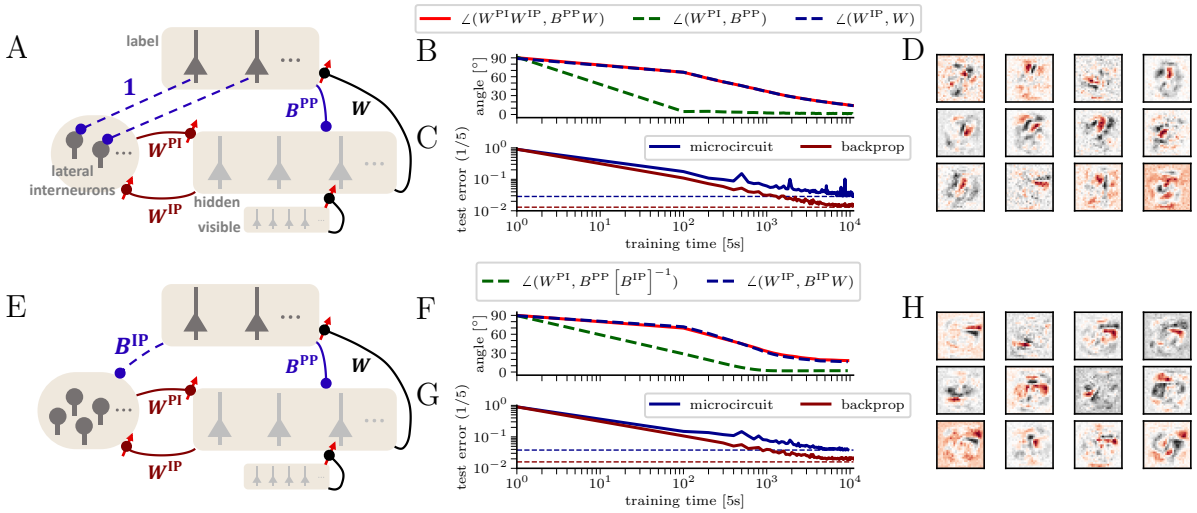

Figure 5.8.: Learning the lateral interneuron connections. **(A)** Similar to *Sacramento et al.* (2018), the lateral connections (W^{PI}, W^{IP}) can be learned. The network contains $784-500-10$ neurons. The feedback connections B^{PP} are randomly initialized and stay constant. Furthermore, the interneurons are nudged one-to-one by the pyramidal neurons of the higher cortical area. This way, during training the interneurons learn to approximate the higher-level pyramidal activity, while the lateral interneuron-to-pyramidal connections W^{PI} learn to compensate the top-down feedback mediated by B^{PP}. Plastic connections are marked by red arrows. **(B)** During training, the lateral as well as forward weights align accordingly to reach a self-predictive state (angles are calculated via the Frobenius inner product). **(C)** Learning MNIST (with a presentation time of 100ms per image) while training the interneuron circuit at the same time. **(D)** Similar receptive fields as in Fig. 5.6 form during training. **(E–H)** Same as in (A–D), however the neurons are not nudged one-to-one but receive mixed nudging (via random and static matrix B^{IP}) from the higher cortical area, in a network with $784-300-10$ neurons. This way, the number of interneurons is not tied to the number of upper-area pyramidal neurons. For instance, here we show how both the lateral weights and the forward weights are learned simultaneously with 20 interneurons encoding the error for the visible-to-hidden connections. The presented results might be further improved by reducing both learning rates as well as the integration time step of the simulations.

The plasticity rule for $\dot{W}_l^{\mathrm{IP}}$ converges towards the fixed point $u_l^{\mathrm{I}} = W_l^{\mathrm{IP}}\bar{r}_l^{\mathrm{P}} = B_l^{\mathrm{IP}}u_{l+1}^{\mathrm{P}}$ and the plasticity rule for $\dot{W}_l^{\mathrm{PI}}$ has the stationary solution $W_l^{\mathrm{PI}}\,u_l^{\mathrm{I}} = B_l^{\mathrm{PP}}\,u_{l+1}^{\mathrm{P}}$, leading to vanishing errors $\bar{e}_l^{\mathrm{P}}$ (Eq. 5.30). In this case, the neurons in the network follow their input $u_{l+1}^{\mathrm{P}} = W_{l+1}\bar{r}_l^{\mathrm{P}}$. Therefore, if both plasticity rules are stationary, we have $B_l^{\mathrm{PP}}\,u_{l+1}^{\mathrm{P}} = B_l^{\mathrm{PP}}\,W_{l+1}\bar{r}_l^{\mathrm{P}}$ and $B_l^{\mathrm{PP}}\,W_{l+1}\bar{r}_l^{\mathrm{P}} = W_l^{\mathrm{PI}}\,u_l^{\mathrm{I}} = W_l^{\mathrm{PI}}\,B_l^{\mathrm{IP}}\,W_{l+1}\bar{r}_l^{\mathrm{P}}$, leading to

$$W_l^{\mathrm{IP}} = B_l^{\mathrm{IP}}\,W_{l+1} \quad \text{and} \quad W_l^{\mathrm{PI}} = B_l^{\mathrm{PP}}\left[B_l^{\mathrm{IP}}\right]^{-1}, \tag{5.34}$$

such that $W_l^{\mathrm{PI}}W_l^{\mathrm{IP}} = B_l^{\mathrm{PP}}W_{l+1}$. Hence, the pyramidal-to-interneuron weights learn to track the forward weights (even when these change[24]), such that the interneurons learn to reproduce a linearly mixed representation of the higher area pyramidal activities. The interneuron-to-

[24]Depending on the interneuron nudging strength β^{I} and the learning rates of the microcircuit, the tracking will not necessarily be perfect. The simulation results in Figs. 5.8 and 5.9 further show that perfect tracking is not needed for training of the forward weights (non-zero final angles).

pyramidal connections learn to correctly match the higher area prediction of the interneurons with the actual feedback. In the self-predictive state, the error backpropagation formula takes the following form

$$\begin{aligned}\bar{e}_l^{\mathrm{P}} &= \bar{r}_l'^{\mathrm{P}} \odot (B_l^{\mathrm{PP}}\, u_{l+1}^{\mathrm{P}} - W_l^{\mathrm{PI}}\, u_l^{\mathrm{I}}) \\ &= \bar{r}_l'^{\mathrm{P}} \odot (B_l^{\mathrm{PP}}\, u_{l+1}^{\mathrm{P}} - W_l^{\mathrm{PI}}\, W_l^{\mathrm{IP}} \bar{r}_l^{\mathrm{P}}) \\ &= \bar{r}_l'^{\mathrm{P}} \odot (B_l^{\mathrm{PP}}\, u_{l+1}^{\mathrm{P}} - B_l^{\mathrm{PP}}\, W_{l+1} \bar{r}_l^{\mathrm{P}}) = \bar{r}_l'^{\mathrm{P}} \odot B_l^{\mathrm{PP}}\, \bar{e}_{l+1}^{\mathrm{P}}\,, \end{aligned} \tag{5.35}$$

where W_{l+1}^{T} in the original formula gets replaced by the feedback matrix B_l^{PP}.

There is one reservation to be made in this analysis: the identities in Eq. 5.34 are learned for activities $\bar{r}_l^{\mathrm{P}}$ produced in the absence of nudging, while for error backpropagation we use it in the presence of nudging. Here, we need to postulate that the activities $\bar{r}_l^{\mathrm{P}}$, when generated by the input alone, cover the full space of activities arising later in the presence of output nudging. Then, after the inhibitory circuit learned to correctly cancel the purely

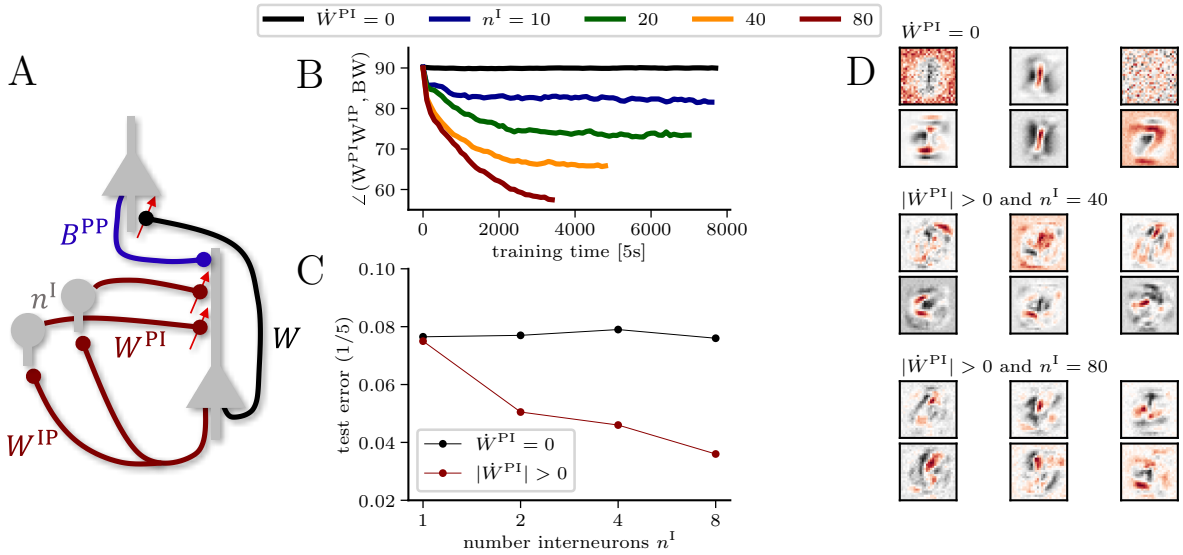

Figure 5.9.: Real-time learning of forward weights W and lateral interneuron-to-pyramidal weights W^{PI} from a randomly initialized configuration (we drop the area indices here for convenience). During training, both plasticity rules are always active and act simultaneously, without any pre-training phases. **(A)** Different from the experiment in Fig. 5.8, the interneurons receive no top-down nudging and only the lateral weights W^{PI} and forward weights W are learned (red arrows). All remaining weights are randomly initialized and stay constant. **(B)** Angle between the bottom-up pathway $W^{\mathrm{PI}}W^{\mathrm{IP}}$ and the top-down pathway $B^{\mathrm{PP}}W$. For large enough pools of interneurons, the weights align accordingly to allow training of forward weights in lower areas. Note that in this case, W^{PI} essentially learns to undo the linear mixing of B^{PP} while it tracks the evolution of W. **(C)** Best error rates achieved for a 3 area network ($784 - 500 - 10$) trained on MNIST, for different pool sizes n^{I}. **(D)** Receptive fields of the hidden neurons for several cases shown in (B) and (C). In future work, the interneurons might form recurrent populations, either as a liquid (*Natschläger et al.*, 2002) or also trained to improve learning of the local prediction error. This might connect to synthetic gradients (*Jaderberg et al.*, 2017), where small subnetworks of neurons also learn to approximate the backpropagated errors.

self-produced top-down input ($B_l^{\mathrm{PP}} u_{l+1}^{\mathrm{P}}$), only an additional drive of the upper-area neurons ($\bar{e}_{l+1}^{\mathrm{P}}$) will cause an error $\bar{e}_l^{\mathrm{P}}$ in the lower-area neurons.

In simulations, we find that both the lateral connections as well as the forward connections can be trained simultaneously, without the need for a prelearning phase as described above (see Fig. 5.8). In this case, Eq. 5.35 holds approximately at any moment in time after a short initial training phase without the need to relax towards the steady state of the dynamics. Thus, $\bar{e}_l^{\mathrm{P}}$ permanently drives plasticity of the hidden weights, $\dot{W}_l = \eta\, \bar{e}_l^{\mathrm{P}} \left[\bar{r}_{l-1}^{\mathrm{P}}\right]^{\mathrm{T}}$, to minimize the output costs, while W_l^{PI} and W_l^{IP} continuously track the changes in the forward weights to keep the network in a self-predictive state. This is demonstrated in Fig. 5.8 for the special case where B_l^{IP} is the identity matrix and the general case where B_l^{IP} is a random matrix. To measure similarity between matrices, we calculate the angle between them based on the Frobenius inner product $\langle A, B\rangle_{\mathrm{F}} = \sum_i \sum_j A_{ij} B_{ij}$, i.e., $\angle(A, B) = \arccos\left(\langle A, B\rangle_{\mathrm{F}} \,/\, \sqrt{\langle A, A\rangle_{\mathrm{F}} \cdot \langle B, B\rangle_{\mathrm{F}}}\right)$. For instance, two randomly generated matrices have an angle of 90°, while two matrices A, B that only differ by a multiplicative, scalar factor k, i.e., $A = k \cdot B$, have an angle of 0°.

If the number of interneurons is large enough, we find that the interneuron-to-pyramidal weights with the plasticity rule given in Eq. 5.33 are sufficient to reach and sustain a self-predictive state, i.e., $W_l^{\mathrm{PI}} = B_l^{\mathrm{PP}} W_{l+1} \left[W_l^{\mathrm{IP}}\right]^{-1}$. In this case, the interneurons receive no top-down nudging and the pyramidal-to-interneuron weights are initialized randomly and stay constant, see Fig. 5.9. The feedback weights B_l^{PP} may be plastic as well, trained to reproduce lower area activities and perform pattern completion, as demonstrated in *Sacramento, Costa, Bengio, and Senn* (2018).

5.2.6 Summary: neuronal least action principle

For convenience, we summarize the results of the previous sections here. For top-down derivation of the dynamics of (i) pyramidal neurons (u^{P}), (ii) interneurons (u^{I}) and (iii) pyramidal apical (dendritic) voltages (u^{d}), we propose the following energy functions (Eqs. 5.9, 5.25 and 5.31)

$$
\begin{aligned}
E^{\mathrm{P}} &= \sum_{l=1}^{N} \frac{1}{2} \|u_l^{\mathrm{P}} - W_l \bar{r}_{l-1}^{\mathrm{P}}\|^2 + \frac{\beta}{2} \|u_N^{\mathrm{target}} - u_N^{\mathrm{P}}\|^2 \,, \\
E^{\mathrm{I}} &= \sum_{l=1}^{n} \frac{1-\beta^{\mathrm{I}}}{2} \|u_l^{\mathrm{I}} - W_l^{\mathrm{IP}} \bar{r}_l^{\mathrm{P}}\|^2 + \frac{\beta^{\mathrm{I}}}{2} \|B_l^{\mathrm{IP}} u_{l+1}^{\mathrm{P}} - u_l^{\mathrm{I}}\|^2 \,, \\
E^{\mathrm{d}} &= \sum_{l=1}^{n} \frac{1-\beta^{\mathrm{d}}}{2} \|u_l^{\mathrm{d}} - B_l^{\mathrm{PP}} u_{l+1}^{\mathrm{P}} + W_l^{\mathrm{PI}} u_l^{\mathrm{I}}\|^2 + \frac{\beta^{\mathrm{d}}}{2} \|u_{\mathrm{rest}} - u_l^{\mathrm{d}}\|^2 \,,
\end{aligned}
$$

with their corresponding actions being defined as $A^x = \int E^x(\tilde{u}^x, \dot{\tilde{u}}^x) \mathrm{d}t$ using generalized neuronal coordinates (predictive voltages) $u^x = \tilde{u}^x - \tau \dot{\tilde{u}}^x$. From these energy functions,

dynamics are derived via the least action principle (Eqs. 5.24, 5.26 and 5.32)

$$\begin{aligned}
\delta A^{\mathrm{P}} = 0 &\iff \tau \dot{u}_l^{\mathrm{P}} = -u_l^{\mathrm{P}} + W_l r_{l-1}^{\mathrm{P}} + e_l^{\mathrm{P}} \,, \\
\delta A^{\mathrm{I}} = 0 &\iff \tau \dot{u}_l^{\mathrm{I}} = W_l^{\mathrm{IP}} r_l^{\mathrm{P}} - u_l^{\mathrm{I}} + \frac{\beta^{\mathrm{I}}}{1-\beta^{\mathrm{I}}} e_l^{\mathrm{I}} \,, \\
\delta A^{\mathrm{d}} = 0 &\iff \tau \dot{u}_l^{\mathrm{d}} = -u_l^{\mathrm{d}} + (1-\beta^{\mathrm{d}})\, u_l^{\mathrm{d},\infty} + \beta^{\mathrm{d}} u_{\mathrm{rest}} \,,
\end{aligned}$$

with lookaheads $x = \bar{x} + \tau \dot{\bar{x}}$, $\bar{e}_l^{\mathrm{P}} = \bar{r}_l'^{\mathrm{P}} \odot [W_{l+1}^{\mathrm{T}} (u_{l+1}^{\mathrm{P}} - W_{l+1} \bar{r}_l^{\mathrm{P}})]$ for $l = 1 \dots N-1$ and $\bar{e}_N = \beta \big(u_N^{\mathrm{target}} - u_N^{\mathrm{P}}\big)$ as well as $\bar{u}_l^{\mathrm{d},\infty} = B_l^{\mathrm{PP}} u_{l+1}^{\mathrm{P}} - W_l^{\mathrm{PI}} u_l^{\mathrm{I}}$ and $\bar{e}_l^{\mathrm{I}} = B_l^{\mathrm{IP}} u_{l+1}^{\mathrm{P}} - u_l^{\mathrm{I}}$. The local errors in the pyramidal neuron dynamics propagate backwards through the network following the error backpropagation algorithm (Eq. 5.15)

$$\bar{e}_l^{\mathrm{P}} = \bar{r}_l'^{\mathrm{P}} \odot W_{l+1}^{\mathrm{T}} \bar{e}_{l+1}^{\mathrm{P}} \,,$$

and synaptic plasticity minimizes the energy functions (Eqs. 5.10, 5.29 and 5.33)

$$\begin{aligned}
\frac{1}{\eta} \dot{W}_l &= -\lim_{\beta \to 0} \frac{1}{\beta} \frac{\partial E}{\partial W_l} = \lim_{\beta \to 0} \frac{1}{\beta} \big(u_l^{\mathrm{P}} - W_l\, \bar{r}_{l-1}^{\mathrm{P}}\big) \, \big(\bar{r}_{l-1}^{\mathrm{P}}\big)^{\mathrm{T}} \,, \\
\frac{1}{\eta^{\mathrm{IP}}} \dot{W}_l^{\mathrm{IP}} &= -\frac{\partial E^{\mathrm{I}}}{\partial W_l^{\mathrm{IP}}} \propto \big(u_l^{\mathrm{I}} - W_l^{\mathrm{IP}} \bar{r}_l^{\mathrm{P}}\big) \, \big(\bar{r}_l^{\mathrm{P}}\big)^{\mathrm{T}} \,, \\
\frac{1}{\eta^{\mathrm{PI}}} \dot{W}_l^{\mathrm{PI}} &= -\frac{\partial E^{\mathrm{d}}}{\partial W_l^{\mathrm{PI}}} \propto \big(B_l^{\mathrm{PP}} u_{l+1}^{\mathrm{P}} - W_l^{\mathrm{PI}} u_l^{\mathrm{I}}\big) \, \big(u_l^{\mathrm{I}}\big)^{\mathrm{T}} \,, \\
\dot{B}_l^{\mathrm{PP}} &= \dot{B}_l^{\mathrm{IP}} = 0 \,,
\end{aligned}$$

where in simulations, we assume $u_{\mathrm{rest}} = 0$ for convenience. The learning rules for forward weights $\dot{W}_l$ and pyramidal-to-interneuron weights $\dot{W}_l^{\mathrm{IP}}$ can be interpreted as learning by the dendritic prediction ($W_l\, \bar{r}_{l-1}$ and $W_l^{\mathrm{IP}} \bar{r}_l^{\mathrm{P}}$) of the somatic potential (u_l^{P} and u_l^{I}), (*Urbanczik and Senn*, 2014), whereas the interneuron-to-pyramidal weight dynamics $\dot{W}_l^{\mathrm{PI}}$ are compatible with the homeostatic plasticity of inhibitory synapses (inhibition $-W_l^{\mathrm{PI}} u_l^{\mathrm{I}}$ canceling excitation $B_l^{\mathrm{PP}} u_{l+1}^{\mathrm{P}}$)[25], (*Vogels, Sprekeler, Zenke, Clopath, and Gerstner*, 2011). The interneuron and apical voltage dynamics learn to approximate the local error $\bar{e}_l^{\mathrm{P}}$ appearing in the pyramidal neural dynamics, stored and calculated in the apical dendrites through a self-projecting interneuron microcircuit and upper-area feedback (Eqs. 5.30 and 5.35)

$$\bar{e}_l^{\mathrm{P}} \approx \bar{r}_l'^{\mathrm{P}} \odot \big(B_l^{\mathrm{PP}} u_{l+1}^{\mathrm{P}} - W_l^{\mathrm{PI}} u_l^{\mathrm{I}}\big) = \bar{r}_l'^{\mathrm{P}} \odot B_l^{\mathrm{PP}}\, \bar{e}_{l+1}^{\mathrm{P}} \,,$$

while the plasticity of the forward weights minimizes these local errors, and by doing so also minimizes a global cost function measuring network performance compared to some target behavior (Eqs. 5.10 and 5.14)

$$\frac{1}{\eta} \dot{W}_l = \lim_{\beta \to 0} \frac{1}{\beta} \bar{e}_l^{\mathrm{P}} \, \big[\bar{r}_{l-1}^{\mathrm{P}}\big]^{\mathrm{T}} = -\frac{\mathrm{d}C}{\mathrm{d}W_l} \quad \text{with} \quad C = \frac{1}{2} \| u_N^{\mathrm{target}} - u_N^{\mathrm{P}} \|^2 \,.$$

All of these dynamics occur simultaneously, without any need for separation of time scales or phases during learning, i.e., turning plasticity on and off during pattern presentations.

[25] Although for simplicity, we neglect strict inhibition/excitation in simulations.

5.3 Folded autoencoders for unsupervised learning in cortical hierarchies

In the previous section, the model was mostly restricted to supervised learning, and although the network structure is recurrent, there is a distinct asymmetry in functionality: forward weights are used for inference, whereas backward (or feedback) and lateral weights are only used for error propagation.

In the following, this approach is generalized to unsupervised learning (i.e., learning with unlabeled data) of networks with bidirectional connections, i.e., between every area, forward and backward weights exist, similar to a restricted Boltzmann machine. In the next section, we will show that this structure resembles a folded autoencoder. An autoencoder (*Hinton and Salakhutdinov*, 2006) is a feedforward network with visible – hidden – latent – hidden – output areas. Compared to the visible and hidden areas, the latent area has the fewest number of neurons, being a bottleneck for information flow in the network. The training objective is to replicate the input as output despite the bottleneck, forcing the network to find useful latent representations of the input data. In a folded autoencoder (*Wang, He, and Prokhorov*, 2012), the network structure is "folded" at the latent area, with the bottleneck now being the highest area of the network (Fig. 5.10A). Due to the recurrent structure, there are two distinct pathways: one from the visible to latent area, representing an encoding (or compression, discriminative pathway) of the visible input (Fig. 5.10B), and from latent to visible area, representing decoding (or decompression, generative pathway) of the latent activity (Fig. 5.10C). Thus, different from an autoencoder where encoder and decoder are separate, here both encoding and decoding happens through the same neurons. Bidirectional connectivity with generative feedback connections – or top-down modulation of activity – has been found to be essential for image processing in artificial and biological neural networks, increasing their robustness towards image occlusion and noise (*Wyatte, Curran, and O'Reilly*, 2012; *Spoerer, McClure, and Kriegeskorte*, 2017; *Tang, Schrimpf, Lotter, Moerman, Paredes,*

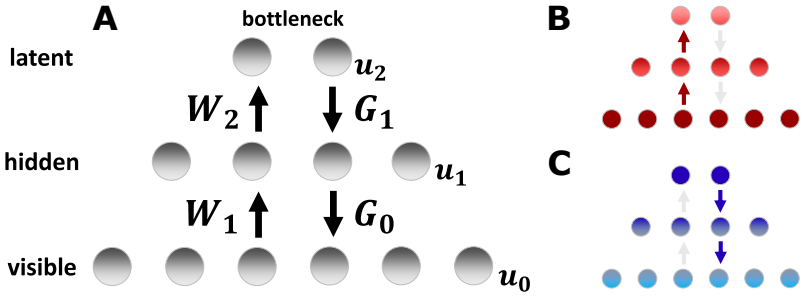

Figure 5.10.: **(A)** The network structure considered here consists of bidirectionally connected areas (visible, hidden, latent), where the highest area forms a bottleneck (less neurons than in the visible area). **(B)** The forward weights encode the activity of the visible area into a compressed latent representation. **(C)** The generative weights decode the compressed latent representation to drive activity in the visual area. Both pathways shown in (B) and (C) happen simultaneously and use the same neurons for coding. To train forward and backward weights, microcircuits as in Section 5.2.5 can be added to calculate errors (not shown here).

Caro, Hardesty, Cox, and Kreiman, 2018) and might even be beneficial against adversarial attacks (*Li, Bradshaw, and Sharma*, 2018; *Pontes-Filho and Liwicki*, 2018).

The network structure used in this section has already been proposed in *Seung* (1998) (and in a strict feedforward way, but with similar error coding to the one used here, in *Oh and Seung* 1998). The structure was also recently proposed to increase robustness against adversarial attacks in discriminative models (*Pontes-Filho and Liwicki*, 2018). However, in our model, encoding, decoding and error coding happens simultaneously through one set of neurons. Furthermore, learning is driven by nudging the visible area to its input (instead of, e.g., backpropagation through time). Thus, simply by nudging, the network learns a model of the observed data based on a compressed representation in the latent area[26]. By further restricting the latent space, this might allow the implementation of models similar to variational autoencoders (*Kingma and Welling*, 2013) in future work, as will be discussed in Section 5.3.3.

5.3.1 Energy function for cortical autoencoders

In order to combine simultaneous discriminative and generative learning in a simple network structure with only one visible area, we use a area-wise recurrent network with convexly gated forward and backward information flow, controlled by a parameter λ. We label the first area as the visible area, which receives both sensory input and top-down input from the first hidden area. The first hidden area receives both input from the visible area and the next, deeper hidden area. The highest area is labeled as the latent area, which forms a bottleneck, i.e., the number of neurons in the latent area is much smaller than in the visible area, similar to the latent area used in autoencoders to enforce the learning of useful features (Fig. 5.10A). This network structure is encoded in an energy function consisting of two error terms and a cost function from which the neurosynaptic dynamics are derived[27]

$$\frac{1}{g_\mathrm{l}}E = \frac{\lambda}{2}\sum_l \|u_l - W_l\bar{r}_{l-1}\|^2 + \frac{1-\lambda}{2}\sum_l \|u_l - G_l\bar{r}_{l+1}\|^2 + \frac{\beta}{2}C\,, \tag{5.36}$$

where u_l and r_l are the membrane potentials and rates of neurons in area l, W_l the discriminative weights from area $l-1$ to l, G_l the generative weights from area $l+1$ to l, g_l the leak conductance and βC the cost function with a scalar weighting $\beta \geq 0$. Discriminative and generative weights could also be combined in a single term $\|u_l - \lambda W_l\bar{r}_{l-1} - (1-\lambda)\,G_l\bar{r}_{l+1}\|^2$, but then all weights projecting into an area will minimize the same error, i.e., forward and backward weights interfere with each other, prohibiting the formation of useful features in higher areas. To allow numerical scalability to larger networks and data sets, we neglect lookaheads here and require that neural dynamics perform gradient descent on this energy

[26] This is similar to how sensory input acts on the cortex, only slightly modulating the ongoing, internally generated network activity (*Fiser, Chiu, and Weliky*, 2004; *Leinweber, Ward, Sobczak, Attinger, and Keller*, 2017).

[27] For ordinary autoencoders, a potential function can be derived that describes the autoencoder acting on the input (i.e., the transformation from input to output) in terms of gradient descent dynamics on said energy potential (*Kamyshanska and Memisevic*, 2014).

function (see Section 5.2.2), i.e., $c_\mathrm{m}\dot{u}_l = -\nabla_{u_l} E$. We obtain

$$\tau\dot{u}_l = -u_l + \lambda(W_l\bar{r}_{l-1} + \bar{e}_l^W) + (1-\lambda)(G_l\bar{r}_{l+1} + \bar{e}_l^G)\,, \tag{5.37}$$

$$\bar{e}_l^W = \bar{r}_l' \odot W_{l+1}^\mathrm{T}(u_{l+1} - W_{l+1}\bar{r}_l) \;\text{ with }\; \bar{e}_N^W = -\beta\,\nabla_{u_N} C\,, \tag{5.38}$$

$$\bar{e}_l^G = \bar{r}_l' \odot G_{l-1}^\mathrm{T}(u_{l-1} - G_{l-1}\bar{r}_l) \;\text{ with }\; \bar{e}_0^G = -\beta\,\nabla_{u_0} C\,, \tag{5.39}$$

with $\tau = \frac{c_\mathrm{m}}{g_\mathrm{l}}$ and N areas. This can be interpreted as a leaky integrator accumulating forward (W_l) and backward (G_l) input as well as the respective prediction errors $\bar{e}_l^W$ and $\bar{e}_l^G$, originating because neurons in area l try to predict the activity of neurons in the surrounding areas $l+1$ and $l-1$. Furthermore, forward and backward input is integrated convexly instead of equally, i.e., the discriminative and generative paths are not necessarily weighted equally. In a pyramidal neuron, such a separated integration might be implemented by projecting the generative and discriminative input to proximal and errors to distal compartments of the apical and basal tree. In fact, by rescaling the error function with a factor $\gamma = 1 + \frac{g_\epsilon}{g_\mathrm{l}}$ with small error conductance $g_\epsilon \ll g_\mathrm{l}$, the dynamics can be rewritten as a 5-compartment neuron composed of 4 dendritic branches and a soma (Fig. 5.11A)

$$c_\mathrm{m}\dot{u}_l = g_\mathrm{l}\lambda(\gamma W_l\bar{r}_{l-1} - u_l) + g_\mathrm{l}\hat{\lambda}(\gamma G_l\bar{r}_{l+1} - u_l) + g_\epsilon\lambda(\gamma\bar{e}_l^W - u_l) + g_\epsilon\hat{\lambda}(\gamma\bar{e}_l^G - u_l)\,, \tag{5.40}$$

$$\bar{e}_l^W = \frac{g_\mathrm{l}}{g_\epsilon}\bar{r}_l' \odot W_{l+1}^\mathrm{T}(u_{l+1} - W_{l+1}\bar{r}_l)\,, \tag{5.41}$$

$$\bar{e}_l^G = \frac{g_\mathrm{l}}{g_\epsilon}\bar{r}_l' \odot G_{l-1}^\mathrm{T}(u_{l-1} - G_{l-1}\bar{r}_l)\,, \tag{5.42}$$

with $\hat{\lambda} = 1 - \lambda$ (see Appendix B.1.3 for a derivation). g_l and g_ϵ represent dendritic transfer conductances. In this form, the generative input might, for instance, project to the apical tree, whereas the generative error projects to an oblique apical branch farther away, hence being weaker ($g_\epsilon \ll g_\mathrm{l}$). The factor γ represents a local amplification of the total synaptic inputs in an individual branch. The prediction errors $\bar{e}_l^W$ and $\bar{e}_l^G$ could be calculated via an interneuron circuit (see Section 5.2.5) or additional pyramidal neurons (*Keller and Mrsic-Flogel*, 2018). For our mathematical analysis we choose, without loss of generality, $\gamma = 1$, recovering Eq. 5.37. However, in simulations we use the 5-compartment neuron representation with $g_\epsilon > 0$, i.e., $\gamma \approx 1$, instead.

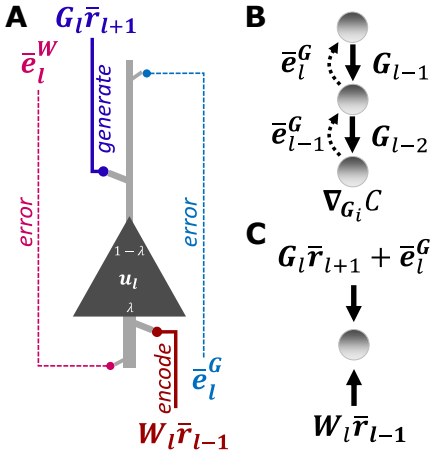

Figure 5.11.: **(A)** Possible implementation of bidirectional learning, with errors projecting to distal and inputs to proximal dendrites (gray branches), justifying the weak coupling $g_\epsilon \ll g_\mathrm{l}$ of errors. **(B)** Learning of the generative weights minimizes the reconstruction error C via error backpropagation. **(C)** Forward weights learn to match the generative top-down input (both signal and error). In a way, the encoder slowly learns to match the decoder.

5.3.2 One learning rule, two optimizations

The plasticity rules are derived as gradient descent on the energy function, i.e., we assign network states that create visible activity resembling the data to be learned low energy regimes

$$\dot{W}_l = -\eta \cdot \nabla_{W_l} E = \eta\, g_l \lambda \cdot (u_l - W_l \bar{r}_{l-1})\, \bar{r}_{l-1}^{\mathrm{T}}\,, \tag{5.43}$$

$$\dot{G}_l = -\eta \cdot \nabla_{G_l} E = \eta\, g_l (1-\lambda) \cdot (u_l - G_l \bar{r}_{l+1})\, \bar{r}_{l+1}^{\mathrm{T}}\,. \tag{5.44}$$

This can again be seen as learning being driven by the dendritic prediction ($W_l \bar{r}_{l-1}$ and $G_l \bar{r}_{l+1}$) of somatic activity (u_l) for both discriminative and generative weights (*Urbanczik and Senn*, 2014). If the network reaches a steady state, e.g., because we present a static input pattern to the network or the membrane time constant τ is much shorter than the pattern presentation times, the membrane potential is given by

$$u_l = \lambda(W_l \bar{r}_{l-1} + \bar{e}_l^W) + (1-\lambda)(G_l \bar{r}_{l+1} + \bar{e}_l^G)\,. \tag{5.45}$$

This can be used to rewrite the differences found in Eqs. 5.43 and 5.44:

$$u_l - W_l \bar{r}_{l-1} = (1-\lambda)\,(G_l \bar{r}_{l+1} + \bar{e}_l^G - W_l \bar{r}_{l-1}) + \lambda\, \bar{e}_l^W\,, \tag{5.46}$$

$$u_l - G_l \bar{r}_{l+1} = \lambda\,(W_l \bar{r}_{l-1} + \bar{e}_l^W - G_l \bar{r}_{l+1}) + (1-\lambda)\, \bar{e}_l^G\,. \tag{5.47}$$

As we are mostly interested in generative properties, that is, strong feedback and only weak forward information flow, we take the limit[28] $\lambda \to 0$

$$\lim_{\lambda\to 0}(u_l - W_l \bar{r}_{l-1}) = (1-\lambda)\,(G_l \bar{r}_{l+1} - W_l \bar{r}_{l-1} + \bar{e}_l^G)\,, \tag{5.48}$$

$$\lim_{\lambda\to 0}(u_l - G_l \bar{r}_{l+1}) = (1-\lambda)\,\bar{e}_l^G\,, \tag{5.49}$$

with feedback errors propagating backwards through the network:

$$\lim_{\lambda\to 0} \bar{e}_l^G = (1-\lambda)\,\bar{r}_l' \odot G_{l-1}^{\mathrm{T}}\, \bar{e}_{l-1}^G\,. \tag{5.50}$$

Using Eqs. 5.46 and 5.47, the plasticity rules take the form

$$\lim_{\lambda\to 0} \dot{G}_l = \eta\,(1-\lambda)^2 \cdot \bar{e}_l^G\, \bar{r}_{l+1}^{\mathrm{T}}\,, \tag{5.51}$$

$$\lim_{\lambda\to 0} \dot{W}_l = \eta\,(1-\lambda)\,\lambda \cdot (G_l \bar{r}_{l+1} + \bar{e}_l^G - W_l \bar{r}_{l-1})\, \bar{r}_{l-1}^{\mathrm{T}}\,. \tag{5.52}$$

$$\tag{5.53}$$

For the weights projecting into the last area we have:

$$\lim_{\lambda\to 0} \dot{W}_N = \eta\,(1-\lambda)\,\lambda \cdot (\bar{e}_N^G - W_N \bar{r}_{N-1})\, \bar{r}_{N-1}^{\mathrm{T}}\,. \tag{5.54}$$

[28] Taking this limit is important to guarantee that Eq. 5.10 approximately holds for the generative weights.

Figure 5.12.: Training of a network with $3072 - 1000 - 1000 - 300$ neurons on 10^4 CIFAR10 images. After training, the network is tested on a subset of the training data by encoding a visual stimulus (top rows) with gating $\lambda = 0.8$ (forward path dominates), reading out the latent representation, resetting network activity and then generating visual activity (bottom rows) with $\lambda = 0.2$ (backward path dominates). A similar experiment was done with MNIST with a correct split between training and test data, resulting in good reconstruction quality on both data subsets (not shown here, see, e.g., *Dold et al.* 2019b). During training on CIFAR10, we found that learning both paths adequately strongly depends on the chosen hyperparameters for large training data sets. E.g., for some parameters, the quality of reconstructed images deteriorates drastically after improving initially (not shown here).

Thus, even though the plasticity rules for forward and backward weights have the same shape (Eqs. 5.43 and 5.44), they perform two different optimizations[29]:

$$\lim_{\lambda \to 0} \dot{G}_l = -\eta \cdot \nabla_{G_l} E \propto -\nabla_{G_l} C \tag{5.55}$$

$$\lim_{\lambda \to 0} \dot{W}_l = -\eta \cdot \nabla_{W_l} E \propto -\nabla_{W_l} \| G_l \bar{r}_{l+1} + \bar{e}_l^G - W_l \bar{r}_{l-1} \|^2 \tag{5.56}$$

The learning rule for the generative weights reduces the cost function $C := \frac{1}{2} \| u_0^{\text{target}} - u_0 \|^2$ defined as a reconstruction error for the visible neurons (Fig. 5.11B), whereas the learning rule for the forward weights adjusts the bottom-up (discriminative) input of a neuron to match its top-down (generative) input (Fig. 5.11C). If the generative errors $\bar{e}_l^G$ vanish throughout the network, the learning rule can be rephrased as the encoder ($W_l \bar{r}_{l-1}$) learning to match

[29] For values of λ close to 0 or 1, the plasticity rules actually minimize the convex sum of two cost functions: the "matching cost" as well as an externally induced cost function C

$$\dot{G}_l \propto -\nabla_{G_l} \left[(1-\lambda)\, C + \lambda \| G_l \bar{r}_{l+1} - W_l \bar{r}_{l-1} - \bar{e}_l^W \|^2 \right] ,$$
$$\dot{W}_l \propto -\nabla_{W_l} \left[\lambda\, C + (1-\lambda) \| G_l \bar{r}_{l+1} + \bar{e}_l^G - W_l \bar{r}_{l-1} \|^2 \right] .$$

the decoder ($G_l \bar{r}_{l+1}$). To demonstrate that both pathways learn their respective optimization simultaneously, we train a network on parts of the CIFAR10 data set (*Krizhevsky, Hinton et al.*, 2009). The CIFAR10 data set is composed of colored images covering ten classes, e.g., airplanes, cars, dogs and cats, with an image size of 32×32 and varying backgrounds. In general, the gating λ remains constant and close to zero throughout the experiment. Only for testing whether forward and backward path learned to match each other, we change the gating to allow a separate investigation of both paths. After training, the forward path ($\lambda \to 1$, Fig. 5.10B) is able to encode presented images in the latent area, which can be fed into the backward path ($\lambda \to 0$, Fig. 5.10C) to reconstruct the original image (Fig. 5.12).

5.3.3 Towards probabilistic generative models

In the following, we give a suggestion how the previous model can be extended from a simple encoder-decoder architecture to a true generative model. However, these ideas are preliminary, mostly empirical and have to be refined in future research.

One prominent extension of autoencoders with powerful generative properties are so-called variational autoencoders (*Kingma and Welling*, 2013). Variational autoencoders further constrain the latent space by (i) introducing a stochastic sampling step and (ii) an additional term in the cost function to shape the distribution the latent space is sampling from. This way, after learning, latent activities can simply be sampled from the assumed latent distribution to create new images alike the data used for training. Furthermore, the learned data distribution allows simple transformation of generated images (e.g., changing the "angle" of a '1', or transforming a '1' smoothly into a '2') by manipulating activities of individual latent neurons without leaving the submanifold the data is living on (i.e., generating images that do not resemble data). Or in simpler terms: linear interpolations in the latent space lead to semantically meaningful variations in the visual space.

In a similar way, we also introduce sampling in the latent area N by adding Gaussian (white) noise $\xi_N \sim \mathcal{N}\left(0, \sigma^2\right)$ with variance σ^2 to the membrane dynamics:

$$\tau \dot{u}_N = -u_N + \lambda W_N \bar{r}_{N-1} + (1-\lambda)\bar{e}_N^G + \xi_N \,. \tag{5.57}$$

However, since learning is only guaranteed to work while network dynamics are stationary, we restrict ourselves to "frozen noise" for now, i.e., the noise vector ξ_N is only sampled in fixed intervals and not in every time step. As in the LIF-based sampling framework, this noise can be seen as network-generated noise (see Section 3.4). In absence of errors ($\bar{e}_N^G = 0$), the latent dynamics perform an Ornstein-Uhlenbeck process with drift towards the input of the previous area $W_N \bar{r}_{N-1}$ and diffusion given by the neuronal noise ξ_N,

$$p(u_N) = \mathcal{N}\left(W_N \bar{r}_{N-1},\, \sigma^2\right), \tag{5.58}$$

assuming the input pattern is static. To discourage a mean that shifts with presented patterns, the sampled distribution is further constrained by adding an additional term in

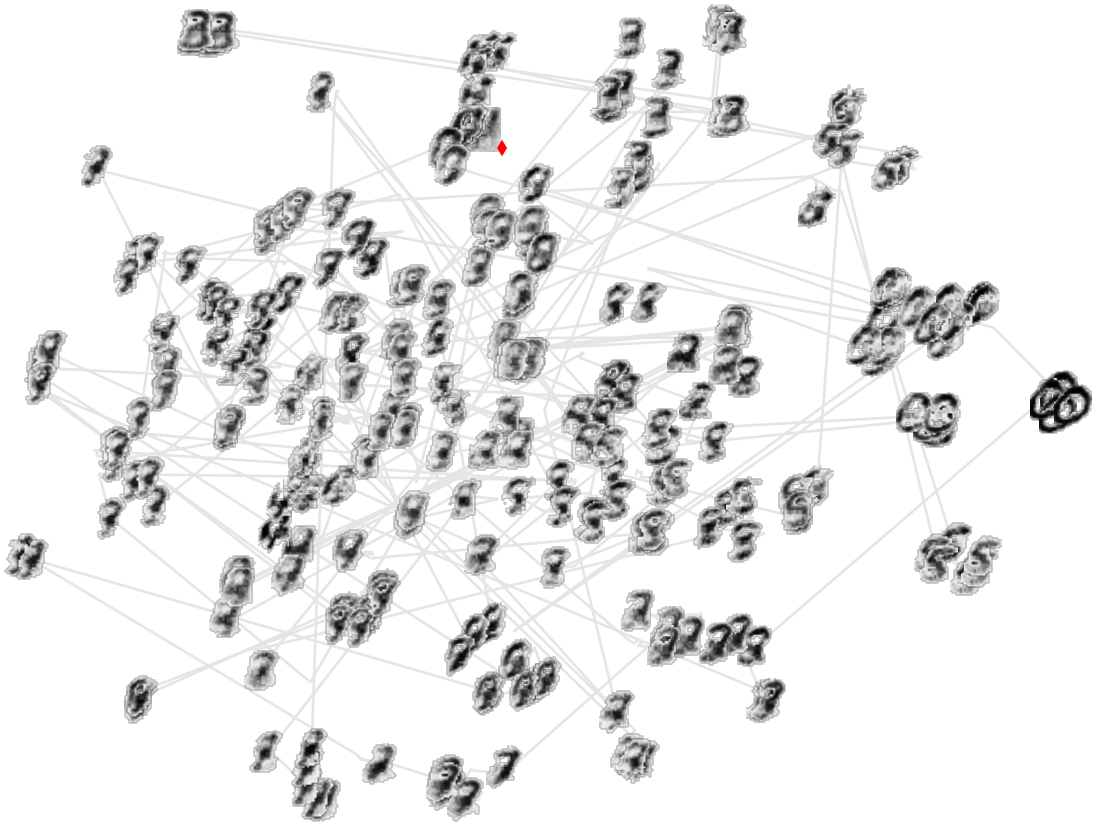

Figure 5.13.: Images generated by a network with $22 \times 18 - 200 - 20$ neurons after training, illustrated using t-SNE (gray lines connect consecutive images), where only every 10th sample is shown here. The red diamond marks the first image. To drive exploration, the frozen noise vector is resampled whenever the network reaches a steady state. The network is capable of generating recognizable images, like '0' (right), '2' (bottom, middle), '3' (top), '4' (middle), '5' (bottom right, top right), '8' (middle) and '9' (left) – although the image quality is improvable and lacks variability. In future work, the quality of the generated images might be further improved by tuning hyperparameters or changing the cost function.

the cost function[30]:

$$C^{\text{VAE}} = C + D_{\text{KL}}\left[p(u_N) \,\|\, \mathcal{N}\big(\mu_0, \sigma^2\big)\right] \quad \text{with} \quad \varphi(\mu_0) = 0\,, \tag{5.59}$$

$$= C + \frac{\|\mu_0 - W_N \bar{r}_{N-1}\|^2}{2\sigma^2} + \text{const.}\,, \tag{5.60}$$

i.e., we want the latent neurons to sample from a product of independent Gaussians with

[30] In variational autoencoders, both mean and variance are trained, while here, we only add a cost term for the mean. However, as shown in Section 3.4, the noise can be generated from other neurons with individual weights (or one effective weight). These weights could also be trained to improve the generative properties of the network.

mean μ_0 (independent of the presented pattern), chosen such that the mean rate response of latent neurons $\langle \bar{r}_N \rangle = 0$ vanishes. This leads to a modified prediction error for area $N-1$

$$\bar{e}^W_{N-1} = \bar{r}'_{N-1} \odot W_N^{\mathrm{T}} \left(u_N - W_N \bar{r}_{N-1} + \frac{\beta}{\sigma^2} \left(\mu_0 - W_N \bar{r}_{N-1} \right) \right) , \tag{5.61}$$

with two targets u_N and μ_0. After training, to generate images, the noise vector ξ_N is resampled whenever neuronal dynamics are stationary (i.e., sampling and network dynamics happen subsequently, in separate phases). This is shown in Fig. 5.13 for an example using the MNIST handwritten digits data set, where the network is capable of generating recognizable images of different digit classes. Although no images that look completely different from digits are generated, the variety and quality of the generated images is still lacking – compare these results, e.g., to the digits generated by the spike-based sampling networks in Fig. 3.23. In future work, the generative property of the proposed network might be further improved by investigating the effect of the additional cost term during learning in more detail.

5.4 Discussion

At the beginning of this chapter, we raise the question whether the brain might consult an optimization scheme like error backpropagation to learn input-output relations of sensory stimuli (Section 5.1). As the brain has to obey physical limitations (Section 5.1.2), exploring possible realizations of approximate error backpropagation is closely tied to mechanistic implementations of the dynamics that suffice realistic limitations like locality and time-continuity. Here, we extend previous work on backprop in the brain and propose a systematic top-down approach to learning in time-continuous systems (Section 5.2): from a scalar energy function E, both synaptic plasticity as well as compatible neuronal dynamics are derived – once via gradient descent and once via a prospective least action principle. The top-down approach enables a clear transition between the performed computation, its algorithmic solution, the derived dynamics implementing the algorithm and possible biological realizations thereof.

One crucial observation we make is the occurrence of lookahead dynamics (Section 5.2.4): in order to allow time-continuous dynamics, neurons respond with lookaheads of errors $e(t) \approx \bar{e}(t+\tau)$ and rates $r(t) \approx \varphi\left(u(t+\tau)\right)$, i.e., they approximate their behavior τ ms in the future to bridge temporal delays introduced by somatic and dendritic filtering. To first order $r(u, \dot{u}) = \bar{r} + \tau \bar{r}' \dot{u}$, this lookahead is equivalent to the inverse operation of low-pass filtering. Thus, by looking ahead in time, neurons can undo the delays enforced by filtering through membrane dynamics – which only requires knowledge about the neuron's current membrane potential value u and rate of change $\dot{u}$. Such a temporal prediction scheme is inspired by experimental observations (*Köndgen, Geisler, Fusi, Wang, Lüscher, and Giugliano*, 2008; *Palmer, Marre, Berry, and Bialek*, 2015). In fact, the advanced activation mechanism $r(u, \dot{u})$ resembles the dynamics underlying spike generation, which depends not only on u but also on $\dot{u}$ (*Hodgkin and Huxley*, 1952). However, further experimental studies are required to solidify this mechanism in biological neurons. Besides the lookahead response, the derived dynamics lend themselves to a biological implementation through

pyramidal neurons coupled with apical dynamics as well as an error-calculating microcircuit (Section 5.2.5 and *Sacramento, Costa, Bengio, and Senn* 2018). Both the microcircuit and apical dynamics can be derived again from energy functions and their corresponding actions, leading to a stacked application of our top-down approach – first for the original model, then for its extended version with errors being stored at the apical compartment, learned by microcircuits using lateral interneurons and top-down feedback.

Although the proposed model demonstrates that an optimization scheme like error backpropagation could be realized in nature, there are still many open questions. First of all, several mechanisms proposed in this model require proper testing, like the biological mechanism of lookaheads[31] and further details of the error calculation, like the modulation by the derivative of the postsynaptic rate $\bar{r}'$ (Eq. 5.30). Moreover, we only investigated error backpropagation in case of hierarchical networks and static input-output mappings. To learn sequential data, the proposed approach has to be extended to algorithms like backpropagation through time, for instance by employing biological plausible memory mechanisms to buffer past activations needed for correct temporal credit assignment (*Lillicrap and Santoro*, 2019).

For mathematical convenience, we use a rate-based model in this chapter, representing the population-averaged spike response of cortical neurons. Recently, error backpropagation for spiking neural networks has been heavily investigated (*Bohte, Kok, and La Poutré*, 2000; *O'Connor and Welling*, 2016; *Mostafa*, 2017; *Huh and Sejnowski*, 2018; *Zenke and Ganguli*, 2018; *Comsa, Potempa, Versari, Fischbacher, Gesmundo, and Alakuijala*, 2019; *Göltz et al.*, 2019; *Kheradpisheh and Masquelier*, 2019; *Bellec, Scherr, Hajek, Salaj, Legenstein, and Maass*, 2019), with many promising results. However, a mechanistic implementation of the whole learning process as presented here, coupled with spike-based communication or even spike-based error coding is still lacking. How elements of the presented model can be paired with spike-based computing is currently being investigated (ongoing work by Laura Kriener).

Furthermore, compared to the model presented in Chapter 3, the networks here are not stochastic and only used to perform classification and regression tasks. However, the original error backpropagation algorithm is pretty flexible in the sense that the cost function to be optimized can be freely chosen. Thus, as long as an appropriate and differentiable cost function exists, unsupervised as well as stochastic models can be trained using backprop (*Kingma and Welling*, 2013; *Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio*, 2014). One prominent example are variational autoencoders, a class of stochastic generative models combining autoencoders with sampling. Inspired by this, we introduce a folded autoencoder structure in Section 5.3 and demonstrate how this might be extended to networks similar to variational autoencoders – although this work is still in its conceptual phase and needs to be further investigated in the future. In the current version, we are able to demonstrate that the folded autoencoder can be trained unsupervised to learn compressed representations of data as well as generate fairly recognizable images through sampling. Unifying spike-based coding, probabilistic computing and end-to-end learning

[31] Which can actually become negative for fast changing inputs. Although we do not expect this to be problematic, as this happens only rarely, the impact of cutting off negative rates should be investigated more thoroughly in future work.

with gradient descent optimization in cortical networks would introduce a powerful model of brain-like computing, harnessing form (spikes, dendrites and microcircuits) to acquire function (learning probabilistic representations with local, error-driven plasticity rules).

Finally, in ongoing work by Akos F. Kungl (*Kungl, Dold, Riedler, Senn, and Petrovici*, 2019a; *Kungl*, 2020 - in prep.), the model is currently being extended to reinforcement learning, a framework where learning is driven by reward (or punishment) signals (*Sutton and Barto*, 2018). This is achieved by introducing a winner-nudges-all circuit in the output area, i.e., the neurons in the output area (i) receive noise to become stochastic and (ii) form a soft winner-take-all circuit (*Oster, Douglas, and Liu*, 2009). This way, the output area is capable of stochastically exploring solutions. The network output is given by the most active neuron in the output area, which slightly inhibits, or nudges, all other neurons of said area – taking on the role of the external target nudging. Combined with a global scalar reward (or punishment) signal, the network is able to generate its own output error, approximating a reinforcement learning scheme for deep cortical networks.

Details about simulations, calculations and implementations in this chapter can be found in Appendix B.

5.5 Contributions

The results in Section 5.2 were done in collaboration with Walter Senn and are currently in preparation for publication. The original idea is from Walter Senn, underpinning of the theory and proofs was done by Walter Senn and me, with help from João Sacramento, Mihai A. Petrovici and Akos F. Kungl. Experiment design and simulations were done by me. The presented theory in Section 5.3 was proposed and implemented by me. The formulation as a 5-compartment neuron (Eq. 5.40) is by Walter Senn.

6 | Epilog

As the end of Moore's law comes closer (*Waldrop*, 2016), it becomes essential to look into alternative ways of computing, distinct from our current understanding in terms of Boolean algebra and logic gates. The two most promising approaches tackle this question on two different levels:

- *Quantum computing* (*Nielsen and Chuang*, 2002) tries to utilize phenomenons from quantum mechanics to perform computations, like superposition, inference and entanglement. Thus, computing is done using the rules governing the dynamics of atomic and subatomic particles.

- *Neuromorphic computing*, or *bio-inspired artificial intelligence*, tries to understand the design principles of the most powerful computing device we currently know – the human brain – in an effort to recreate computational systems alike, but not necessarily identical to it.

Within the scope of this thesis, we address the second approach and ask how complex and flexible computations can naturally occur in physical systems like the brain, or similar systems like analog neuromorphic hardware; representing a class of devices containing unreliable and non-identical components – i.e., both temporal as well as fixed pattern noise. This is an unpleasantry every algorithmic design intended to harness the benefits of brain-inspired computing has to face. To tackle this question, we carefully tread between function and form, or more specifically, computational theory and its physical realization, guaranteeing an effortless transition from interpretive to mechanistic models (*Dayan and Abbott*, 2001) and vice versa. Furthermore, we draw inspiration not only from neuroscience, but also from deep learning to derive efficient and learning-capable mechanistic models, e.g., the neuronal least action principle takes inspiration from error backpropagation and the spike-based sampling networks from Boltzmann machines.

The results of this thesis are partitioned into two major chapters, one focusing on the explicit coding used for computation – asynchronous and discrete spike-based coding – and one on local, error-driven learning rules implementable by biological neuronal systems. We propose a self-consistent model of sampling with spikes, based on the idea of a network of networks using its own functional spike output as background noise input. When shaped by training data, such networks build a generative model of the observed data, allowing them to predict sensory stimuli, especially when occluded, noised or ambiguous inputs are presented to the network (pattern completion). Spike-based sampling ensembles have been prototyped on

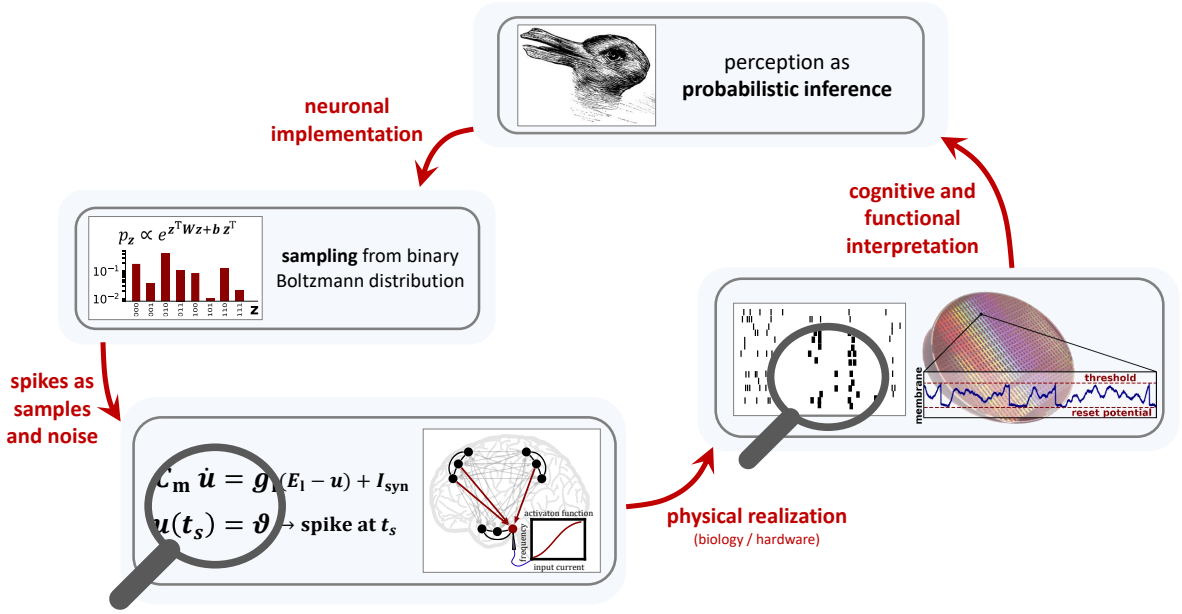

Figure 6.1.: Schematic illustration of the spike-based sampling framework. Probabilistic inference is implemented via sampling and its spike-based realization, allowing a functional and cognitive interpretation of spike activity and stochastic behavior in neuromorphic as well as biological substrates. A key feature of the framework is the proposed dual role of spikes, being carriers of function and noise at the same time, enabling a self-contained implementation of neural sampling.

the BrainScaleS–1 system (Section 3.4.6 and *Dold, Bytschok, Kungl, Baumbach, Breitwieser, Senn, Schemmel, Meier, and Petrovici* 2019a) and spike-based sampling with external noise sources on the BrainScaleS–1 and 2 architectures *Kungl et al.* (2019b); *Billaudelle et al.* (2019); hence, they have already been successfully realized on physical computing devices.

We further propose a theoretical framework, the neuronal least action principle, for gradient-based learning in dynamic neuronal systems, where neurons make use of both predictions in time (*what will my response be several ms in the future*) and predictions of network activity (*can the network predict / explain its own activity, or are there external factors?*) while learning is governed by predictive plasticity rules (*dendritic prediction of somatic activity* and *bottom-up prediction of top-down feedback*) as well. We are currently working on a spike-based implementation of the neuronal least action principle that allows a mapping to spiking neuromorphic hardware, either in analog (BrainScaleS) or digital (Loihi) circuits (ongoing work by Laura Kriener).

Both models can be described in a compact form linking computation, algorithm and implementation (*Marr*, 1982). For the spike-based model, the performed computation is probabilistic inference, algorithmically realized by Markov Chain Monte Carlo sampling and realized in a recurrently connected pool of spiking neurons (Fig. 6.1). For the neuronal least action principle, the performed computation is time-continuous learning of discriminative or generative models, algorithmically realized by error backpropagation. Through the neuronal least action principle, we obtain both a compressed network representation (the energy function) as well as neuronal and synaptic dynamics implementing this algorithm.

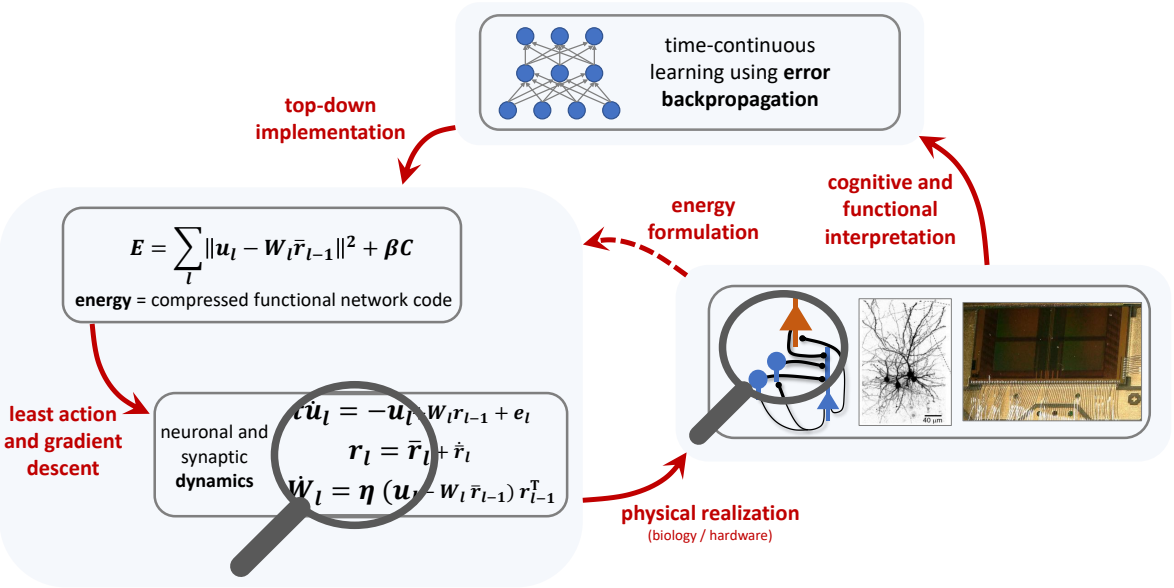

Figure 6.2.: Schematic illustration of the neuronal least action principle. On the computational and algorithmic side, we want to implement time-continuous gradient-based learning, realized by error backpropagation. The neuronal least action principle provides a top-down approach, starting with a compressed network code (the energy) from which neuronal and synaptic dynamics can be derived that implement the required computation. The dynamics can be mapped to features of biological neural networks, suggesting a cognitive and functional interpretation of them. These biological features, e.g., local microcircuits, can again be described by a functional network code and corresponding dynamics.

Furthermore, the dynamics can be mapped to features of biological neural networks, again describable via the neuronal least action principle (Fig. 6.2). We hope that the proposed models not only further our understanding of computing in the brain, but also pave the way to more biological network models and algorithms that inherit the functionality and clarity of their abstract counterparts in deep learning – or even surpass them. The results presented here demonstrate that biologically inspired neural networks exhibit intriguing and powerful functionalities useful for analog computing.

Still, there remain many open questions and challenges regarding coding and learning in the brain, hindering the progress that can be achieved with current neuromorphic technologies. One major challenge is to identify the scheme (or schemes) deployed by nature to encode and transform information in the brain. In this thesis, we work under the assumption that neurons turn analog signals (postsynaptic potentials) into digital ones (spikes), representing samples from a probability distribution. However, several other temporal coding schemes utilizing spike times have been proposed, e.g., latency codes (analog → delay to first spike, *Thorpe and Gautrais* 1998; *Mostafa* 2017; *Kheradpisheh and Masquelier* 2019; *Comsa, Potempa, Versari, Fischbacher, Gesmundo, and Alakuijala* 2019; *Göltz et al.* 2019) and rank ordering codes (analog → order of spikes in a population, *Thorpe and Gautrais* 1998). Biological neurons also show much more complex spiking behavior than simple LIF neurons (*Izhikevich*, 2004), e.g., spike bursts might play a crucial role for information processing and transmission (*Shai, Anastassiou, Larkum, and Koch*, 2015; *Zeldenrust, Wadman, and Englitz*, 2018) as well

as plasticity (*Bittner, Milstein, Grienberger, Romani, and Magee*, 2017). Alternatively, to encode information into precise spike times, spiking neural networks can be trained to output a target spike train given an input, similarly to how abstract neural networks are trained end-to-end to produce a target output given an input (*Zenke and Ganguli*, 2018). Discarding exact temporal information, integrated or averaged quantities like spike counts or rates can be used to encode analog signals (*Schmitt et al.*, 2017; *Wu, Chua, Zhang, Yang, Li, and Li*, 2019b), a coding scheme that we also employ in this thesis. Most likely, the brain utilizes a mixture of coding schemes suitable for the many tasks it has to perform, e.g., spike latency in retinal ganglion cells of salamanders carries a wealth of information about the spatial structure of briefly presented images, allowing quick perception of a visual scene to detect both prey and predators (*Gollisch and Meister*, 2008). Similarly, a sampling-based coding scheme allows decision making on both short and long time scales: an initial conclusion can already be made with the first sample (spike) and be further refined by accumulating more samples, better approximating the conditional probability distribution.

Throughout this work, we use two different plasticity rules: (i) a predictive rule (*dendritic prediction of somatic spiking*, Section 5.2) and a correlation-based rule (wake-sleep learning, Section 3.2). Although both plasticity rules are quite different, they can be tied to biological plasticity as observed in experiments. Plasticity in biological systems is commonly described via an unsupervised mechanism called Hebbian learning (*Hebb*, 1949), where changes in synaptic weights are a function of the pre- and postsynaptic activity, i.e., $\dot{W} = f\left(\bar{r}_{\text{pre}}, \bar{r}_{\text{post}}\right)$, with additional non-Hebbian contributions like homeostatic plasticity (*Turrigiano*, 2012), which stabilizes neuronal activity, or heterosynaptic plasticity (*Chistiakova, Bannon, Bazhenov, and Volgushev*, 2014), where plastic changes in a synapse also influence neighboring synapses. If we think of spiking neural networks, pre- and postsynaptic activity become pre- and postsynaptic spike time, and a causal spike pattern (presynaptic neuron spikes before postsynaptic neuron) leads to a strengthening of the synapse and an acausal spike pattern (post spikes before pre) to a weakening (*Bi and Poo*, 2001). The strength of the change commonly decays exponentially as a function of the delay between pre- and postsynaptic spike (see Fig. 6.3B), although many other dependencies have been observed as well (*Abbott and Nelson*, 2000). In *Clopath, Büsing, Vasilaki, and Gerstner* (2010), a voltage-based Hebbian plasticity rule, depending on presynaptic spike time and postsynaptic membrane potential, is introduced capable of reproducing the original spike time dependent plasticity (STDP) rule as well as additional phenomenological effects like spike frequency dependency (*Sjöström, Turrigiano, and Nelson*, 2001).

The spike-based version of the predictive learning rule used in Chapter 5 has been shown to reproduce several properties of biological plasticity when introducing backpropagating action potentials into the neuronal membrane dynamics, see Fig. 6.3A (results by *Spicher, Clopath, and Senn* 2016, 2018, 2019). For example, the rule is capable of reproducing the characteristic shape of STDP (Fig. 6.3B) as well as additional properties of synaptic weight updates like spike frequency dependence and synaptic distance dependence (not shown here) observed in biology (*Sjöström, Turrigiano, and Nelson*, 2001; *Sjöström and Häusser*, 2006). The wake-sleep algorithm used in Chapter 3 only depends on pre- and postsynaptic activity and therefore fits into the class of Hebbian learning rules. It was further shown that wake-sleep learning can be approximated using a symmetric STDP rule (*Neftci, Das, Pedroni, Kreutz-*

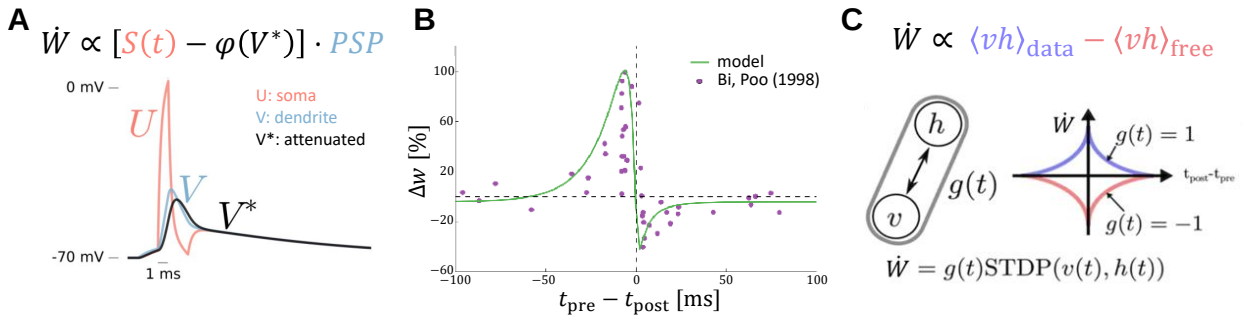

Figure 6.3.: **(A)** The plasticity rule introduced in *Urbanczik and Senn* (2014), learning by the dendritic prediction V^* of somatic spiking $S(t)$, where V^* is the attenuated dendritic potential V, shows several experimental properties of STDP when extended with action potentials (U) that propagate back into the dendritic tree. **(B)** The learning rule can recover the characteristic STDP weight update curve. Comparison is with experimental data by *Bi and Poo* (2001). Results and figures are taken from *Spicher et al.* 2016, 2018, 2019 (publication in prep.). **(C)** The wake-sleep algorithm (top) can be approximated using a symmetric STDP rule, taking on the role of measuring correlations between visible and hidden neurons. To implement the two phases, the STDP rule is multiplied by -1 for the free phase (colors mark the STDP rules for the two different phases of wake-sleep). Image adjusted from *Neftci et al.* (2014).

Delgado, and Cauwenberghs, 2014), which simply measures correlations between pre-and postsynaptic spiking, independent of the causality of the spikes. To implement the data and free phase of the wake-sleep algorithm, the STDP rule is flipped for the second phase, such that correlated activity leads to a weakening of synapses (see Fig. 6.3C). We are currently investigating how stable spike-based sampling is when (i) the weights and (ii) STDP updates are not perfectly symmetric and whether homeostatic mechanisms can be introduced into the learning rule to drive the network towards a symmetric weight configuration (ongoing work by Timo Gierlich). It should be noted here that in this work, we neglect Dale's law which states that biological neurons are exclusively excitatory (depolarizing effect on postsynaptic potential) or inhibitory (polarizing effect), depending on the neurotransmitters produced in the neuron. Similarly to neurons in deep learning, we allow neurons to be both excitatory and inhibitory, which might be realized in nature by more intricate circuitry between excitatory and inhibitory neurons.

Although conceptually pleasing and well rooted in experiments, simple STDP is not enough to optimize neural networks with similar success as in deep learning. A modification of basic Hebbian learning rules are so-called "three-factor learning rules", which introduce a third factor to the pre- and postsynaptic activity (*Frémaux and Gerstner*, 2016; *Kuśmierz, Isomura, and Toyoizumi*, 2017). This third factor can take the form of a reward signal, mediated by neuromodulators like dopamine, noradrenaline and serotonin, attentional feedback or supervised errors, introducing a more global signal into the plasticity rule informing the synapse about the overall performance of the network. Supervised errors might be propagated through biological neural networks using algorithms akin to error backpropagation, although as of now, it is not at all clear how the brain might accomplish such a feat. In Chapter 5, we propose the neuronal least action principle as one possible solution to this problem and briefly reviewed several others.

Taking a more network-centric approach to learning, where additional neural networks may take over key functionalities supporting learning in a principal network, several interesting ideas have been proposed in the deep learning literature. For instance, a crucial component for uniting reinforcement and deep learning is to add a playback memory (i.e., the possibility to learn from replays) to assist learning (*Mnih et al.*, 2015), inspired by the episodic memory found in the hippocampus (*Nicola and Clopath*, 2018). In case of learning via error backpropagation, generative adversarial networks (*Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, and Bengio*, 2014) have been proposed to learn appropriate cost functions for generative networks in a supervised architecture, and in *Jaderberg, Czarnecki, Osindero, Vinyals, Graves, Silver, and Kavukcuoglu* (2017), small subnetworks learn to represent the areawise errors required for error backpropagation in the principal network, lifting the requirement for actual backpropagation of errors to train it. Whether adversarial networks might be realized in a biologically plausible way is currently an open question (ongoing work by Nicolas Deperrois). In addition, a subnetwork might be trained to improve learning in the principal network, i.e., creating from one learning rule (subnetwork) a new, more efficient learning rule for the principal network, known as "meta-learning" or "learning to learn" in the literature (*Thrun and Pratt*, 2012; *Wang, Kurth-Nelson, Kumaran, Tirumala, Soyer, Leibo, Hassabis, and Botvinick*, 2018). All these examples demonstrate that there still remains a lot to be learned about "learning" that goes far beyond synaptic plasticity rules.

Throughout this thesis, we mostly use simulations and mathematical modeling, but stay close to mechanistic realizable systems. We further successfully prototype part of this work on an analog neuromorphic device, with more results expected in the future. Apart from the presented modeling methods used in this thesis, digital neuromorphic hardware (*Akopyan et al.*, 2015; *Davies et al.*, 2018; *Mayr, Hoeppner, and Furber*, 2019) is also a suitable substrate for investigating spike-based computing and local plasticity rules, without the negative effects of analog hardware, e.g., fixed pattern noise, but also without its benefits: power efficiency and speed. An intriguing future technology are memristors (a contraction of memory and resistor), an electrical component with history-dependent resistance that could potentially be used to build low-power plastic synapses as well as neurons in neuromorphic devices (*Xia and Yang*, 2019; *Yao, Wu, Gao, Tang, Zhang, Zhang, Yang, and Qian*, 2020) – however, this technology is still in its infancy. The previously mentioned approaches are all based on electronic circuits, but photonic solutions for neural network accelerators are being developed as well, e.g., *Brunner, Soriano, and Van Der Sand* (2019); *Hamerly, Bernstein, Sludds, Soljačić, and Englund* (2019), and represent an alternative realization of physical modeling systems. Currently, the most pressing issue of all neuromorphic platforms lies in suitable algorithms and scalability rivaling current AI standards. Still, as cited in the prolog, *if you want to understand a complicated device like a brain, you should build one* – which in our case translates to mathematical and physical modeling of brain-inspired circuits. This way, we hope to unravel the computational principles of the mammalian brain piece by piece, as done in this and many other recent works, to further our understanding of the brain and potentially overcome the initial challenges of the neuromorphic approach, initiating a new generation of neural networks molded into energy efficient artificial substrates.

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing, Computing machinery and intelligence, 1950

Acknowledgments

First of all, I would particularly like to thank Mihai A. Petrovici for supervising, mentoring, guiding and supporting me throughout the last years. I am very grateful for all the opportunities you opened up for me – and without you, many of the great achievements (and fun, or dare I say even *osum!*, times) of the past years would not have been possible! Furthermore, I would like to thank the late Karlheinz Meier for giving me the freedom to dive into the field of bio-inspired AI, as well as Andreas Mielke and Walter Senn for adopting me after his unforeseen demise. I would further like to express my gratitude to Walter Senn for welcoming me into his group in Bern, and especially for supporting and mentoring me. Thanks to you, I grew a lot in the last years. Many thanks also to João Sacramento for always being available for discussions during my time in Bern and beyond, as well as Rui P. Costa for helping me whenever I needed feedback. I would also like to extend a special thank you to Manfred Stärk for his continuous support and deep interest in my work.

Many thanks are in order to Katja Fahrion, Andi Baumbach, Akos F. Kungl, Hartmut Schmidt, Benjamin Ellenberger, Elena Kreutzer, Jakob Jordan, João Sacramento, Jana Fahrion, Jose Montes and Nicolas Deperrois for critical reading of this work and all the helpful suggestions. The text would be quite horrible without your help!

Special thanks to the occupants of the TMA office, especially the other two Musketeers, Akos F. Kungl and Andi Baumbach. I will miss you guys, especially our discussions about... well... everything! I won't miss Akos's list of quotes though. Going through the PhD with you was a great experience, and together, it was never a lonesome road. Also many thanks to the other long-term guests of the TMA office, like Max Zenk, Nico Gürtler, Julian Göltz, Madison Cotteret, Timo Gierlich, Malte Prinzler as well as everyone else (past and present) from the Electronic Vision(s) group, which are too numerous to be listed here (sorry!). Another special thank you goes to Elena Kreutzer, Chang Zhao, Dominik Spicher, Kristin Völk, Pascal Leimer and Willem Wybo for making me feel super welcome in Bern (and all the great times)! Even more special thanks go to my physics crew, Alex^2, Chris, Binh, JHB, Bine, Flo, Hannah, Jannis and Philipp (as well as the many (former) fellow students) for the awesome time in Heidelberg! I hope we will all stay in contact :) Moreover, a big thanks goes to Sven and Harald (and the others from Villy) for being such amazing lifelong friends!

Last but not least, many thanks to my Mom and Dad, my brother Michael and my sister Simone for raising me (haha) and always supporting and believing in me. You are the best![1] :) My deepest gratitude goes to Katja, for always helping, supporting and caring for me. You are the bright light that guides me and I am deeply grateful for having you at my side.

[1] An extra shout out goes to the glorious next generation, my nieces Lenja and Lara and my nephew Mats!

Funding

The research leading to these results has received funding from the European Union research and innovation programmes under grant agreement no 269921 (BrainScaleS) and 604102, 720270 and 785907 (Human Brain Project, HBP) and from the Manfred Stärk Foundation. Calculations were performed on UBELIX, the HPC cluster of the University of Bern, and the BwForCluster NEMO, the HPC cluster of the state Baden Württemberg for neuroscience, elementary particle physics, microsystems engineering and material science (NEMO), funded by the DFG through grant no INST 39/963-1 FUGG.

A | Appendix of chapter 3

A.1 Calculations

In the following, the mathematical analysis of results from Section 3.2 and Section 3.4 are shown. Appendix A.1.1 to Appendix A.1.7 are taken from *Dold, Bytschok, Kungl, Baumbach, Breitwieser, Senn, Schemmel, Meier, and Petrovici* (2019a) and were performed and written up by me as the sole author.

A.1.1 Free membrane potential distribution with colored noise

In the high-conductance state (HCS), it can be shown that the temporal evolution of the free membrane potential (FMP) of an leaky integrate-and-fire (LIF) neuron stimulated by balanced Poisson inputs is equivalent to an Ornstein-Uhlenbeck (OU) process with the following Green's function (*Petrovici*, 2016):

$$f(u,t|u_0) = \sqrt{\frac{1}{2\pi\sigma^2(1-e^{-2\theta t})}} \cdot \exp\left(-\frac{1}{2\sigma^2}\frac{(u-\mu+(\mu-u_0)e^{-\theta t})^2}{1-e^{-2\theta t}}\right). \tag{A.1}$$

with

$$\theta = \frac{1}{\tau^{\text{syn}}}, \tag{A.2a}$$

$$\mu = \frac{g_\text{l}E_\text{l} + \sum_{k\in\{\text{e,i}\}} \nu_k w_k E_k^{\text{rev}} \tau^{\text{syn}}}{\langle g^{\text{tot}}\rangle}, \tag{A.2b}$$

$$\sigma^2 = \frac{\sum_{k\in\{\text{e,i}\}} \nu_k w_k^2 (E_k^{\text{rev}}-\mu)^2 \tau^{\text{syn}}}{\langle g^{\text{tot}}\rangle^2}, \tag{A.2c}$$

$$\langle g^{\text{tot}}\rangle = g_\text{l} + \sum_{k\in\{\text{e,i}\}} w_k \nu_k \tau_k^{\text{syn}}, \tag{A.2d}$$

where ν_k are the noise frequencies, w_k the noise weights and we dropped the index notation u_k^{free} used in previous sections for convenience. The stationary FMP distribution is then given by a Gaussian (*Gerstner and Kistler*, 2002; *Petrovici*, 2016):

$$f(u) = \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right). \tag{A.3}$$

Replacing the white noise $\eta(t)$ in the OU process, defined by $\langle\eta\rangle = \text{const.}$ and $\langle\eta(t)\eta(t')\rangle = \nu\delta(t-t') + \nu^2$ (*Gerstner and Kistler*, 2002), with (Gaussian) colored noise η_c, defined by $\langle\eta_c\rangle = \text{const.}$ and $\langle\eta_c(t)\eta_c(t')\rangle = \gamma(t-t')$ (*Häunggi and Jung*, 1994) where $\gamma(t-t')$ is a function that does not vanish for $t - t' \neq 0$, the stationary solution of the FMP distribution is still given by a Gaussian with mean μ' and width σ' (*Häunggi and Jung*, 1994; *Cáceres*, 1999). Since the noise correlations only appear when calculating higher-order moments of the FMP, the mean value of the FMP distribution remains unchanged $\mu' = \mu$. However, the variance $\sigma'^2 = \langle\big(u(t) - \langle u(t)\rangle\big)^2\rangle$ of the stationary FMP distribution changes due to the correlations, as discussed in the next section.

A.1.2 Width of free membrane potential distribution

In the HCS, the FMP can be approximated analytically as (*Petrovici*, 2016)

$$u(t) = u_0 + \sum_{k\in\{\mathrm{e,i}\}} \sum_{\text{spikes s}} \Lambda_k \Theta(t - t_s) \cdot \left[\exp\left(-\frac{t-t_s}{\tau_k^{\mathrm{syn}}}\right) - \exp\left(-\frac{t-t_s}{\langle\tau_{\mathrm{eff}}\rangle}\right)\right], \tag{A.4}$$

with

$$u_0 = \frac{g_\mathrm{l} E_\mathrm{l} + (\langle g^{\mathrm{tot}}\rangle - g_\mathrm{l})\mu}{\langle g^{\mathrm{tot}}\rangle}, \tag{A.5a}$$

$$\Lambda_k = \frac{\tau_k^{\mathrm{syn}} w_k \big(E_k^{\mathrm{rev}} - \mu\big)}{\langle g^{\mathrm{tot}}\rangle \left(\tau_k^{\mathrm{syn}} - \langle\tau_{\mathrm{eff}}\rangle\right)}, \tag{A.5b}$$

$$\langle\tau_{\mathrm{eff}}\rangle = \frac{C_\mathrm{m}}{\langle g^{\mathrm{tot}}\rangle}. \tag{A.5c}$$

By explicitly writing the excitatory and inhibitory noise spike trains as $S_{\mathrm{e/i}}(t') = \sum_{\text{spikes s}} \delta(t' - t_s)$, this can be rewritten to

$$u(t) = u_0 + \sum_{k\in\{\mathrm{e,i}\}} \Lambda_k \int \mathrm{d}t' S_k(t') \Theta(t - t') \cdot \left[\exp\left(-\frac{t-t'}{\tau_k^{\mathrm{syn}}}\right) - \exp\left(-\frac{t-t'}{\langle\tau_{\mathrm{eff}}\rangle}\right)\right] \tag{A.6a}$$

$$= u_0 + \Lambda_\mathrm{e}\big(S_\mathrm{e} * \kappa_e\big)(t) + \Lambda_\mathrm{i}\big(S_\mathrm{i} * \kappa_i\big)(t) \tag{A.6b}$$

$$= u_0 + \big[(\Lambda_\mathrm{e} S_\mathrm{e} + \Lambda_\mathrm{i} S_\mathrm{i}) * \kappa\big](t), \tag{A.6c}$$

where $*$ denotes the convolution operator and with

$$\kappa_{\mathrm{e/i}}(t) = \Theta(t)\left[\exp\left(-\frac{t}{\tau_{\mathrm{e/i}}^{\mathrm{syn}}}\right) - \exp\left(-\frac{t}{\langle\tau_{\mathrm{eff}}\rangle}\right)\right]. \tag{A.7}$$

For simplicity, we assume $\tau_{\mathrm{e}}^{\mathrm{syn}} = \tau_{\mathrm{i}}^{\mathrm{syn}}$. The width of the FMP distribution can now be calculated as

$$\begin{aligned}
&\left\langle \big(u(t) - \langle u(t)\rangle\big)^2 \right\rangle = \langle u(t)^2\rangle - \langle u(t)\rangle^2 && \text{(A.8a)}\\
&= \left\langle \big[u_0 + (S_{\mathrm{tot}} * \kappa)(t)\big]^2 \right\rangle - \langle u_0 + (S_{\mathrm{tot}} * \kappa)(t)\rangle^2 && \text{(A.8b)}\\
&= \left\langle \big[(S_{\mathrm{tot}} * \kappa)(t)\big]^2 \right\rangle - \langle (S_{\mathrm{tot}} * \kappa)(t)\rangle^2 && \text{(A.8c)}\\
&= \left\langle \big[(S_{\mathrm{tot}} * \kappa)(t) - \langle (S_{\mathrm{tot}} * \kappa)(t)\rangle\big]^2 \right\rangle , && \text{(A.8d)}
\end{aligned}$$

where the average is calculated over t and $S_{\mathrm{tot}}(t) = \Lambda_{\mathrm{e}} S_{\mathrm{e}}(t) + \Lambda_{\mathrm{i}} S_{\mathrm{i}}(t)$. Since the average is an integral over t, i.e. $\langle(\cdot)\rangle \to \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2}(\cdot)\,\mathrm{d}t$, we can use the identity $\int (f * g)(t)\,\mathrm{d}t = (\int f(t)\,\mathrm{d}t)(\int g(t)\,\mathrm{d}t)$, that is $\langle f * g\rangle = \langle f\rangle \int g(t)\,\mathrm{d}t = \langle f\rangle * g$ in the limit of $T \to \infty$, to arrive at the following solution:

$$\begin{aligned}
&\left\langle \big(u(t) - \langle u(t)\rangle\big)^2 \right\rangle \\
&= \left\langle \big[(S_{\mathrm{tot}}(t) - \langle S_{\mathrm{tot}}(t)\rangle) * \kappa(t)\big]^2 \right\rangle . && \text{(A.9)}
\end{aligned}$$

More generally, we obtain with a similar calculation the autocorrelation function (ACF) of the FMP:

$$\begin{aligned}
&\langle \bar{u}(t)\bar{u}(t+\Delta)\rangle \\
&= \left\langle \big((\bar{S}_{\mathrm{tot}} * \kappa)(t)\big)\big((\bar{S}_{\mathrm{tot}} * \kappa)(t+\Delta)\big)\right\rangle , && \text{(A.10)}
\end{aligned}$$

with $\bar{x}(t) = x(t) - \langle x(t)\rangle$ and by using $\langle \bar{u}(t)\bar{u}(t+\Delta)\rangle = \langle u(t)u(t+\Delta)\rangle - \langle u(t)\rangle^2$. This can be further simplified by applying the Wiener–Khintchine theorem (*Wiener*, 1930; *Khintchine*, 1934), which states that $\lim_{T\to\infty} \langle x(t)x(t+\Delta)\rangle_T = \mathcal{F}^{-1}\big(|\mathcal{F}(x)|^2\big)(\Delta)$ with $\langle(\cdot)\rangle_T \to \int_{-T/2}^{T/2}(\cdot)\,\mathrm{d}t$ (due to $\int x(t)x(t+\Delta)\,\mathrm{d}t = \big(x(t) * x(-t)\big)(\Delta)$). Thus, for the limit $T \to \infty$, we can rewrite this as

$$\begin{aligned}
&\left\langle \big((\bar{S}_{\mathrm{tot}} * \kappa)(t)\big)\big((\bar{S}_{\mathrm{tot}} * \kappa)(t+\Delta)\big)\right\rangle \\
&= \lim_{T\to\infty} \frac{1}{T} \mathcal{F}^{-1}\big(|\mathcal{F}(\bar{S}_{\mathrm{tot}} * \kappa)|^2\big)(\Delta) && \text{(A.11a)}\\
&= \lim_{T\to\infty} \frac{1}{T} \mathcal{F}^{-1}\big(|\mathcal{F}(\bar{S}_{\mathrm{tot}})\mathcal{F}(\kappa)|^2\big)(\Delta) && \text{(A.11b)}\\
&= \lim_{T\to\infty} \frac{1}{T} \Big(\mathcal{F}^{-1}\big(|\mathcal{F}(\bar{S}_{\mathrm{tot}})|^2\big) \\
&\quad * \mathcal{F}^{-1}\big(|\mathcal{F}(\kappa)|^2\big) \Big)(\Delta) , && \text{(A.11c)}
\end{aligned}$$

and by applying the Wiener–Khintchine theorem again in reverse

$$\begin{aligned}
\langle \bar{u}(t)\bar{u}(t+\Delta)\rangle = \big(&\lim_{T\to\infty} \frac{1}{T} \left\langle \big[\bar{S}_{\mathrm{tot}}(t)\big]\big[\bar{S}_{\mathrm{tot}}(t+\Delta')\big]\right\rangle_T \\
&* \left\langle \big[\kappa(t)\big]\big[\kappa(t+\Delta')\big]\right\rangle_\infty \big)(\Delta) , && \text{(A.12)}
\end{aligned}$$

where the variance of the FMP distribution is given for $\Delta = 0$. Thus, the unnormalized ACF of the FMP can be calculated by convolving the unnormalized ACF of the background spike trains (S_{tot}) and the postsynaptic potential (PSP) shape (κ). In case of independent excitatory and inhibitory Poisson noise (i.e., $\langle \bar{S}(t)\bar{S}(t')\rangle = \nu\delta(t-t')$), we get

$$\begin{aligned}\left\langle \left[\bar{S}_{\text{tot}}(t)\right]\left[\bar{S}_{\text{tot}}(t+\Delta')\right]\right\rangle &= \Lambda_{\text{e}}^2 \left\langle \bar{S}_{\text{e}}(t)\bar{S}_{\text{e}}(t+\Delta')\right\rangle \\ &\quad + \Lambda_{\text{i}}^2 \left\langle \bar{S}_{\text{i}}(t)\bar{S}_{\text{i}}(t+\Delta')\right\rangle && \text{(A.13a)} \\ &= \sum_{k\in\{\text{e,i}\}} \Lambda_k^2 \nu_k \delta(\Delta') && \text{(A.13b)}\end{aligned}$$

and therefore

$$\begin{aligned}\text{Var}(u) &= \Big(\sum_{k\in\{\text{e,i}\}} \Lambda_k^2 \nu_k \delta(\Delta') \\ &\quad * \left\langle \left[\kappa(t)\right]\left[\kappa(t+\Delta')\right]\right\rangle_{\infty} \Big)(\Delta = 0) && \text{(A.14a)} \\ &= \sum_{k\in\{\text{e,i}\}} \Lambda_k^2 \nu_k \left\langle \kappa^2(t)\right\rangle && \text{(A.14b)} \\ &= \sum_{k\in\{\text{e,i}\}} \Lambda_k^2 \nu_k \int_0^\infty \kappa^2(t)\ \mathrm{d}t\,, && \text{(A.14c)}\end{aligned}$$

which agrees with the result given in *Petrovici* (2016). If the noise spike trains are generated by processes with refractory periods, the absence of spikes between refractory periods leads to negative contributions in the ACF of the noise spike trains. This leads to a reduced value of the variance of the FMP and hence, also to a reduced width of the FMP distribution. The factor $\sqrt{\beta}$ by which the width of the FMP distribution changes due to the introduction of colored background noise is given by

$$\begin{aligned}\beta &= \frac{\sigma^2_{\text{colored}}}{\sigma^2_{\text{Poisson}}} && \text{(A.15)} \\ &= \frac{\int \mathrm{d}\Delta\ \left\langle \bar{S}_{\text{tot}}(t)\bar{S}_{\text{tot}}(t+\Delta)\right\rangle \cdot \int \mathrm{d}t\ \kappa(t)\kappa(t+\Delta)}{\sum_{k\in\{\text{e,i}\}} \Lambda_k^2\nu_k \int_0^\infty \kappa^2(t)\ \mathrm{d}t}\,. && \text{(A.16)}\end{aligned}$$

For the simplified case of a Poisson process with refractory period, one can show that $\int \mathrm{d}\Delta\ \langle \bar{S}_{\text{tot}}(t)\bar{S}_{\text{tot}}(t+\Delta)\rangle$ has a reduced value compared to a Poisson process without refractory period (*Gerstner and Kistler*, 2002), leading to $\beta \leq 1$. Even though we do not show this here for neuron-generated spike trains, the two cases are similar enough that $\beta \leq 1$ can be assumed to apply in this case as well.

In the next section, we will show that the factor β can be used to rescale the inverse slope of the activation function to transform the activation function of a neuron receiving white noise to the activation function of a neuron receiving equivalent (in frequency and weights), but colored noise. That is, the rescaling of the FMP distribution width due to the autocorrelated background noise translates into a rescaled inverse slope of the activation function.

A.1.3 Approximate inverse slope of LIF activation function

As stated earlier, the FMP of an LIF neuron in the HCS is described by an OU process with a Gaussian stationary FMP distribution (both for white and colored background noise). As a first approximation, we can define the activation function as the probability of the neuron having a FMP above threshold (see Eq. A.3)

$$p(z_i = 1) \approx \int_{\vartheta}^{\infty} f(u)\mathrm{d}u \tag{A.17}$$

$$= \int_{\vartheta}^{\infty} \sqrt{\frac{1}{2\pi\sigma^2}} \exp\left(-\frac{(u-\mu)^2}{2\sigma^2} \right) \mathrm{d}u \tag{A.18}$$

$$= \frac{1}{2}\left(1 - \mathrm{erf}\left(\frac{\vartheta - \mu}{\sqrt{2}\sigma}\right)\right). \tag{A.19}$$

Even though this is only an approximation (as we are neglecting the effect of the reset), the error function is already similar to the logistic activation function observed in simulations (*Petrovici, Bill, Bytschok, Schemmel, and Meier*, 2016).

The inverse slope of a logistic activation function is defined at the inflection point, i.e.,

$$\alpha^{-1} = \frac{\mathrm{d}}{\mathrm{d}\mu}\varphi\left(\frac{\mu - u_0}{\alpha}\right)\bigg|_{\mu=u_0}. \tag{A.20}$$

By calculating the inverse slope via the activation function derived from the FMP distribution, we get

$$\alpha^{-1} = \frac{\mathrm{d}}{\mathrm{d}\mu} p(z_i = 1)\bigg|_{\mu=\vartheta}, \tag{A.21}$$

$$= \sqrt{\frac{1}{2\pi\sigma^2}}, \tag{A.22}$$

from which it follows that the inverse slope α is proportional to the width of the FMP distribution σ. Thus, rescaling the variance of the FMP distribution by a factor β leads, approximately, to a rescaling of the inverse slope of the activation function $\alpha' = \sqrt{\beta}\alpha$.

A.1.4 Origin of side-peaks in the noise autocorrelation function

For high rates, the spike train generated by an LIF neuron in the HCS shows regular patterns of interspike intervals which are roughly equal to the absolute refractory period. This occurs (i) due to the refractory period introducing regularity for higher rates, since ISI's $< \tau_{\mathrm{ref}}$ are not allowed and the maximum firing rate of the LIF neuron is bounded by $\frac{1}{\tau_{\mathrm{ref}}}$, and (ii) due to an LIF neurons's tendency to spike consecutively when the effective membrane potential

$$u_{\mathrm{eff}}(t) = \frac{g_{\mathrm{l}}E_{\mathrm{l}} + \sum_{k\in\{\mathrm{e,i}\}} g_k^{\mathrm{syn}}(t)E_k^{\mathrm{rev}}}{g^{\mathrm{tot}}(t)}, \tag{A.23}$$

$$\tau_{\mathrm{eff}}\dot{u} = u_{\mathrm{eff}} - u, \tag{A.24}$$

is suprathreshold after the refractory period (*Petrovici*, 2016). The probability of a consecutive spike after the neuron has spiked once at time t is given by (under the assumption of the HCS)

$$\begin{aligned} p_1 &= p(\text{spike at } t+\tau_{\text{ref}} | \text{ first spike at } t) \\ &= \int_{\vartheta}^{\infty} \mathrm{d}u_{t+\tau_{\text{ref}}} f(u_{t+\tau_{\text{ref}}},\ \tau_{\text{ref}} \mid u_t = \vartheta)\,, \end{aligned} \tag{A.25}$$

due to the effective membrane potential following an OU process (whereas the FMP is a low-pass filter thereof, however with a very low time constant τ_{eff}), see Eq. A.1. The probability to spike again after the second refractory period is then given by

$$\begin{aligned} p_2 &= p(\text{spike at } t+2\tau_{\text{ref}} | \text{ spike at } t+\tau_{\text{ref}},\ t) \\ &= \frac{\int_{\vartheta}^{\infty}\int_{\vartheta}^{\infty} \mathrm{d}u_2 \mathrm{d}u_1\ f(u_2, \tau_{\text{ref}} \mid u_1) f(u_1, \tau_{\text{ref}} \mid u_0 = \vartheta)}{\int_{\vartheta}^{\infty} \mathrm{d}u_1\ f(u_1, \tau_{\text{ref}} \mid u_0 = \vartheta)}\,, \end{aligned} \tag{A.26}$$

with $u_n = u_{t+n\tau_{\text{ref}}}$, or in general after $n-1$ spikes

$$\begin{aligned} p_n &= p(\text{spike at } t+n\tau_{\text{ref}} \mid \text{spike at } t+(n-1)\tau_{\text{ref}}, \ldots,\ t) \\ &= \int_{\vartheta}^{\infty} \mathrm{d}u_{n-1}\ f^{(n-1)}(u_{n-1})\,, \end{aligned} \tag{A.27}$$

$$f^n(u_n) = \frac{\int_{\vartheta}^{\infty} \mathrm{d}u_{n-1}\ f(u_n, \tau_{\text{ref}} \mid u_{n-1}) f^{(n-1)}(u_{n-1})}{\int_{\vartheta}^{\infty} \mathrm{d}u_{n-1}\ f^{(n-1)}(u_{n-1})}\,, \tag{A.28}$$

for $n > 1$ and $f^1(u_1) = f(u_1, \tau_{\text{ref}} \mid u_0 = \vartheta)$. The probability to observe n spikes in such a sequence is then given by

$$P_n = \prod_{i=1}^{n-1} p_i\,, \tag{A.29}$$

and the probability to find a burst of length n (i.e., the burst ends)

$$p(\text{burst of length n}) = P_n \cdot (1 - p_n)\,. \tag{A.30}$$

With this, one can calculate the average length of the occurring bursts $\sum_{i=1}^{\infty} i \cdot p(\text{burst of length i})$, from which we can already see how the occurrence of bursts depends on the mean activity of the neuron. A simple solution can be found for the special case of $\tau^{\text{syn}} \ll \tau_{\text{ref}}$, since then the effective membrane potential distribution has already converged to the stationary distribution after every refractory period, i.e., $f(u_n, \tau_{\text{ref}} \mid u_{n-1}) = f(u_n)$ and hence

$$\begin{aligned} p_n &= p(\text{spike at } t+n\tau_{\text{ref}} | \text{ spike at } t+(n-1)\tau_{\text{ref}}, \ldots,\ t) \\ &= \int_{\vartheta}^{\infty} \mathrm{d}u f(u) = \bar{p} \end{aligned} \tag{A.31}$$

for all n. Thus, for this special case the average burst length can be expressed as

$$\sum_{i=1}^{\infty} i \cdot p(\text{burst of length i}) = \sum_{i=1}^{\infty} i \cdot \bar{p}^{i-1}(1-\bar{p})\,, \tag{A.32}$$

$$= \frac{1}{1-\bar{p}}\,. \tag{A.33}$$

By changing the mean membrane potential (e.g., by adjusting the leak potential or adding an external (bias) current), the probability of consecutive spikes $\bar{p}$ can be directly adjusted and hence, also the average length of bursts. Since these bursts are fixed structures with interspike intervals equal to the refractory period, they translate into side-peaks at multiples of the refractory period in the spike train ACF, as we demonstrate below.

The ACF of the spike train S is given by

$$\mathcal{C}(S,S,\Delta) = \frac{\langle S_t S_{t+\Delta}\rangle - \langle S\rangle^2}{\mathrm{Var}(S)}\,, \tag{A.34}$$

where the first term of the numerator is $\langle S_t S_{t+\Delta}\rangle = p(\text{spike at } t+\Delta, \text{spike at } t)$ (notation as in Eqs. A.6a and A.8a). This term can be expressed as

$$p(\text{spike at } t+\Delta, \text{spike at } t)$$

$$= p(\text{spike at } t+\Delta \mid \text{spike at } t)\cdot p(\text{spike at } t)\,, \tag{A.35}$$

$$= p(\text{spike at } t+\Delta \mid \text{spike at } t)\cdot \langle S\rangle\ , \tag{A.36}$$

where we assumed that the first spike starts the burst at a random time t. Therefore, in order to calculate the ACF, we have to calculate the probability that a spike occurs at time $t+\Delta$ given that the neuron spikes at time t. This has to include every possible combination of spikes during this interval. In the following, we argue that at multiples of the refractory period, the main contribution to the ACF comes from bursts.

- First, for $\Delta < \tau_{\mathrm{ref}}$, the term $p(\text{spike at } t+\Delta \mid \text{spike at } t)$ in Eq. A.35 vanishes since the neuron is refractory and cannot spike during this interval. Thus, the ACF becomes negative as only the term $-\frac{\langle S\rangle^2}{\mathrm{Var}(S)}$ in Eq. A.34 remains, where both numerator and denominator are positive.

- For $\Delta = \tau_{\mathrm{ref}}$, a spike can only occur when the neuron bursts with probability $p_1 = \int_{\vartheta}^{\infty} \mathrm{d}u_{t+\tau_{\mathrm{ref}}} f(u_{t+\tau_{\mathrm{ref}}},\ \tau_{\mathrm{ref}} \mid u_t = \vartheta)$, where we assumed for simplicity that the first spike starts the burst spontaneously.

- Since for $\tau_{\mathrm{ref}} < \Delta < 2\tau_{\mathrm{ref}}$, the neuron did not burst with probability $1-p_1$, it is possible to find a spike in this interval, leading again to negative, but diminished, values in the ACF.

- For $\Delta = 2\tau_{\mathrm{ref}}$, we now have two ways to observe spikes at t and $t+2\tau_{\mathrm{ref}}$: (i) The spikes are part of a burst of length 2 or (ii) there was no intermediate spike and the spikes have an ISI of $2\tau_{\mathrm{ref}}$. Since for larger rates, having large ISIs that are exact multiples of τ_{ref} is unlikely, we can neglect the contribution of (ii).

- If we go further to $\Delta = n\tau_{\text{ref}}$, we get even more additional terms including bursts of length $< n$. However, these terms can again be neglected as, compared to having a burst of length n, it is rather unlikely to get a burst pattern with missing intermediate spikes, i.e., having partial bursts which are a multiple of τ_{ref} apart.
- Finally, for $\Delta \to \infty$, we have $\langle S_t S_{t+\Delta}\rangle - \langle S\rangle^2 = \langle S_t\rangle \langle S_{t+\Delta}\rangle - \langle S\rangle^2 = \langle S\rangle \langle S\rangle - \langle S\rangle^2 = 0$ and the ACF (Eq. A.34) vanishes.

Consequently, we can approximate the ACF at multiples of the refractory period by calculating the probability of finding a burst of n spikes (Eq. A.28):

$$\mathcal{C}(S, S, n\tau_{\text{ref}}) \approx \sum_{k=1}^{\infty} P_{k+1}\delta\big([n-k]\tau_{\text{ref}}\big)\,, \tag{A.37}$$

and for the special case of $\tau^{\text{syn}} \ll \tau_{\text{ref}}$

$$\mathcal{C}(S, S, n\tau_{\text{ref}}) \approx \sum_{k=1}^{\infty} \bar{p}^k \delta\big([n-k]\tau_{\text{ref}}\big) \tag{A.38}$$

$$= \sum_{k=1}^{\infty} e^{k \ln \bar{p}} \delta\big([n-k]\tau_{\text{ref}}\big)\,. \tag{A.39}$$

Hence, since increasing the mean rate (or bias) of the neuron leads to an increase in $\bar{p}$ and thus to a reduced decay constant $\ln \bar{p}$, more significant side-peaks emerge.

For $\tau^{\text{syn}} \approx \tau_{\text{ref}}$, the effective membrane distribution is not yet stationary and therefore, this approximation does not hold. To arrive at the exact solution, one would have to repeat the above calculation for all possible spike time combinations, leading to a recursive integral (*Gerstner and Kistler*, 2002). Furthermore, one would also need to take into account the situation where the first spike is itself part of a burst, i.e., is not the first spike in the burst. To circumvent a more tedious calculation, we use an approximation which is in between the two cases $\tau^{\text{syn}} \ll \tau_{\text{ref}}$ and $\tau^{\text{syn}} \approx \tau_{\text{ref}}$: we use $\bar{p} = \int_{\vartheta}^{\infty} \mathrm{d}u f(u, \tau_{\text{ref}} \mid \vartheta)$, which provides a reasonable approximation for short bursts.

A.1.5 Cross-correlation of free membrane potentials receiving correlated input

Similarly to the ACF of the membrane potential, one can calculate the cross-correlation function of the FMPs of two neurons receiving correlated noise input. First, the membrane potentials are given by

$$u_1 = u_0^1 + S_{\text{tot},1} * \kappa\,, \tag{A.40}$$

$$u_2 = u_0^2 + S_{\text{tot},2} * \kappa\,. \tag{A.41}$$

The covariance function can be written as

$$\begin{aligned}
&\langle \bar{u}_1(t)\bar{u}_2(t+\Delta)\rangle \\
&= \langle u_1(t)u_2(t+\Delta)\rangle - \langle u_1(t)\rangle \langle u_2(t)\rangle && \text{(A.42a)}\\
&= \langle \big(S_{\mathrm{tot},1} * \kappa\big)(t)\big(S_{\mathrm{tot},2} * \kappa\big)(t+\Delta)\rangle \\
&\quad - \langle \big(S_{\mathrm{tot},1} * \kappa\big)(t)\rangle \langle \big(S_{\mathrm{tot},2} * \kappa\big)(t)\rangle && \text{(A.42b)}\\
&= ... \\
&= \big(\lim_{T\to\infty} \frac{1}{T} \langle \bar{S}_{\mathrm{tot},1}(t)\bar{S}_{\mathrm{tot},2}(t+\Delta')\rangle_T \\
&\quad * \langle \kappa(t)\kappa(t+\Delta')\rangle_\infty \big)(\Delta)\,, && \text{(A.42c)}
\end{aligned}$$

with $\bar{u} = u - \langle u\rangle$, from which we obtain the cross-correlation function by normalizing with the product of standard deviations of u_1 and u_2 (for notation, see Eq. A.12). The term containing the input correlation coefficient is $\langle \bar{S}_{\mathrm{tot},1}(t)\bar{S}_{\mathrm{tot},2}(t+\Delta')\rangle$. Plugging in the spike trains, we get four cross-correlation terms

$$\begin{aligned}
&\langle \bar{S}_{\mathrm{tot},1}(t)\bar{S}_{\mathrm{tot},2}(t+\Delta')\rangle \\
&= \sum_{l,m\in\{\mathrm{e,i}\}} \Lambda_{l,1}\Lambda_{m,2} \langle \bar{S}_{l,1}(t)\bar{S}_{m,2}(t+\Delta')\rangle\ . && \text{(A.43)}
\end{aligned}$$

Since excitatory as well as inhibitory noise inputs are randomly drawn from the same pool of neurons, we can assume that $\langle \bar{S}_{l,1}(t)\bar{S}_{m,2}(t+\Delta')\rangle$ is approximately equal for all combinations of synapse types when averaging over enough inputs, regardless of the underlying correlation structure/distribution of the noise pool. The first term, however, depends on the synapse types since the Λ-terms (Eq. A.5b) contain the distance between reversal potentials and mean FMP:

$$\begin{aligned}
&\langle \bar{u}_1(t)\bar{u}_2(t+\Delta)\rangle \\
&= \zeta_1\zeta_2 \sum_{l,m\in\{\mathrm{e,i}\}} w_l w_m \big(E_l^{\mathrm{rev}} - \mu_1\big)\big(E_m^{\mathrm{rev}} - \mu_2\big) \\
&\cdot \Big[\langle \bar{S}_{l,1}(t)\bar{S}_{m,2}(t+\Delta')\rangle * \langle \kappa(t)\kappa(t+\Delta')\rangle \Big](\Delta)\,, && \text{(A.44)}
\end{aligned}$$

with constants $\zeta_i = \frac{\tau^{\mathrm{syn}}}{\langle g^{\mathrm{tot}}{}_i\rangle}\big(\tau^{\mathrm{syn}} - \langle \tau_{\mathrm{eff}}{}^i\rangle\big)$. The cross-correlation vanishes when, after summing over many inputs, the following identities hold:

$$\begin{aligned}
\langle \Lambda_{\mathrm{e},1}\Lambda_{\mathrm{e},2}\rangle_{\mathrm{inputs}} &= -\langle \Lambda_{\mathrm{e},1}\Lambda_{\mathrm{i},2}\rangle_{\mathrm{inputs}}\ , && \text{(A.45a)}\\
\langle \Lambda_{\mathrm{i},1}\Lambda_{\mathrm{i},2}\rangle_{\mathrm{inputs}} &= -\langle \Lambda_{\mathrm{i},1}\Lambda_{\mathrm{e},2}\rangle_{\mathrm{inputs}}\ , && \text{(A.45b)}
\end{aligned}$$

where $\langle(\cdot)\rangle$ is an average over all inputs, i.e., all neurons that provide noise.

While not relevant for our simulations, it is worth noting that the excitatory and inhibitory weights with which each neuron contributes its spike trains can be randomly drawn from non-identical distributions. By enforcing the following correlation between the noise weights

of both neurons, one can introduce a skew into the weight distribution which compensates for the differing distance to the reversal potentials:

$$\begin{aligned}&(E_{\text{rev}}^{\text{e},1}-\mu_1)(E_{\text{rev}}^{\text{e},2}-\mu_2)\left\langle w_{\text{e}}^1 w_{\text{e}}^2\right\rangle_{\text{inputs}}\\&=-(E_{\text{rev}}^{\text{e},1}-\mu_1)(E_{\text{rev}}^{\text{i},2}-\mu_2)\left\langle w_{\text{e}}^1 w_{\text{i}}^2\right\rangle_{\text{inputs}}\end{aligned}\tag{A.46}$$

A simple procedure to accomplish this is the following: First, we draw the absolute weights w^1 and w^2 from an arbitrary distribution and assign synapse types randomly with probabilities $p_{\text{e/i}}$ afterwards. If w^2 is excitatory, we multiply w^1 by $\frac{|E_{\text{rev}}^{i,2}-\mu_2|}{p_e|E_{\text{rev}}^{i,2}-\mu_2|+p_i|E_{\text{rev}}^{e,2}-\mu_2|}$, otherwise by $\frac{|E_{\text{rev}}^{e,2}-\mu_2|}{p_e|E_{\text{rev}}^{i,2}-\mu_2|+p_i|E_{\text{rev}}^{e,2}-\mu_2|}$. This way, $\langle w^1\rangle$ remains unchanged and the resulting weights suffice Eq. A.46.

A.1.6 State space switch from {0,1} to {-1,1}

To switch from the state space $\boldsymbol{z}\in\{0,1\}$ to $\boldsymbol{z}'\in\{-1,1\}$ while conserving the state probabilities (i.e., $p(\boldsymbol{z})=p(\boldsymbol{z}')$) one has to adequately transform the distribution parameters $\boldsymbol{W}$ and $\boldsymbol{b}$. Since the distributions are of the form $p(\boldsymbol{z})=\exp\left(\boldsymbol{z}^T\boldsymbol{W}\boldsymbol{z}+\boldsymbol{z}^T\boldsymbol{b}\right)$, this is equivalent to requiring that the energy $E(\boldsymbol{z})=\boldsymbol{z}^T\boldsymbol{W}\boldsymbol{z}+\boldsymbol{z}^T\boldsymbol{b}$ of each state remains, up to a constant, unchanged.

First, we can write the energy of a state $\boldsymbol{z}'$ and use the transformation $\boldsymbol{z}'=2\boldsymbol{z}-1$ to get

$$E(\boldsymbol{z}')=\frac{1}{2}\sum_{i,j}z_i'W_{ij}'z_j'+\sum_i z_i'b_i'\tag{A.47a}$$

$$\begin{aligned}&=\frac{1}{2}\Bigg(4\sum_{i,j}z_iW_{ij}'z_j-2\sum_{i,j}z_iW_{ij}'-2\sum_{i,j}W_{ij}'z_j\\&\quad+\sum_{i,j}W_{ij}'\Bigg)-\sum_i b_i'+2\sum_i z_ib_i'\end{aligned}\tag{A.47b}$$

$$\begin{aligned}&=\frac{1}{2}\sum_{i,j}z_i4W_{ij}'z_j+\sum_i z_i\big(2b_i'\\&\quad-2\sum_j W_{ij}'\big)+C\,,\end{aligned}\tag{A.47c}$$

where C is a constant $C=\frac{1}{2}\sum_{i,j}W_{ij}'-\sum_i b_i'$ and we used the fact that W_{ij}' is symmetric. Since constant terms in the energy leave the probability distribution invariant, we can simply compare $E(\boldsymbol{z}')$ and $E(\boldsymbol{z})$

$$E(\boldsymbol{z})=\frac{1}{2}\sum_{i,j}z_i^TW_{ij}z_j+\sum_i z_i^Tb_i\,,\tag{A.48}$$

and extract the correct parameter transformation:

$$W_{ij}=4W_{ij}'\,,\tag{A.49}$$

$$b_i=2b_i'-2\sum_j W_{ij}'\,.\tag{A.50}$$

From this, we can also calculate the inverse transformation rule for $\boldsymbol{z} = \frac{1}{2}(\boldsymbol{z}' + 1)$:

$$W'_{ij} = \frac{1}{4} W_{ij} \,, \tag{A.51}$$

$$b'_i = \frac{1}{2} b_i + \frac{1}{4} \sum_j W_{ij} \,. \tag{A.52}$$

A.1.7 Translation from Boltzmann to neurosynaptic parameters

Following *Petrovici, Bill, Bytschok, Schemmel, and Meier* (2016), the activation function of LIF neurons in the HCS is approximately logistic and can be written as

$$\begin{aligned} p(\boldsymbol{z}_k = 1 \mid \boldsymbol{z}_{/k}) &= \varphi(\mu) \\ &= \left(1 + \exp\left(-(\mu - u_0)/\alpha\right)\right)^{-1} \,, \end{aligned} \tag{A.53}$$

where $\boldsymbol{z}_{/k}$ is the state vector of all other neurons except the k'th one and μ the mean membrane potential (Eq. A.2b). u_0 and α are the inflection point and the inverse slope, respectively. Furthermore, the conditional probability $p(\boldsymbol{z}_k = 1 \mid \boldsymbol{z}_{/k})$ of a Boltzmann distribution over binary random variables $\boldsymbol{z}_k$, i.e., $p(\boldsymbol{z}) \propto \exp\left(\frac{1}{2}\boldsymbol{z}^T \boldsymbol{W} \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{b}\right)$, is given by

$$\begin{aligned} &p(\boldsymbol{z}_k = 1 \mid \boldsymbol{z}_{/k}) \\ &= \left(1 + \exp\left(-\sum_j W_{kj} \boldsymbol{z}_j - b_k\right)\right)^{-1} \,, \end{aligned} \tag{A.54}$$

with symmetric weight matrix $\boldsymbol{W}$, $W_{ii} = 0 \;\forall i$, and biases $\boldsymbol{b}$. These equations allow a translation from the parameters of a Boltzmann distribution (b_i, W_{ij}) to parameters of LIF neurons and their synapses (E_l, w_{ij}), such that the state dynamic of the network approximates sampling from the target Boltzmann distribution.

First, the biases $\boldsymbol{b}$ can be mapped to leak potentials E_l (or external currents) by requiring that, for $\boldsymbol{W} = 0$ (that is, no synaptic input from other neurons), the activity of each neuron equals the conditional probability of the target Boltzmann distribution

$$\left(1 + \exp\left(-(\mu - u_0)/\alpha\right)\right)^{-1} \stackrel{!}{=} \left(1 + \exp\left(-b_k\right)\right)^{-1} \,, \tag{A.55}$$

leading to the translation rule

$$\boldsymbol{E}_\mathrm{l} = \frac{\tau_\mathrm{m}}{\tau_\mathrm{eff}}(\alpha \boldsymbol{b} + u_0) - \sum_{x \in \{\mathrm{e,i}\}} \frac{\langle g_x^\mathrm{syn} \rangle}{g_\mathrm{l}} E_x^\mathrm{rev} \,. \tag{A.56}$$

To map Boltzmann weights W_{ij} to synaptic weights w_{ij}, we first have to rescale the W_{ij}, as done for the biases in Eq. A.56. However, leaky integrator neurons have non-rectangular PSPs, so their interaction strength is modulated over time. This is different from the interaction in Boltzmann machines, where the PSP shape is rectangular. Nevertheless, we can derive

a heuristic translation rule by requiring that the mean interaction during the refractory period of the presynaptic neuron is the same in both cases, i.e.,

$$\int_0^{\tau_{\text{ref}}} \mathrm{d}t\ PSP(t) \stackrel{!}{=} \int_0^{\tau_{\text{ref}}} \mathrm{d}t\ \alpha W_{ij} \tag{A.57a}$$

$$= \alpha W_{ij} \tau_{\text{ref}}\,, \tag{A.57b}$$

where $PSP(t)$ is given by Eq. A.4. From this, we get the translation rule for synaptic weights:

$$w_{kj} = \frac{\alpha W_{kj} C_{\text{m}} \frac{\tau_{\text{ref}}}{\tau^{\text{syn}}} \left(1 - \frac{\tau^{\text{syn}}}{\tau_{\text{eff}}}\right) \left(E_{kj}^{\text{rev}} - \mu\right)^{-1}}{\left[\tau^{\text{syn}} \left(e^{-\frac{\tau_{\text{ref}}}{\tau^{\text{syn}}}} - 1\right) - \tau_{\text{eff}} \left(e^{-\frac{\tau_{\text{ref}}}{\tau_{\text{eff}}}} - 1\right)\right]}\,. \tag{A.58}$$

A.2 Additional results

Animations of the experiments using hierarchical networks (Figs. 3.22 and 3.23) or neuromorphic hardware (Fig. 3.24) can be found online at https://doi.org/10.1016/j.neunet.2019.08.002 (the online version of *Dold, Bytschok, Kungl, Baumbach, Breitwieser, Senn, Schemmel, Meier, and Petrovici* 2019a) in the supplementary material. The captions of the online videos are given in the following subsections (taken from the above source).

A.2.1 Video S1: ensemble sampling on BrainScaleS-1

Video showing a single sampling run of an ensemble on BrainScaleS-1 (Fig. 3.24). Sampled distribution over time from an autonomous ensemble (no explicit noise) of 15 4-neuron networks on an artificial neural substrate (the BrainScaleS-1 system). (top) Median D_{KL} of the ensemble (red) and the individual networks (red, transparent) as a function of time after training. The median D_{KL} pre-training is shown in black. (Bottom) Comparison between target distribution (blue) and sampled distribution (red) for all networks. Most networks are able to approximate their target distribution well (e.g., the networks at position (1,1), (4,1) and (4,2) with (*row*, *column*)) or at least approximate the general shape of the target distribution (e.g., (0,1) and (0,2)). The networks at position (2,2) and (3,0) show strong deviations from their respective target distributions due to single neuron deficiencies. Because of the speed-up of the BrainScaleS-1 system, it only takes 100ms to emulate 10^6ms of biological time.

A.2.2 Video S2: ensemble of networks dreaming of MNIST

Video of the data shown in Fig. 3.22. An ensemble of five hierarchical networks with 784-500-10 (visible-hidden-label) neurons trained on the MNIST handwritten data set generating digits without explicit noise sources is shown (in simulation). Every network in the ensemble receives the spiking activity from hidden neurons of the other networks as stochastic input

only. (top) Activity of the hidden area of each network. (middle) Class label currently predicted by the label neurons. (Bottom) Activity of the visible area. To generate gray scale images from spikes, we averaged the spiking activity of each neuron over a window of size ±90ms.

A.2.3 Video S3: single-network ensemble dreaming of MNIST

Video of the data shown in Fig. 3.23. A single hierarchical network with 784-200 (visible-hidden) neurons generating samples of the MNIST handwritten digits data set without explicit noise sources is shown (in simulation). To initialize the network, we trained a Boltzmann machine and translated the weights and biases to neurosynaptic parameters to reduce simulation time. (top) Illustration of the used network architecture. Lateral (non-plastic) connections in each area were utilized as a noise source (red), with an interconnectivity of $\epsilon = 0.2$. (bottom) Averaged activity (average window ±90ms) of the visible area (left) after initializing the network, (middle) after further training the network and (right) for the case of explicit Poisson noise instead of lateral interconnections. After initialization, the network is able to generate recognizable images but does not mix well between different digit classes. Further training the network on images of the MNIST training set improves both image quality and mixing, rivaling the quality of the reference setup with explicit Poisson noise. During the second training phase, neurosynaptic parameters are adjusted such that every neuron is able to perform its task with the available background activity it receives.

A.3 Simulation details

The following simulation details for Figs. 3.10 to 3.12 are adapted from *Zenk* (2018) and the remaining ones are taken from the supplemental material of *Dold, Bytschok, Kungl, Baumbach, Breitwieser, Senn, Schemmel, Meier, and Petrovici* (2019a), written by me. In all simulations, the model was integrated with a time step of $\Delta t = 0.1$ms.

A.3.1 Figs. 3.10 to 3.12: spatio-temporal predictions

For these experiments, we used CuBa LIF neurons with neuron parameters given in Table A.1. The initial networks were trained on the trajectory data sets with the hyperparameters given in Table A.2. All remaining experimental details are summarized in the main text.

C_m	0.2 nF	membrane capacitance
τ_m	0.1 ms	membrane time constant
E_l	-50.01 mV	leak potential
ϑ	-50.0 mV	threshold potential
ϱ	-50.01 mV	reset potential
$\tau_\mathrm{e}^\mathrm{syn}$	10.0 ms	exc. synaptic time constant
$\tau_\mathrm{i}^\mathrm{syn}$	10.0 ms	inh. synaptic time constant
τ_ref	10.0 ms	refractory time
$w_\mathrm{e}^\mathrm{noise}$	0.001 μA	exc. Poisson weights
$w_\mathrm{i}^\mathrm{noise}$	-0.001 μA	inh. Poisson weights
$\nu_\mathrm{i/e}^\mathrm{noise}$	400 Hz	Poisson noise frequency

Table A.1.: Neuron parameters used in simulations, along with background noise specifics.

Parameter	flat	Gaussian
number hidden neurons n_h	500	500
batch size	50	50
number epochs	35	40
number wake-sleep steps	15	10
initial learning rate η_0	0.05	0.05
momentum scaling α	0.5	0.5
regularization λ	10^{-4}	10^{-3}

Table A.2.: Hyperparameters used for learning the flat and Gaussian potential data sets.

A.3.2 Figs. 3.15, 3.16 and 3.18 to 3.23: general details

For these experiments, we used CoBa LIF neurons. Since spike-based stochastic networks are time-continuous systems, we could, in principle, retrieve a sample at every integration step. However, as individual neurons only change their state on the time scale of refractory periods, and hence new states emerge on a similar time scale, we read out the states in intervals of $\frac{\tau_\mathrm{ref}}{2}$. If not stated otherwise, we used $U_\mathrm{SE} = 1.0$ and $\tau_\mathrm{rec} = 10$ms as short term plasticity parameters for connections within each spiking network to ensure PSPs with equal height, as discussed in *Petrovici* (2016). For background connections, i.e., Poisson input or background input coming from other spiking networks in an ensemble, we use static synapses ($U_\mathrm{SE} = 1.0$ and $\tau_\mathrm{rec} \to 0$) instead to facilitate the mathematical analysis. For the interconnections of an ensemble, we expect that short-term depression will not alter the performance of individual spiking networks in a drastic way, as the effect will be rather small on average if most neurons are far away from tonic bursting. Thus, to allow a clear comparability to spiking networks receiving Poisson input, we chose to neglect short-term depression for ensemble interconnections. The neuron parameters used throughout all simulations are listed in Table A.3.

C_m	0.1 nF	membrane capacitance
τ_m	1.0 ms	membrane time constant
E_l	-65.0 mV	leak potential
$E_\mathrm{e}^\mathrm{rev}$	0.0 mV	exc. reversal potential
$E_\mathrm{i}^\mathrm{rev}$	-90.0 mV	inh. reversal potential
ϑ	-52.0 mV	threshold potential
ϱ	-53.0 mV	reset potential
$\tau_\mathrm{e}^\mathrm{syn}$	10.0 ms	exc. synaptic time constant
$\tau_\mathrm{i}^\mathrm{syn}$	10.0 ms	inh. synaptic time constant
τ_ref	10.0 ms	refractory time
$w_\mathrm{e}^\mathrm{noise}$	0.001 μS	exc. Poisson weights
$w_\mathrm{i}^\mathrm{noise}$	-0.00135 μS	inh. Poisson weights

Table A.3.: Neuron parameters used in simulations. The membrane time constant was chosen small such that smaller noise frequencies suffice to reach a high-conductance state, allowing us to use smaller ensembles and hence reduce simulation time.

A.3.3 Fig. 3.15: autocorrelations of ensemble background

The parameters for the target distributions of all networks were randomly drawn from beta distributions, i.e., $W \sim 2 \cdot (\mathrm{beta}(0.5, 0.5) - 0.5)$ and $b \sim 1.2 \cdot (\mathrm{beta}(0.5, 0.5) - 0.5)$. The bias of each background-providing neuron was adjusted to yield the desired firing rate $p \in \{0.1, 0.6, 0.9\}$ in case of no synaptic input ($W = 0$) $b = \log\left(\frac{p}{1-p}\right)$. For the different activity cases, we used $N_{0.1} = 260$, $N_{0.6} = 50$ and $N_{0.9} = 34$ networks as background input for each neuron to reach the desired background noise frequency. The Poisson frequency of the noise-providing networks was set to 3000Hz. Activation functions were recorded by providing every neuron with background noise for $5 \cdot 10^5$ms and varying its leak potential. For the ACFs, we first merged all individual noise spike trains and binned the resulting spike train with a bin size of 0.5ms before calculating the ACF.

A.3.4 Fig. 3.16: cross-correlations of ensemble background

Background input was generated from a pool of pairwise connected neurons (i.e., small subnetworks with two neurons each) with strong positive or negative weights to yield highly positively or negatively correlated spike trains. Each pair of neurons in the main network (i.e, the network receiving no Poisson input) received the spikes of 80 such subnetworks as background input. The weights of the noise-generating subnetworks were drawn from beta distributions $w_{ij}^\mathrm{pre} \sim 4 \cdot [\mathrm{beta}(5.0, 0.5) - 0.5]$ (distribution strongly skewed to positive weights) or $w_{ij}^\mathrm{pre} \sim 4 \cdot [\mathrm{beta}(0.5, 5.0) - 0.5]$ (skewed to negative weights). The parameters of the main network were randomly generated as in Appendix A.3.3. The absolute values of the weights W_noise projecting from the noise-generating subnetworks to the main network were randomly generated from a (Gaussian-like) beta distribution $W_\mathrm{noise} \sim (\mathrm{beta}(4.0, 4.0) - 0.5) \cdot 2 \cdot 0.001 + 0.001\mu\mathrm{S}$. The synapse type of each weight W_noise was determined randomly with equal probability. Furthermore, inhibitory synapses were scaled by a factor of 1.35 such

that inhibitory and excitatory weights have the same mean value as for the simulations with Poisson noise (see Table A.3). For the three traces shown, the absolute value of each synaptic weight was drawn independently. Synapse types were either drawn according to a pattern (Fig. 3.16A, left and middle) or independently (Fig. 3.16A, right).

A.3.5 Figs. 3.18 and 3.19: calibration and plasticity scheme

For every network, the target distribution parameters were again drawn from a beta distribution as in Appendix A.3.3. The connectivity of the noise connections was set to $\epsilon = 0.05$, i.e., each neuron received the spikes of 5% of the remaining subnetworks' neurons as stochastic background input, for the ensemble of 400 3-neuron networks (no training, Fig. 3.18) and to $\epsilon = 0.10$ for the ensemble of 100 6-neuron networks (training, Fig. 3.19).

The training was done for subnetworks with 6 neurons each, where every subnetwork was initialized with different weights and biases than the target parameters, also generated randomly. As an initial guess for the neurons' activation functions, we used the activation function of a neuron receiving 2000Hz excitatory and inhibitory Poisson input, leading to a slope of $\alpha = 1.47\text{mV}$ and a mid-point at -52.97mV. These parameters were subsequently used to translate the weight and bias updates given by the Hebbian wake-sleep algorithm (see Eqs. 3.9 and 3.10) to updates of synaptic weights and leak potentials. The subnetworks were all trained simultaneously with wake-sleep, where the model term was approximated by sampling for $1 \cdot 10^5\text{ms}$. The training was done for 2000 steps and with a learning rate of $\frac{400}{t+2000}$. As a reference, 50 subnetworks receiving only Poisson noise (2000Hz) were also trained in the same way for 2000 steps.

Self-activation of the network can be observed when a large enough fraction of neurons have a suprathreshold rest potential, in our case around 30%.

A.3.6 Figs. 3.20 to 3.22: ensemble-based inference on (E)MNIST

To reduce the training time, we pre-trained classical restricted Boltzmann machines on their respective data sets, followed by direct translation to spiking network parameters. To obtain better generative models, we utilized the CAST algorithm (*Salakhutdinov*, 2010) which combines contrastive divergence with simulated tempering. Each subnetwork was trained for 200000 steps with a minibatch size of 100, a learning rate of $\frac{20}{t+2000}$, an inverse temperature range $\beta \in [1., 0.6]$ with 20 equidistant intervals and an adaptive factor $\gamma_t = \frac{9}{1+t}$. States between the fast and slow chain were exchanged every 50 samples. To collect the background statistics of these subnetworks, we first simulated all networks with stochastic Poisson input. To improve the Poisson-stimulated reference networks' mixing properties, we utilized short-term depression to allow an easier escape from local energy minima faster ($U_{\text{SE}} = 0.01$, $\tau_{\text{rec}} = 280\text{ms}$, global weight rescale $\delta W = 0.014^{-1}$). For classification, the gray scale value of image pixels was translated to spiking probabilities of the visible units, which can be adjusted by setting the biases as $\ln\left(\frac{\text{grey value}}{1-\text{grey value}}\right)$. Spike probabilities of 0 and 1 were mapped to biases of -50 and 50. Furthermore, during classification, the connections

projecting back from the hidden neurons to the visible neurons were silenced in order to prevent the hidden area from influencing the clamped input. For pattern rivalry, the non-occluded pixels were binarized. In total, each spiking network received background input from 20% of the other networks' hidden neurons. For the classification results, the experiment was repeated 10 times for different random seeds (leading to different connectivity matrices between the ensemble networks). For training and testing, we used 400 and 200 images per class. In Fig. 3.20D, consecutive images are 400ms apart. In Fig. 3.20E, for the clamped 'B', consecutive images are 2s apart, for the 'L' 1.5s.

The experiments with MNIST used an ensemble of five networks with 784 visible neurons and 500 hidden neurons each (Fig. 3.22), trained on $6 \cdot 10^3$ images per digit class (where we took the digits provided by the EMNIST set to have balanced classes). Since the generative properties of larger spiking networks depend heavily on the synaptic interaction, we also used short-term plasticity for the case without Poisson noise ($U_{\mathrm{SE}} = 0.01$, $\tau_{\mathrm{rec}} = 280\mathrm{ms}$, $\delta W = 0.01^{-1}$) to allow fluent mixing between digit classes (*Leng, Martel, Breitwieser, Bytschok, Senn, Schemmel, Meier, and Petrovici*, 2018). For MNIST, each spiking network received background input from 30% of the other networks' hidden neurons. Furthermore, the excitatory noise weight was set to $w_{\mathrm{e}}^{\mathrm{noise}} = 0.0009\mu\mathrm{S}$.

The network-generated images were obtained by averaging the firing activity of the visible neurons (for pattern rivalry ±90ms, for classification and dreaming of EMNIST ±80ms and for MNIST ±140ms). The time intervals were chosen to reduce the blur caused by mixing when plotting averaged spiking activity.

A.3.7 Fig. 3.23: single-network ensemble dreaming of MNIST

In this experiment, lateral (non-plastic) connections in each area were utilized as a noise source, with an interconnectivity of $\epsilon = 0.2$. The additional training was done using standard wake-sleep learning with batch size 100, learning rate $\frac{40}{t+2000}$ with t the number of updates, for 1000 training updates and a presentation time per training sample of 200ms.

A.3.8 Fig. 3.24: neuromorphic deterministic sampling

The emulated ensemble consists of 15 4-neuron networks which were randomly initialized on two HICANN chips (HICANN 367 and 376 on Wafer 33 of the BrainScaleS–1 system). The analog hardware parameters which determine the physical range of weights adjustable by the 4bit setting were set to $gmax = 500$ and $gmax\_div = 1$. Given the current state of development of the BrainScaleS–1 system and its surrounding software, we limited the experiment to small ensembles in order to avoid potential communication bottlenecks.

Biases were implemented by assigning every neuron a bias neuron, with its rest potential set above threshold to force continuous spiking. While a more resource-efficient implementation of biases is possible, this implementation allowed an easier mapping of neuron - bias pairs on the neuromorphic hardware. Bias strengths can then be adjusted by modifying the synaptic weights between neurons and their allocated bias neurons. The networks were trained with the wake-sleep learning rule to sample from their respective target distributions. Biases were randomly drawn from a normal distribution with $\mu = 0$ and $\sigma = 0.25$. The weight matrices

were randomly drawn from $W \propto 2 \cdot (\text{beta}(0.5, 0.5) - 0.5)$ and subsequently symmetrized by averaging with their respective transposes $0.5 \cdot (W + W^{\mathrm{T}})$.

Since the refractory periods of hardware neurons vary, we further measured the refractory period of every neuron in the ensemble, which was later used to calculate the binary neuron states from spike raster plots. Refractory periods were measured by setting biases to large enough values to drive neurons to their maximal firing rate. After running the experiment, the duration of the refractory period can be approximated by dividing the experiment time by the number of measured spikes.

During the whole experiment, the ensemble did not receive external Poisson noise. Instead, individual spiking networks received spikes from 20% of the remaining ensemble as background input, with noise connections having hardware weights of ± 4 with the sign chosen randomly with equal probability. Translation between theoretical and 4bit hardware parameters was done by clipping the values into the range $[-15, 15]$ and rounding to integer values. Calculation of weight and bias updates was performed on a host computer. The learning rate was set to $\eta = 1.0$ for all networks performing better than the median and twice this value for networks performing worse. For every training step, the ensemble was recorded for 10^5ms biological time before applying a parameter update.

For the experiments with Poisson noise, every neuron received 300Hz external Poisson noise provided by the host computer.

The neuron parameters used for all hardware experiments are listed in Table A.4.

	ensemble neurons	bias neurons
C_{m}	0.2 nF	
τ_{m}	7 ms	
E_{l}	-20.0 mV	60.0 mV
$E_{\mathrm{e}}^{\mathrm{rev}}$	60.0 mV	
$E_{\mathrm{i}}^{\mathrm{rev}}$	-100.0 mV	
ϑ	-20.0 mV	
ϱ	-35.0 mV	-30.0 mV
$\tau_{\mathrm{e}}^{\mathrm{syn}}$	8.0 ms	5.0 ms
$\tau_{\mathrm{i}}^{\mathrm{syn}}$	8.0 ms	5.0 ms
τ_{ref}	4.0 ms	1.5 ms

Table A.4.: Neuron parameters used for the implementations in an artificial neural substrate. Note that these are intended parameters and the realized ones can vary from neuron to neuron.

A.4 Implementation details

A.4.1 Simulation software

The simulations in Figs. 3.10 to 3.12 were done using Python 2.7, PyNN 0.8beta (*Davison, Brüderle, Eppler, Kremkow, Muller, Pecevski, Perrinet, and Yger*, 2009) and NEST 2.4.2 (*Gewaltig and Diesmann*, 2007) on CPUs of the BwForCluster NEMO, the HPC cluster

of the state Baden Württemberg (https://wiki.bwhpc.de/e/Category:BwForCluster_NEMO, 24.01.2020). All remaining simulations were done using PyNN 0.8 and NEST 2.4.2 as well, but on the private HPC cluster of the Electronic Vision(s) group (with special thanks to Eric Müller and Christian Mauch for maintaining the cluster). For all sampling-based simulations, a software package by Oliver Breitwieser (SBS - spike-based sampling) that builds upon NEST and PyNN – i.e., offering convenience functions to set up spike-based sampling networks – was used.

A.4.2 Emulation software

For the BrainScaleS-1 experiments, the following software modules were used: *spack_visionary-defaults/2017-01-26_0.2.11*, *nmpm_software/2017-12-11-spack-2017-01-26-1* and *reticleCtrl*.

B | Appendix of chapter 5

B.1 Calculations

B.1.1 Lookaheads arising from spike-generating mechanisms

The following calculation is adapted from Walter Senn. When we introduce the voltage dynamics by a first-order differential equation of the form $\tau\dot{u} = -u + I$ for some time-dependent current I and a time constant τ, the voltage u becomes delayed with respect to I. In the following, we demonstrate how biological neurons might be able to compensate (or even over-compensate) for that delay by the mechanism to generate action potentials. In the classical Hodgkin-Huxley (HH) model (*Hodgkin and Huxley*, 1952), an action potential arises from a voltage-dependent sodium conductance that can roughly be described as $g_{\mathrm{Na}}(u, \dot{u}) \approx m_\infty^{k+1}(u)\, h$, with an instantaneous activation function $m_\infty(u)$ and a delayed inactivation, $h = h_\infty(u) - \tau_h \dot{h}$, that follows $h_\infty(u)$ with some variable delay. As compared to the original HH model we absorbed the roughly constant driving force into the variables m_∞ and h_∞. We set $h_\infty(u) = m_\infty^{-k}(u)$, and to get smaller numbers for h_∞ we choose $k = 1$ (instead of 2). Moreover, given the abstractness of the theory, we suppress constants that do not affect the dynamics and hence the dynamic variables (u, r, e, I, and their low-pass filtered versions) have all units 1/time, while the synaptic strengths are unitless.

In our formalism, the instantaneous rate r of a neuron is identified with the sodium conductance, $r \approx g_{\mathrm{Na}}$, and we approximate $\dot{h} \approx \frac{\mathrm{d}}{\mathrm{d}t} h_\infty = h'_\infty(u)\, \dot{u}$ to be plugged into $h = h_\infty(u) - \tau_h \dot{h}$. This motivates our definition of the instantaneous rate $r = r(t)$ at any time t as

$$r = m_\infty^{k+1}(u)\, h(u, \dot{u})\,, \quad \text{with} \quad h(u, \dot{u}) = h_\infty(u) - \tau_h h'_\infty(u)\, \dot{u}\,. \tag{B.1}$$

Note that this rate could become negative, but if u does not vary too fast as compared to τ_h this does not occur. To relate this definition to the energy framework discussed in Section 5.2.4, we identify $\bar{r} = m_\infty(u) \equiv \bar{\rho}(u)$ (which might be argued to be rather dubious). With this, and with $h_\infty = m_\infty^{-k}$ and $-h'_\infty = k\, m_\infty^{-k-1} m'_\infty$ plugged into Eq. B.1, the instantaneous rate becomes

$$r = m_\infty^{k+1} h = m_\infty + \tau_h k\, m'_\infty \dot{u} = \bar{r} + \tau \bar{r}' \dot{u} = \bar{r} + \tau \dot{\bar{r}}\,, \tag{B.2}$$

with $\tau = \tau_h k$. For a slowly varying membrane potential we can approximate $r(t) \approx \bar{r}(t + \tau)$, expressing that r looks ahead of $\bar{r}$ in time.

B.1.2 Variation of weights leading to variation of potentials

The following idea is originally by Walter Senn, with additions by Walter Senn, Akos F. Kungl and me. The proof itself is not required to understand why and how the model works, but it gives an interesting new perspective on the problem of real-time learning in dynamical systems.

The goal of our thought experiment is to demonstrate that synaptic plasticity and neuronal membrane dynamics do not interfere with each other. To do so, we introduce the following idea: changes in the weights W in time can be seen as arbitrary variations of the weight trajectory $\delta W(t)$ that minimize the energy function E. Since the weights directly impact the membrane potential, a variation in W, δW, leads to variations in the membrane potential δu, or more specifically, the prospective potential $\delta\tilde{u}$. Such variations $\delta\tilde{u}$ should not influence the energy function over a certain time interval $\int E\mathrm{d}t$, since weight updates are derived from the energy function. Hence, in order for synaptic and neuronal dynamics not to interfere with each other, the prospective potential needs to follow a critical trajectory which leaves the action $A = \int E\mathrm{d}t$ invariant, i.e., $\delta A = 0$.

More formally, we want to show that $\dot{W}_l \propto -\lim_{\beta\to 0}\frac{1}{\beta}\frac{\partial E}{\partial W_l} = -\frac{\mathrm{d}C}{\mathrm{d}W_l}$ at all time, i.e., weight dynamics reduce the cost function continuously in time. The crucial property we have to show for this is that the total derivative of the energy function $\frac{\mathrm{d}E}{\mathrm{d}W}$ (and $\frac{\mathrm{d}E}{\mathrm{d}\beta}$) becomes equal to the partial derivatives $\frac{\partial E}{\partial W}$ (and $\frac{\partial E}{\partial \beta}$). To prove this statement we consider a small (continuous) variation δW in the synaptic weights that vanishes at the boundary t_1 (but not necessarily at t_2, which is a big difference to how the Euler-Lagrange equations are applied in physics[1]) and their induced variation in the prospective potentials, $\tilde{u} + \epsilon\delta\tilde{u}$ (Fig. B.1). Using the Taylor expansion in ϵ, the directional derivative of $E(\tilde{u}_W, \dot{\tilde{u}}_W, W)$ in the direction δW at a given point in time is $\frac{\mathrm{d}E}{\mathrm{d}W}\delta W = \frac{\mathrm{d}}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} E(\tilde{u}+\epsilon\delta\tilde{u},\, \dot{\tilde{u}}+\epsilon\delta\dot{\tilde{u}},\, W+\epsilon\delta W) = \frac{\partial E}{\partial\tilde{u}}\delta\tilde{u} + \frac{\partial E}{\partial\dot{\tilde{u}}}\delta\dot{\tilde{u}} + \frac{\partial E}{\partial W}\delta W$. With this we calculate

$$\int_{t_1}^{t_2}\frac{\mathrm{d}E}{\mathrm{d}W}\,\delta W\mathrm{d}t = \frac{\mathrm{d}}{\mathrm{d}\epsilon}\Big|_{\epsilon=0}\int_{t_1}^{t_2}\mathrm{d}t\, E(\tilde{u}+\epsilon\delta\tilde{u},\, \dot{\tilde{u}}+\epsilon\delta\dot{\tilde{u}},\, W+\epsilon\delta W) \tag{B.3a}$$

$$= \int_{t_1}^{t_2}\mathrm{d}t\left(\frac{\partial E}{\partial\tilde{u}}\delta\tilde{u} + \frac{\partial E}{\partial\dot{\tilde{u}}}\delta\dot{\tilde{u}} + \frac{\partial E}{\partial W}\delta W\right) \tag{B.3b}$$

$$= \int_{t_1}^{t_2}\mathrm{d}t\left(\left(\frac{\partial E}{\partial\tilde{u}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial E}{\partial\dot{\tilde{u}}}\right)\delta\tilde{u} + \frac{\partial E}{\partial W}\delta W\right) + \frac{\partial E}{\partial\dot{\tilde{u}}}\delta\tilde{u}\Big|_{t_1}^{t_2} \tag{B.3c}$$

$$= \int_{t_1}^{t_2}\frac{\partial E}{\partial W}\delta W\mathrm{d}t\,. \tag{B.3d}$$

The second last equality uses the partial integration trick of the calculus of variation and the last equality uses our assumption that the variational derivative actually vanishes for the trajectories $\tilde{u}$, $\frac{\delta E}{\delta\tilde{u}} = \frac{\partial E}{\partial\tilde{u}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial E}{\partial\dot{\tilde{u}}} = 0$. Moreover, the term $\frac{\partial E}{\partial\dot{\tilde{u}}}\delta\tilde{u}\Big|_{t_1}^{t_2}$ vanishes for two reasons. First, at the initial time t_1 the voltage variation $\delta\tilde{u}$ vanishes by definition, $\delta\tilde{u}(t_1) = 0$,

[1] Since changing the weights ($\delta W(t') \neq 0$ with $t_1 < t' < t_2$) will always affect $\tilde{u}(t_2)$, making it impossible to impose $\delta\tilde{u}(t_2) = 0$ through weight variations.

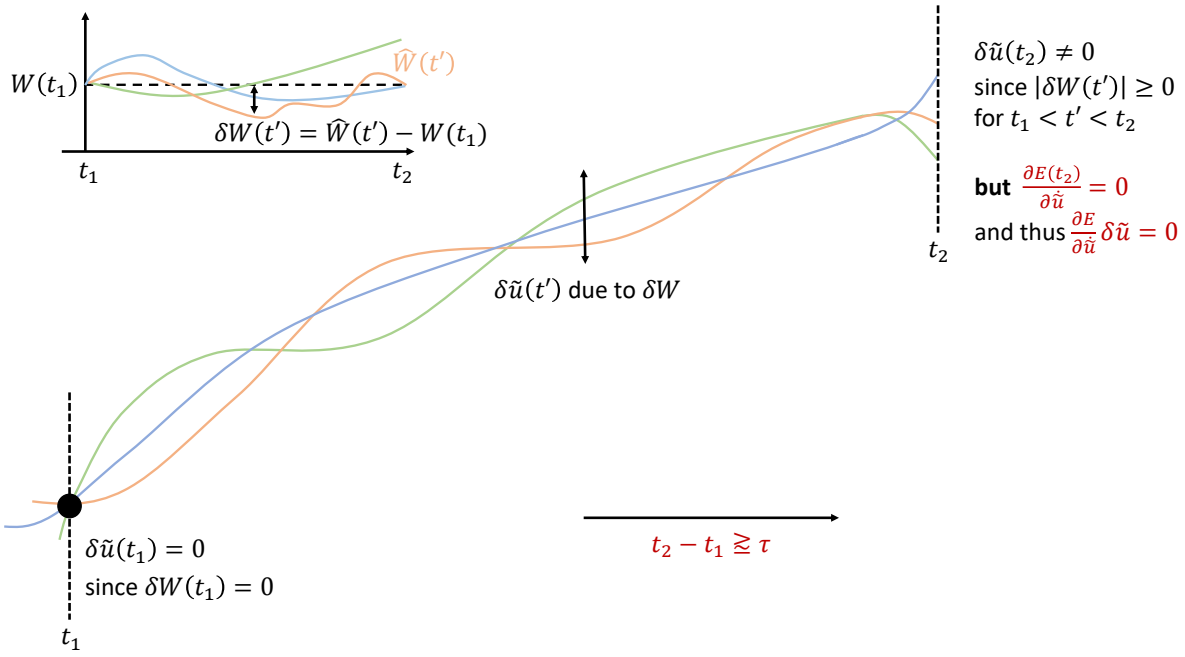

Figure B.1.: Illustration of the main idea behind the neuronal least action principle. Variations in the weights δW (top left) lead to variations in the prospective potentials $\delta\tilde{u}$. However, different from physics, by introducing the variation of $\tilde{u}$ this way, we cannot enforce it to vanish at time t_2 (right). Thus, for neuronal and synaptic dynamics to act simultaneously, it is crucial to couple the least action principle with the prospective voltage, since this way, the boundary term at t_2 vanishes due to the vanishing partial derivative $\frac{\partial E}{\partial \dot{\tilde{u}}}$. At t_1, $\delta\tilde{u}(t_1)$ vanishes since $\delta W(t_1) = 0$.

since the weight variation itself vanishes, $\delta\tilde{W}(t_1) = 0$. Second, assuming that the interval is considerably longer than the time constant, $t_2 \gg t_1 + \tau$, we conclude that $\frac{\partial E}{\partial \dot{\tilde{u}}}(t_2) = 0$. This latter equality is derived from the fact that, due to $\tilde{u} - \tau\dot{\tilde{u}} = u$, the Euler-Lagrange equations in $\tilde{u}$, $\frac{\partial E}{\partial \tilde{u}} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial E}{\partial \dot{\tilde{u}}} = 0$, translate to $\frac{\partial E}{\partial u} + \tau\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial E}{\partial u} = 0$, with unique solution $\frac{\partial E}{\partial u}(t) = ce^{-(t-t_1)/\tau}$ for $t \geq t_1$ and $c = \frac{\partial E}{\partial u}(t_1)$ (see also Fig. B.1). From Eq. (B.3d) we now conclude that

$$\int_{t_1}^{t_2} \mathrm{d}t \left(\frac{\mathrm{d}E}{\mathrm{d}W} - \frac{\partial E}{\partial W} \right) \delta W = 0 \,, \tag{B.4}$$

and because δW is an arbitrary variation in the given time interval $[t_1, t_2]$, we conclude that $\frac{\mathrm{d}E}{\mathrm{d}W} = \frac{\partial E}{\partial W}$ throughout this interval. We can formally write

$$\frac{\mathrm{d}}{\mathrm{d}W} E(\tilde{u}^\beta, \dot{\tilde{u}}^\beta, W, \beta) = \frac{\delta E}{\delta \tilde{u}^\beta}\frac{\mathrm{d}\tilde{u}^\beta}{\mathrm{d}W} + \frac{\partial E}{\partial W} = \frac{\partial}{\partial W} E(\tilde{u}^\beta, \dot{\tilde{u}}^\beta, W, \beta) \,, \tag{B.5}$$

if the system is far away from initialization. Here, we added an upper index β to indicate whether nudging is on or off ($\beta \neq 0$ or $\beta = 0$). Similarly, we get $\frac{\mathrm{d}}{\mathrm{d}\beta} E(\tilde{u}^\beta, \dot{\tilde{u}}^\beta, W, \beta) = \frac{\partial}{\partial \beta} E(\tilde{u}^\beta, \dot{\tilde{u}}^\beta, W, \beta)$. For differentiable functions the total derivatives commute, $\frac{\mathrm{d}}{\mathrm{d}W}\frac{\mathrm{d}}{\mathrm{d}\beta} E = \frac{\mathrm{d}}{\mathrm{d}\beta}\frac{\mathrm{d}}{\mathrm{d}W} E$, and based on the above statement we can replace the inner total derivatives by the partial derivatives while evaluating the derivatives with respect to β at $\beta = 0$,

$$\frac{\mathrm{d}}{\mathrm{d}W}\frac{\partial}{\partial \beta} E(\tilde{u}^\beta, \dot{\tilde{u}}^\beta, W, \beta)\Big|_{\beta=0} = \frac{\mathrm{d}}{\mathrm{d}\beta}\frac{\partial}{\partial W} E(\tilde{u}^\beta, \dot{\tilde{u}}^\beta, W, \beta)\Big|_{\beta=0} \,. \tag{B.6}$$

Since β enters linearly into the energy function (Eq. 5.9), the inner partial derivative on the left-hand side of Eq. B.6 is just the cost function C,

$$\frac{\partial}{\partial\beta}E(\tilde{u}^\beta,\dot{\tilde{u}}^\beta,W,\beta)\Big|_{\beta=0}=\frac{\partial}{\partial\beta}\Big|_{\beta=0}(V^\beta+\beta C^\beta)=C\big(u_N^0,u_N^{0,\text{target}}\big)\,. \tag{B.7}$$

We now plug Eq. B.7 into Eq. B.6 and use that the inner partial derivative on the right-hand side of Eq. B.6 is the plasticity rule $\dot{W}_l$ defined in Eq. 5.10 (setting $\eta=1$). Hence, we have for $l=1...N$,

$$-\frac{\mathrm{d}}{\mathrm{d}W_l}C\big(u_N^0,u_N^{0,\text{target}}\big)\;=-\frac{\mathrm{d}}{\mathrm{d}\beta}\frac{\partial}{\partial W_l}E(\tilde{u}^\beta,\dot{\tilde{u}}^\beta,W,\beta)\Big|_{\beta=0}=\lim_{\beta\to0}\frac{1}{\beta}\big(u_l^\beta-W_l\,\bar{r}^\beta\big)\,\bar{r}^{\beta\,\mathrm{T}}\,, \tag{B.8}$$

where we used that the energy is zero for $\beta=0$, and hence prediction errors vanish throughout the network. Note that the cost function is evaluated for trajectories without nudging ($\beta=0$), indicating that the derived equation is only valid if with nudging, the obtained trajectories do not differ too much from the nudging-free case, which is approximately true for small $\beta>0$.

B.1.3 Formulation as 5-compartment neuron model

To arrive at the 5-compartment representation, we rescale the energy function by $\gamma=1+\frac{g_\epsilon}{g_\mathrm{l}}$, i.e.,

$$c_\mathrm{m}\dot{u}_l=-\gamma\nabla_{u_l}E\,, \tag{B.9}$$

which results in

$$\begin{aligned}
c_\mathrm{m}\dot{u}_l &= -\gamma\,g_\mathrm{l}\,u_l && \text{(B.10)}\\
&\quad+\gamma\,g_\mathrm{l}\,\lambda W_l\bar{r}_{l-1} && \text{(B.11)}\\
&\quad+\gamma\,g_\mathrm{l}\,\lambda\,\bar{r}_l'\odot W_{l+1}^\mathrm{T}(u_{l+1}-W_{l+1}\bar{r}_l) && \text{(B.12)}\\
&\quad+\gamma\,g_\mathrm{l}\,(1-\lambda)\,G_l\bar{r}_{l+1} && \text{(B.13)}\\
&\quad+\gamma\,g_\mathrm{l}\,(1-\lambda)\,\bar{r}_l'\odot G_{l-1}^\mathrm{T}(u_{l-1}-G_{l-1}\bar{r}_l)\,. && \text{(B.14)}
\end{aligned}$$

The first term can be split up to

$$\gamma\,g_\mathrm{l}\,u_l=g_\epsilon\,u_l+g_\mathrm{l}\,u_l\,, \tag{B.15}$$

while the prediction error terms are rewritten as follows

$$\begin{aligned}
\gamma\,g_\mathrm{l}\,\bar{r}_l'\odot W_{l+1}^\mathrm{T}(u_{l+1}-W_{l+1}\bar{r}_l) &= g_\epsilon\,\gamma\,\frac{g_\mathrm{l}}{g_\epsilon}\,\bar{r}_l'\odot W_{l+1}^\mathrm{T}(u_{l+1}-W_{l+1}\bar{r}_l) && \text{(B.16)}\\
&= g_\epsilon\,\gamma\bar{e}_l^W\,, && \text{(B.17)}\\
\gamma\,g_\mathrm{l}\,\bar{r}_l'\odot G_{l-1}^\mathrm{T}(u_{l-1}-G_{l-1}\bar{r}_l) &= g_\epsilon\,\gamma\,\frac{g_\mathrm{l}}{g_\epsilon}\,\bar{r}_l'\odot G_{l-1}^\mathrm{T}(u_{l-1}-G_{l-1}\bar{r}_l) && \text{(B.18)}\\
&= g_\epsilon\,\gamma\bar{e}_l^G\,. && \text{(B.19)}
\end{aligned}$$

with

$$\bar{e}_l^W = \frac{g_\mathrm{l}}{g_\epsilon}\,\bar{r}_l' \odot W_{l+1}^\mathrm{T}(u_{l+1} - W_{l+1}\bar{r}_l)\,, \tag{B.20}$$

$$\bar{e}_l^G = \frac{g_\mathrm{l}}{g_\epsilon}\,\bar{r}_l' \odot G_{l-1}^\mathrm{T}(u_{l-1} - G_{l-1}\bar{r}_l)\,. \tag{B.21}$$

Thus, the equation of motion can be written as

$$\begin{aligned} c_\mathrm{m}\dot{u}_l &= -g_\epsilon\, u_l + g_\mathrm{l}\, u_l && \text{(B.22)}\\ &\quad + g_\mathrm{l}\,\lambda\,\gamma W_l \bar{r}_{l-1} && \text{(B.23)}\\ &\quad + g_\epsilon\,\lambda\,\gamma \bar{e}_l^W && \text{(B.24)}\\ &\quad + g_\mathrm{l}\,(1-\lambda)\,\gamma G_l \bar{r}_{l+1} && \text{(B.25)}\\ &\quad + g_\epsilon\,(1-\lambda)\,\gamma \bar{e}_l^G\,. && \text{(B.26)} \end{aligned}$$

which is identical to Eq. 5.40 after distributing the leak term

$$\begin{aligned} c_\mathrm{m}\dot{u}_l &= g_\mathrm{l}\,\lambda\,\left(\gamma W_l \bar{r}_{l-1} - u_l\right) + g_\epsilon\,\lambda\,\left(\gamma \bar{e}_l^W - u_l\right) && \text{(B.27)}\\ &\quad + g_\mathrm{l}\,(1-\lambda)\,\left(\gamma G_l \bar{r}_{l+1} - u_l\right) + g_\epsilon\,(1-\lambda)\,\left(\gamma \bar{e}_l^G - u_l\right)\,. && \text{(B.28)} \end{aligned}$$

B.2 Additional results

B.2.1 Learning MNIST with lookahead dynamics

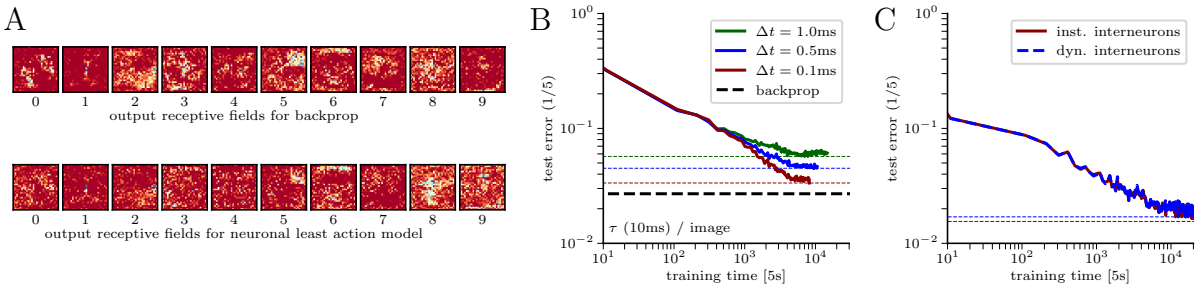

Figure B.2.: Additional results of the MNIST experiments presented in Fig. 5.6. **(A)** Comparison of the "receptive fields" of the output neurons for a network trained with (top) error backpropagation and (bottom) the model derived via neuronal least action, for one hidden area and 100ms presentation time per image. Output receptive fields are determined by calculating the response to unit vector inputs $\bar{r}_\mathrm{output} = \varphi\left(W_2\varphi(W_1\bar{r}_0)\right)$ (i.e., input images with one black pixel and all others white). Apart from blurriness, the response behavior obtained with the neuronal least action model is similar to the one obtained with backprop. **(B)** Training performance for different integration time steps. We found that for brief image presentation times, the integration step size strongly affects the final error rate achieved by the model. **(C)** Comparison of the training performance with ($\tau\dot{u}_l^\mathrm{I} = -u_l^\mathrm{I} + W_{l+1}r_l^\mathrm{P}$) and without dynamic interneurons ($u_l^\mathrm{I} = W_{l+1}\bar{r}_l^\mathrm{P}$). Both cases perform equally well.

B.2.2 Truncated backprop for recurrent networks

The derived model (Eq. 5.24) trivially generalizes to recurrent networks by dropping the area indices. In this case, the weight matrix W encodes the connections between all neurons of the network, and, e.g., u is a vector containing the membrane potential of all neurons. The weight dynamics then effectively implement truncated error backpropagation through time, which takes into account only the last network state $u(t - \mathrm{d}t)$. This is demonstrated for an areawise recurrent model Fig. B.3A-C and a fully recurrent network Fig. B.3D-F.

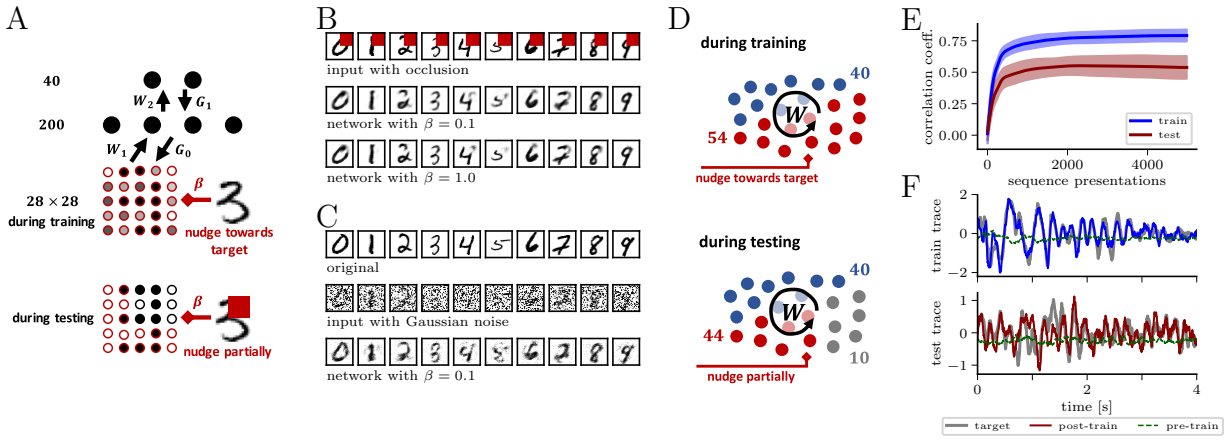

Figure B.3.: Unsupervised learning experiments for MNIST and iEEG traces. **(A)** For MNIST, we use again a deep network with additional (recurrent) top-down connections G_i between areas. The top-down input modulates the network activity in lower areas and is able to fill in missing or incorrect activities, e.g., caused by an occluding or noised part of the input. During training, the visible area is nudged towards reproducing the input pattern, i.e., images of the MNIST training data set. For testing, we nudge only part of the visible neurons, such that a small pool of visible neurons only get top-down input from the network. Furthermore, we also test whether the network is able to denoise its input (not illustrated here). **(B)** Testing with occluded images. We show the firing rate of the 28x28 visible neurons here. Note that the input is not clamped, but the visible area is nudged towards the input signal. Thus, for weak nudging ($\beta = 0.1$), the top-down propagated network activity also influences the visible neurons that do receive forward input. By varying the nudging strength β to higher values, harder "clamping" can be achieved. **(C)** Same as (B), but instead of occlusion, the input image is corrupted by static, Gaussian noise and consequently cleaned up by the top-down input from the 200 hidden neurons. **(D)** For the iEEG signals, we use a fully connected recurrent network where 54 of 94 neurons are labeled as output neurons. During training, these are constantly nudged towards the input signal (top), i.e., 8s of the iEEG signal. For testing, we chose 10 output neurons randomly which are not nudged (input occlusion) and compare how well the network is able to reproduce the target behavior from its network activity alone. **(E)** Mean correlation coefficient between the network generated behavior and the corresponding target for the 10 neurons that are not nudged. The mean is taken over 10 runs with different selections of neurons that are not nudged. The shaded area gives the standard deviation. For testing, we use another 8s sequence from the same recording. **(F)** Example traces for the pattern completion task described in (D) and (E) for the training sequences (top) and the test sequences (bottom) before and after training (only 4s are shown here).

B.3 Simulation details

B.3.1 Fig. 5.5: supervised learning of iEEG traces

As an activation function, we used the logistic function $\varphi(u_l) = \left(1 + e^{-u_l}\right)^{-1}$. The human intracranial electroencephalography (iEEG) data was recorded at the Inselspital Bern. In total, 54 electrodes were used to record the local field potentials of various brain areas before and after the onset of a seizure (for our studies, only data before seizure onset was used). The electrodes were separated into 46 input and 8 output membrane potentials. Further, the raw data was divided by a factor 200 to ensure that the membrane potentials are approximately in a range of $\pm 1 - 2$. The network was trained on a 8s excerpt of the recorded data, and consequently tested on a different 8s excerpt. Data points of the iEEG signal were sampled with a frequency of 512Hz. For simplicity, we therefore assumed that successive data points are separated by 2ms. Integration step size was set to $\Delta t = 1$ms. Since data points were 2ms apart, we up-sampled the signal to 1ms resolution by simple interpolation. Weights were initialized randomly from a normal distribution $\mathcal{N}(0, 0.1^2)$ with a cut-off at ± 0.3. Biases were not used (i.e., set constant to zero). Learning rate and nudging strength were again set to $\eta = 10^{-3}$ and $\beta = 0.1$. The membrane time constant was set to $\tau = 10$ms.

B.3.2 Figs. 5.6 and B.2: real-time learning of MNIST

As an activation function, we used a hard sigmoid function $\varphi(u_l) = u_l\theta(u_l) - [u_l - 1]\theta(u_l - 1)$, with θ the Heaviside step function. For the experiments with lookahead and 10ms and 50ms presentation time per MNIST image, we used an integration step size of $\Delta t = 0.1$ms and $\Delta t = 0.5$ms to solve the differential equation, respectively. For all other experiments, we used an integration step size of $\Delta t = 1$ms. We also used bias neurons (neurons with constant rate 1) to model offsets (or thresholds) in the activation function. However, this is not crucial for the obtained results and can also be neglected, as done in all other experiments. Biases and weights were initialized randomly from a normal distribution $\mathcal{N}(0, 0.01^2)$ with a cut-off at ± 0.03. The learning rates were set constant with $\eta = 10^{-3}$ and the nudging strength was set to $\beta = 0.1$ (note that this leads to an effective learning rate of $\eta \cdot \beta = 10^{-4}$). After 22000s training time, the learning rates were both reduced to $\eta = 10^{-4}$. The membrane time constant was set to $\tau = 10$ms. As a reference model for backprop, we implemented lookahead dynamics $\tau \dot{u}_l = -u_l + W_l r_{l-1}$, but without errors on the membrane potential and weight updates calculated "offline" via the error backpropagation algorithm. Furthermore, we found that the performance of the model without lookaheads can be further improved by increasing the separation of time scales. For instance, we found that classification rates of $\approx 93 - 94\%$ can be reached when decreasing the membrane time constant to 1ms and presenting each image for 100-200ms. MNIST images were converted to membrane potentials by normalizing the pixel values into the range $[0, 1]$.

B.3.3 Fig. 5.8: error learning with cortical microcircuits

For both experiments, the membrane time constant was set to $\tau = 10$ms. Only weights were trained and biases were set to zero. As an activation function, we used a hard sigmoid function. Forward, pyramidal-to-interneuron and interneuron-to-pyramidal weights were initialized randomly from a normal distribution $\mathcal{N}(0, 0.01^2)$ with a cut-off at ± 0.03.

For the case where nudging is one-to-one, we used a network with $784 - 500 - 10$ neurons. Furthermore, all learning rates were chosen equal $\eta = 10^{-3}$ and were subsequently reduced to $\eta = 10^{-4}$ after 53500s training time. The nudging parameters were set to $\beta = 0.1$ and $\beta^{\mathrm{I}} = \frac{0.01}{1.01}$. The feedback connections B_l^{PP} were initialized randomly from a normal distribution $\mathcal{N}(0, 0.01^2)$ with a cut-off at ± 0.03. The used integration step size was $\Delta t = 0.5$ms and the number of interneurons was fixed to 10.

For the case with mixed nudging, we used a network with $784 - 300 - 10$ neurons. All learning rates were chosen equal $\eta = 10^{-3}$ and were subsequently reduced to $\eta = 10^{-4}$ after 22000s training time. The nudging parameters were set to $\beta = 0.1$ and $\beta^{\mathrm{I}} = \frac{0.1}{1.1}$. The feedback connections B_l^{PP} and the nudging matrices B_l^{IP} were initialized randomly from a normal distribution $5 \cdot \mathcal{N}(0, 0.01^2)$ with a cut-off at ± 0.15. The used integration step size was $\Delta t = 0.25$ms and the number of interneurons was fixed to 20.

In both experiments, all connections were trained simultaneously without a pre-training phase.

B.3.4 Fig. 5.9: microcircuit learning without top-down nudging

For this experiment (in general: only for this), weight derivative terms in the differential equation for the membrane potentials, originating from the lookaheads, were considered and the membrane time constant was set to $\tau = 10$ms. As an activation function, we used a hard sigmoid function. Different from the microcircuit experiment in Fig. 5.8, interneurons are not nudged and only forward and interneuron-to-pyramidal weights are trained. The used network consisted of three areas with $784 - 500 - 10$ neurons. All weights were initialized randomly from a normal distribution $\mathcal{N}(0, 0.01^2)$ with a cut-off at ± 0.03. Nudging strength and learning rate were set to $\beta = 0.1$ and $\eta = 10^{-3}$. The used integration step size was $\Delta t = 1$ms.

B.3.5 Fig. 5.12: folded autoencoder learning CIFAR10

For the CIFAR experiments, we used a network with 3072-1000-1000-300 neurons, a logistic activation function and 10^4 training images. The integration step size was set to 0.1ms and the gating to $\lambda = 0.2$ during training. Weights were initialized randomly from a normal distribution $\mathcal{N}(0, 0.1^2)$, and sampled values were dropped and resampled if they spread by more than two standard deviations. The leak conductance was chosen as $g_{\mathrm{l}} = 2$ and the conductance of error compartments as $g_\epsilon = 0.05$. The learning rate for the generative weights was $\eta_G = 10^{-5}$, for the forward weights $\eta_W = 10^{-4}$ and the nudging strength $\beta = 0.1$. During training, each image was presented for 50ms. Furthermore, for this experiment we

used mini-batch learning with size 100. Both forward and backward weights were trained simultaneously. CIFAR images were converted to membrane potentials by first normalizing the pixel values into the range $[0, 1]$, applying the inverse activation function and truncating at -5 and 5 afterwards.

B.3.6 Fig. 5.13: dreaming of MNIST using frozen noise

For this experiment, we used a network with $22 \times 18 - 200 - 20$ neurons, where we trimmed the MNIST data set to a size of 22×18 to remove unnecessary white spaces at the borders. Weights were initialized randomly from a normal distribution $\mathcal{N}(0, 0.5^2)$, with resampling of extreme values as in the CIFAR10 experiment. Conductances, integration time step and nudging strength were chosen as in the CIFAR10 experiment. Throughout the experiment, gating was set to $\lambda = 0.1$ and the learning rate for both forward and backward weights was set to 10^{-3}. During training, after presenting a new image, the noise vector for the latent neurons was first sampled from a normal distribution $\mathcal{N}(0, 2^2)$ and the network simulated for 50ms with plasticity turned off. Only after this initial phase when the network has reached a steady state, plasticity was turned on for an additional 10ms before switching to a new input image.

B.3.7 Fig. B.3: truncated backprop for recurrent networks

For both experiments, the membrane time constant was set to $\tau = 10$ms. As an activation function, we used the logistic function. Weights were initialized randomly from a normal distribution $\mathcal{N}(0, 0.1^2)$ with a cut-off at ± 0.3 and biases from $\mathcal{N}(0, 0.01^2)$ with a cut-off at ± 0.03.

For the MNIST experiment, we used a areawise recurrent network with $784 - 200 - 40$ neurons. We used a decaying learning rate $\eta = \frac{10^{-4}}{1+n/10}$, where n starts at 0 and is increased by 1 every 1000 training examples. The used integration step size is $\Delta t = 1$ms. Every MNIST image is presented for 100ms. Image pixel values were translated to membrane potentials via $u^{\text{target}} = \varphi^{-1}\left(\frac{\text{image}+a_1}{a_2}\right)$, where the factors $a_1 = 0.1$ and $a_2 = \frac{1.1}{0.9}$ were chosen to appropriately limit the range of the membrane potentials. For the denoising task, we added Gaussian noise $\mathcal{N}(0, 4^2)$ to the images before nudging the network. The results shown in the figure are, however, not the membrane potentials, but the rate responses.

For the iEEG experiment, we used a fully recurrent network with 54+40 neurons. The integration step size was set to $\Delta t = 2$ms. Learning rates and nudging parameter were set to $\eta = 10^{-3}$ and $\beta = 0.1$. For training and testing, we used two 8s sequences taken from the iEEG recording.

B.4 Implementation details

B.4.1 Numerical methods

The gradient-based model can be implemented using simple Euler integration. Solving the model with lookahead, however, is a bit more demanding, as will be explained in the following. For generality and convenience, we use the notation for recurrent networks mentioned in Appendix B.2.2. To solve Eq. 5.24, several integration methods can be used. Since both sides of the differential equation depend on $\dot{u}$, i.e.,

$$\tau\dot{u}(t) = f\left(u(t), \dot{u}(t)\right) , \tag{B.29}$$

a simple perturbative approximation is to use the $\dot{u}$ of the previous time step on the right-hand side

$$\tau\dot{u}(t) \approx f\left(u(t), \dot{u}(t-\Delta t)\right) , \tag{B.30}$$

where Δt is the integration step size. This way, Euler Integration can be used to solve the ODE. We found that such an integration scheme works well with and without nudging for small Δt and small weights, but numerical instabilities can occur when plasticity is additionally turned on. Even though the behavior might become more stable by adjusting the integration method, decreasing the learning rate and integration time step or using a batch-in-time approach (accumulating updates before applying them), we decided to use an exact solver of the ODE.

Since on the right-hand side of Eq. 5.24, $\dot{u}$ only appears linearly, the differential equation can be rewritten in the following form

$$F\dot{u} = m \tag{B.31}$$

with matrix F and vector m that are independent of $\dot{u}$. F and m are given by

$$\frac{1}{\tau}F = \mathbb{1} - \zeta_{\mathrm{T}}\left(B^{\mathrm{PP}}\right) - \zeta\left(W\right) + \zeta_{\mathrm{T}}\left(W^{\mathrm{PI}}\right)W^{\mathrm{I}} - \mathrm{diag}\left(\bar{r}'' \odot e_{\mathrm{A}}\right) + \beta\mathbb{1}_y , \tag{B.32}$$

$$m = \bar{r}' e_{\mathrm{A}} + \beta(y-u) + \left(W\bar{r} + x - u\right) + m_{\dot{W}} , \tag{B.33}$$

where $\mathrm{diag}\left(\cdot\right)$ turns vectors into a diagonal matrix, with

$$\zeta\left(W\right)_{ij} = W_{ij}\bar{r}'_j , \quad \zeta_{\mathrm{T}}\left(W\right)_{ij} = W_{ij}\bar{r}'_i , \tag{B.34}$$

$$W^{\mathrm{I}} = \tilde{\beta}^{\mathrm{I}}\zeta\left(W^{\mathrm{IP}}\right) + \beta^{\mathrm{I}}B^{\mathrm{IP}} , \tag{B.35}$$

$$e_{\mathrm{A}} = B^{\mathrm{PP}}u - W^{\mathrm{PI}}u^{\mathrm{I}} , \tag{B.36}$$

$$\frac{1}{\tau}m_{\dot{W}} = \dot{W}\bar{r} + \bar{r}'\left(\dot{B}^{\mathrm{PP}}u - \dot{W}^{\mathrm{PI}}u^{\mathrm{I}} - \tilde{\beta}^{\mathrm{I}}W^{\mathrm{PI}}\dot{W}^{\mathrm{IP}}\bar{r}\right) , \tag{B.37}$$

where $\tilde{\beta}^{\mathrm{I}} = 1 - \beta^{\mathrm{I}}$, B^{IP} are the top-down connections nudging the interneurons u^{I} and $\mathbb{1}_y$ is a unit matrix with zeros everywhere except the diagonal elements for the output neurons (for a derivation, see end of section). We dropped the upper index 'P' for pyramidal neurons to increase readability. $\zeta\left(\cdot\right)$ ($\zeta_{\mathrm{T}}\left(\cdot\right)$) multiplies matrix columns (rows) with the elements of

the vector $\bar{r}'$. For this operation, we can immediately see that $\zeta\left(W\right)^{\mathrm{T}} = \zeta_{\mathrm{T}}\left(W^{\mathrm{T}}\right)$ because $\zeta_{\mathrm{T}}\left(W^{\mathrm{T}}\right)_{ij} = W_{ji}\bar{r}'_i = \zeta\left(W\right)_{ji}$. The time-dependence of the weights is accounted for by $m_{\dot{W}}$ and can be neglected for small learning rates $\eta \ll \tau^{-1}$. Input currents and output targets are given by $x = \bar{x} + \tau\dot{\bar{x}}$ and $y = \bar{y} + \tau\dot{\bar{y}}$. The interneuron potential can either be calculated via Eq. 5.28 or Eq. 5.26. In the latter case, the term $\zeta_{\mathrm{T}}\left(W^{\mathrm{PI}}\right) \cdot (u^{\mathrm{I}} - \beta^{\mathrm{I}} B^{\mathrm{IP}} u - \tilde{\beta}^{\mathrm{I}} W^{\mathrm{IP}} \bar{r})$ has to be added to Eq. B.33.

For the backprop-like configuration of interneuron weights ($B^{\mathrm{PP}} = W^{\mathrm{T}}$, $W^{\mathrm{PI}} = W^{\mathrm{T}}$, $W^{\mathrm{IP}} = W$ and $\beta^{\mathrm{I}} = 0$), $F = F^{\mathrm{T}}$ is symmetric (see end of this section for a quick proof). Furthermore, during simulations F turned out to be always positive definite, allowing us to solve Eq. B.31 via Cholesky decomposition. In cases where this was not possible (e.g., when using independent interneuron connections), we used LU decomposition instead. Since the updates for the weight and bias derivatives depend only on the current weights, biases and voltages, these can be calculated before updating $\dot{u}$ with the learning rules Eqs. 5.10, 5.29 and 5.33. A single integration step (after initialization) is summarized in Algorithm 1.

Algorithm 1 Network update after initialization

current state: $[u(t), W(t)]$, $[u^{\mathrm{I}}(t), W^{\mathrm{PI}}(t), W^{\mathrm{IP}}(t), B^{\mathrm{PP}}(t)]$
 # drop (t)-notation for convenience
calculate weight derivatives
$\dot{W} \leftarrow \eta_W (u - W\,\bar{r})\,\bar{r}^{\mathrm{T}}$
if no plastic interneurons **then**
    $\dot{B}^{\mathrm{PP}}, \dot{W}^{\mathrm{PI}} \leftarrow \dot{W}^{\mathrm{T}}$
    $\dot{W}^{\mathrm{IP}} \leftarrow \dot{W}$
else
    $\dot{B}^{\mathrm{PP}} \leftarrow 0$
    $\dot{W}^{\mathrm{PI}} \leftarrow \eta_{W^{\mathrm{PI}}} \left(B^{\mathrm{PP}}\,u - W^{\mathrm{PI}} u^{\mathrm{I}}\right)\,(u^{\mathrm{I}})^{\mathrm{T}}$
    $\dot{W}^{\mathrm{IP}} \leftarrow \eta_{W^{\mathrm{IP}}} (u^{\mathrm{I}} - W^{\mathrm{IP}} \bar{r})\,\bar{r}^{\mathrm{T}}$
calculate voltage derivatives
$\dot{u} \leftarrow$ solve $F\dot{u} = m$ via Cholesky or LU decomposition
if dynamic interneurons **then**
    $\dot{u}^{\mathrm{I}} \leftarrow \frac{1}{\tau}\left(-u_{\mathrm{I}} + (1-\beta^{\mathrm{I}})(W^{\mathrm{IP}} r) + \beta^{\mathrm{I}}\; B^{\mathrm{IP}}(u + \tau\dot{u})\right)$
update network state
for $X \in [u,\, W,\, W^{\mathrm{PI}},\, W^{\mathrm{IP}},\, B^{\mathrm{PP}}]$ **do**
    $X \leftarrow X + \dot{X}\Delta t$
if dynamic interneurons **then**
    $u^{\mathrm{I}} \leftarrow u^{\mathrm{I}} + \dot{u}^{\mathrm{I}}\Delta t$
else
    $u^{\mathrm{I}} \leftarrow (1-\beta^{\mathrm{I}})(W^{\mathrm{IP}}\,\bar{r}) + \beta^{\mathrm{I}} B^{\mathrm{IP}} u$

Of course if the interneuron connections are not plastic, it is not necessary to update them explicitly but simply let them track the forward weights W.

Note that even though solving the differential equation with an implicit solver as described here comes with the benefit of numerical stability, the complexity of solving the linear system

Eq. B.29 is a huge bottleneck when simulating larger neural networks. Thus, we hope that in future work, the presented theory can be simplified or faster (approximate) solution methods can be found to speed up simulation time, e.g., by using approximation schemes like Eq. B.30.

Symmetry of $\boldsymbol{F}$: For $B^{\mathrm{PP}} = W^{\mathrm{T}}$, $W^{\mathrm{PI}} = W^{\mathrm{T}}$, $W^{\mathrm{IP}} = W$ and $\beta^{\mathrm{I}} = 0$, F is a symmetric matrix. This can be checked for each term separately: first of all, the first and the two last terms are all diagonal matrices. The sum of the second and third term is symmetric since $\zeta_{\mathrm{T}}\left(B^{\mathrm{PP}}\right)^{\mathrm{T}} = \zeta_{\mathrm{T}}\left(W^{\mathrm{T}}\right)^{\mathrm{T}} = \zeta\left(W\right)$ and $\zeta\left(W\right)^{\mathrm{T}} = \zeta_{\mathrm{T}}\left(W^{\mathrm{T}}\right)$. The last term is symmetric as well, since $\left[\zeta_{\mathrm{T}}\left(W^{\mathrm{T}}\right)\zeta\left(W\right)\right]^{\mathrm{T}} = \zeta\left(W\right)^{\mathrm{T}}\zeta_{\mathrm{T}}\left(W^{\mathrm{T}}\right)^{\mathrm{T}} = \zeta_{\mathrm{T}}\left(W^{\mathrm{T}}\right)\zeta\left(W\right)$, proving that F is a symmetric matrix in this special case.

Derivation of alternative ODE: We start with the differential equation of neuronal dynamics:

$$\begin{aligned}
\dot{u} &= -u + Wr + e + x\,,\\
r &= \bar{r} + \tau\dot{\bar{r}}\,,\\
e &= \bar{e} + \tau\dot{\bar{e}} + \beta(y - u - \tau\dot{u})\,,
\end{aligned}$$

with $\bar{e} = \bar{r}' \odot \left(B^{\mathrm{PP}}u - W^{\mathrm{PI}}u^{\mathrm{I}}\right)$. To write this in the form of Eq. B.31, we first separate the equation into terms that either do or do not depend on $\dot{u}$. For convenience, we neglect weight derivatives here. The input term Wr can be written as

$$\begin{aligned}
Wr &= W\bar{r} + \tau W\dot{\bar{r}} && \text{(B.38)}\\
&= W\bar{r} + \tau W(\bar{r}' \odot \dot{u}) && \text{(B.39)}\\
&= W\bar{r} + \tau\zeta\left(W\right)\dot{u}\,, && \text{(B.40)}
\end{aligned}$$

where we used that $W(\bar{r}' \odot \dot{u})_i = \sum_j W_{ij}\bar{r}'_j\dot{u}_j = \sum_j \zeta\left(W\right)_{ij}\dot{u}_j$. The input x is independent of $\dot{u}$. Thus, we get:

$$-u + Wr + x = -u + W\bar{r} + x + \tau\zeta\left(W\right)\dot{u}\,. \tag{B.41}$$

Looking at e, $\bar{e}$ only depends on u, leaving us with $\dot{\bar{e}}$:

$$\dot{\bar{e}} = \dot{\bar{r}}' \odot \left(B^{\mathrm{PP}}u - W^{\mathrm{PI}}u^{\mathrm{I}}\right) + \bar{r}' \odot B^{\mathrm{PP}}\dot{u} - \bar{r}' \odot W^{\mathrm{PI}}\dot{u}^{\mathrm{I}} \tag{B.42}$$

where the first term simplifies to $\mathrm{diag}\left(\bar{r}'' \odot e_{\mathrm{A}}\right)\dot{u}$ using the chain rule, i.e., $\dot{\bar{r}}' = \bar{r}'' \odot \dot{u}$. The second term can be simplified to

$$\bar{r}' \odot B^{\mathrm{PP}}\dot{u} = \zeta_{\mathrm{T}}\left(B^{\mathrm{PP}}\right)\dot{u}\,, \tag{B.43}$$

using $(\bar{r}' \odot B^{\mathrm{PP}}\dot{u})_i = \sum_j \bar{r}'_i B^{\mathrm{PP}}_{ij}\dot{u}_j = \sum_j \zeta_{\mathrm{T}}\left(B^{\mathrm{PP}}\right)_{ij}\dot{u}_j$. The last term simplifies to

$$\begin{aligned}
\zeta_{\mathrm{T}}\left(W^{\mathrm{PI}}\right)\dot{u}^{\mathrm{I}} &= \zeta_{\mathrm{T}}\left(W^{\mathrm{PI}}\right)\left[(1-\beta^{\mathrm{I}})W^{\mathrm{IP}}(\bar{r}' \odot \dot{u}) + \beta^{\mathrm{I}}B^{\mathrm{IP}}\dot{u}\right] && \text{(B.44)}\\
&= \zeta_{\mathrm{T}}\left(W^{\mathrm{PI}}\right)\left[\tilde{\beta}^{\mathrm{I}}\zeta\left(W^{\mathrm{IP}}\right)\dot{u} + \beta^{\mathrm{I}}B^{\mathrm{IP}}\dot{u}\right] && \text{(B.45)}\\
&= \zeta_{\mathrm{T}}\left(W^{\mathrm{PI}}\right)W^{\mathrm{I}}\dot{u}\,, && \text{(B.46)}
\end{aligned}$$

using $u^{\mathrm{I}} = (1-\beta^{\mathrm{I}})(W^{\mathrm{IP}}\bar{r}) + \beta^{\mathrm{I}} B^{\mathrm{IP}} u$. The term containing the nudging towards target y splits up to

$$\beta(y - u - \tau\dot{u}) = \beta(y - u) - \beta\tau\mathbb{1}_y\dot{u} \tag{B.47}$$

Thus, the error can be rewritten as

$$e = \bar{r}' e_{\mathrm{A}} + \beta(y - u) + \tau\left[\mathrm{diag}\left(\bar{r}'' \odot e_{\mathrm{A}}\right) + \zeta_{\mathrm{T}}\left(B^{\mathrm{PP}}\right) - \zeta_{\mathrm{T}}\left(W^{\mathrm{PI}}\right) W^{\mathrm{I}} - \beta\mathbb{1}_y\right]\dot{u}\,. \tag{B.48}$$

Combining Eqs. B.41 and B.48 and moving all terms containing $\dot{u}$ on the left hand side, we obtain F and m as in Eqs. B.32 and B.33.

B.4.2 Simulation software

The results in Fig. 5.12 were obtained using Python 3.7.3 and Tensorflow 1.13.1, and all other results in Chapter 5 using Python 2.7.12 and Tensorflow 1.3.0. Initial studies were done on CPUs of the BwForCluster NEMO, the HPC cluster of the state Baden Württemberg (https://wiki.bwhpc.de/e/Category:BwForCluster_NEMO, 24.01.2020), and full experiments on UBELIX, the HPC cluster at the University of Bern (http://www.id.unibe.ch/hpc, 24.01.2020), using Nvidia Geforce GTX 1080 Ti and Nvidia Tesla P100 GPUs.

C | List of published articles

The following articles were published as part of this thesis (* marks equal contributions):

Billaudelle*, S., Stradmann*, Y., Schreiber*, K., Cramer*, B., Baumbach*, A., **Dold*, D.**, Göltz*, J., Kungl*, A. F., Wunderlich*, T. C. et al. (2020). *Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate*. 2020 IEEE International Symposium on Circuits and Systems (ISCAS).

Göltz, J., Baumbach, A., Billaudelle, S., Breitwieser, O., **Dold, D.**, Kriener, L., ... Petrovici, M. A. (2019). *Fast and deep neuromorphic learning with time-to-first-spike coding*. arXiv:1912.11443 preprint.

Kungl, A. F., Schmitt, S., Klähn, J., Müller, P., Baumbach, A., **Dold, D.**, ... Kleider, M. et al. (2019). *Accelerated physical emulation of Bayesian inference in spiking neural networks*. Frontiers in Neuroscience, 13, 1201.

Dold*, D., I., Bytschok*, Kungl, A. F., Baumbach, A., Breitwieser, O., Schemmel, J., Meier, K. and Petrovici*, M. A. (2019). *Stochasticity from function - why the Bayesian brain may need no noise*. Neural Networks, 119, 200-213.

Bytschok*, I., **Dold*, D.**, Schemmel, J., Meier, K. and Petrovici*, M. A. (2017). *Spike-based probabilistic inference with correlated noise*. arXiv:1707.01746 preprint.

Bibliography

Abbott, L. F., and S. B. Nelson, Synaptic plasticity: taming the beast, *Nature neuroscience*, *3*(11), 1178–1183, 2000.

Ackley, D. H., G. E. Hinton, and T. J. Sejnowski, A learning algorithm for boltzmann machines, *Cognitive science*, *9*(1), 147–169, 1985.

Aitchison, L., and M. Lengyel, The hamiltonian brain: efficient probabilistic inference with excitatory-inhibitory neural circuit dynamics, *PLoS computational biology*, *12*(12), e1005,186, 2016.

Akopyan, F., et al., Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *34*(10), 1537–1557, 2015.

Allen, C., and C. F. Stevens, An evaluation of causes for unreliability of synaptic transmission, *Proceedings of the National Academy of Sciences*, *91*(22), 10,380–10,383, 1994.

Almeida, L. B., A learning rule for asynchronous perceptrons with feedback in a combinatorial environment., in *Proceedings, 1st First International Conference on Neural Networks*, vol. 2, pp. 609–618, IEEE, 1987.

Arieli, A., A. Sterkin, A. Grinvald, and A. Aertsen, Dynamics of ongoing activity: explanation of the large variability in evoked cortical responses, *Science*, *273*(5283), 1868, 1996.

Azouz, R., and C. M. Gray, Cellular mechanisms contributing to response variability of cortical neurons in vivo, *Journal of Neuroscience*, *19*(6), 2209–2223, 1999.

Baumbach, A., J. Schemmel, and M. A. Petrovici, Magnetic phenomena in ensembles of spiking neurons, *Bernstein Conference*, doi:10.12751/nncn.bc2019.0240, 2019.

Bellec, G., D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, Long short-term memory and learning-to-learn in networks of spiking neurons, in *Advances in Neural Information Processing Systems*, pp. 787–797, 2018.

Bellec, G., F. Scherr, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets, *arXiv preprint arXiv:1901.09049*, 2019.

Bengio, Y., G. Mesnil, Y. Dauphin, and S. Rifai, Better mixing via deep representations, in *International conference on machine learning*, pp. 552–560, 2013.

Benjamin, B. V., et al., Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations, *Proceedings of the IEEE*, *102*(5), 699–716, 2014.

Berkes, P., G. Orbán, M. Lengyel, and J. Fiser, Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment, *Science*, *331*(6013), 83–87, 2011.

Bertolero, M. A., B. T. Yeo, and M. D'Esposito, The modular and integrative functional architecture of the human brain, *Proceedings of the National Academy of Sciences*, *112*(49), E6798–E6807, 2015.

Bi, G.-q., and M.-m. Poo, Synaptic modification by correlated activity: Hebb's postulate revisited, *Annual review of neuroscience*, *24*(1), 139–166, 2001.

Billaudelle, S., et al., Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate, *arXiv preprint arXiv:1912.12980*, 2019.

Bittner, K. C., A. D. Milstein, C. Grienberger, S. Romani, and J. C. Magee, Behavioral time scale synaptic plasticity underlies ca1 place fields, *Science*, *357*(6355), 1033–1036, 2017.

Blake, R., and N. K. Logothetis, Visual competition, *Nature Reviews Neuroscience*, *3*(1), 13, 2002.

Bohte, S. M., J. N. Kok, and J. A. La Poutré, Spikeprop: backpropagation for networks of spiking neurons., in *ESANN*, pp. 419–424, 2000.

Boole, G., *The mathematical analysis of logic*, Philosophical Library, 1847.

Brascamp, J. W., R. Van Ee, A. J. Noest, R. H. Jacobs, and A. V. van den Berg, The time course of binocular rivalry reveals a fundamental role of noise, *Journal of vision*, *6*(11), 8–8, 2006.

Brea, J., A. T. Gaál, R. Urbanczik, and W. Senn, Prospective coding by spiking neurons, *PLoS computational biology*, *12*(6), e1005,003, 2016.

Brette, R., and W. Gerstner, Adaptive exponential integrate-and-fire model as an effective description of neuronal activity, *J. Neurophysiol.*, *94*, 3637 – 3642, 2005.

Brooks, R., D. Hassabis, D. Bray, and A. Shashua, Is the brain a good model for machine intelligence?, *Nature*, *482*(7386), 462, 2012.

Brunel, N., Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons, *Journal of computational neuroscience*, *8*(3), 183–208, 2000.

Brunner, D., M. Soriano, and G. Van Der Sand, *Photonic Reservoir Computing (Optical Recurrent Neural Networks)*, De Gruyter Studies in Mathematical Physics, 2019.

Buesing, L., J. Bill, B. Nessler, and W. Maass, Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons, *PLoS computational biology*, *7*(11), e1002,211, 2011.

Bullmore, E., and O. Sporns, Complex brain networks: graph theoretical analysis of structural and functional systems, *Nature Reviews Neuroscience*, *10*(3), 186, 2009.

Burrello, A., K. Schindler, L. Benini, and A. Rahimi, One-shot learning for ieeg seizure detection using end-to-end binary operations: Local binary patterns with hyperdimensional computing, in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1–4, IEEE, 2018.

Bytschok, I., Computing with noise in spiking neural networks, Ph.D. thesis, 2017.

Cáceres, M. O., Harmonic potential driven by long-range correlated noise, *Physical Review E*, *60*(5), 5208, 1999.

Carandini, M., Amplification of trial-to-trial response variability by neurons in visual cortex, *PLoS biology*, *2*(9), e264, 2004.

Carleo, G., and M. Troyer, Solving the quantum many-body problem with artificial neural networks, *Science*, *355*(6325), 602–606, 2017.

Chen, Z. J., Y. He, P. Rosa-Neto, J. Germann, and A. C. Evans, Revealing modular architecture of human brain structural networks by using cortical thickness from mri, *Cerebral cortex*, *18*(10), 2374–2381, 2008.

Chistiakova, M., N. M. Bannon, M. Bazhenov, and M. Volgushev, Heterosynaptic plasticity: multiple mechanisms and multiple roles, *The Neuroscientist*, *20*(5), 483–498, 2014.

Ciregan, D., U. Meier, and J. Schmidhuber, Multi-column deep neural networks for image classification, in *2012 IEEE conference on computer vision and pattern recognition*, pp. 3642–3649, IEEE, 2012.

Clopath, C., L. Büsing, E. Vasilaki, and W. Gerstner, Connectivity reflects coding: a model of voltage-based stdp with homeostasis, *Nature neuroscience*, *13*(3), 344, 2010.

Cohen, G., S. Afshar, J. Tapson, and A. van Schaik, Emnist: an extension of mnist to handwritten letters, *arXiv preprint arXiv:1702.05373*, 2017.

Comsa, I. M., K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, Temporal coding in spiking neural networks with alpha synaptic function, *arXiv preprint arXiv:1907.13223*, 2019.

Costa, R., I. A. Assael, B. Shillingford, N. de Freitas, and T. Vogels, Cortical microcircuits as gated-recurrent neural networks, in *Advances in neural information processing systems*, pp. 272–283, 2017.

Crick, F., The recent excitement about neural networks, *Nature*, *337*(6203), 129–132, 1989.

Davies, M., Benchmarks for progress in neuromorphic computing, *Nature Machine Intelligence*, *1*(9), 386–388, 2019.

Davies, M., et al., Loihi: A neuromorphic manycore processor with on-chip learning, *IEEE Micro*, *38*(1), 82–99, 2018.

Davison, A., D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, Pynn: a common interface for neuronal network simulators, *Frontiers in Neuroinformatics*, *2*, 11, doi:10.3389/neuro.11.011.2008, 2009.

Dayan, P., and L. F. Abbott, Theoretical neuroscience: computational and mathematical modeling of neural systems, 2001.

Deepmind, The google deepmind challenge match, URL: https://deepmind.com/alphago-korea, 2016.

Destexhe, A., M. Rudolph, and D. Paré, The high-conductance state of neocortical neurons in vivo, *Nature reviews neuroscience*, *4*(9), 739, 2003.

Dold, D., Stochastic computation in spiking neural networks without noise, Master's thesis, Universität Heidelberg, 2016.

Dold, D., I. Bytschok, A. F. Kungl, A. Baumbach, O. Breitwieser, W. Senn, J. Schemmel, K. Meier, and M. A. Petrovici, Stochasticity from function — why the bayesian brain may need no noise, *Neural Networks*, *119*, 200 – 213, doi:https://doi.org/10.1016/j.neunet.2019.08.002, 2019a.

Dold, D., J. Sacramento, A. F. Kungl, W. Senn, and M. A. Petrovici, An energy-based model of folded autoencoders for unsupervised learning in cortical hierarchies, in *Bernstein Conference*, Berlin, doi:10.12751/nncn.bc2019.0156, 2019b.

Douglas, R. J., K. A. Martin, and D. Whitteridge, A canonical microcircuit for neocortex, *Neural computation*, *1*(4), 480–488, 1989.

Echeveste, R., L. Aitchison, G. Hennequin, and M. Lengyel, Cortical-like dynamics in recurrent circuits optimized for sampling-based probabilistic inference, *bioRxiv*, p. 696088, 2019.

Elman, J. L., Finding structure in time, *Cognitive science*, *14*(2), 179–211, 1990.

Elsayed, G., S. Shankar, B. Cheung, N. Papernot, A. Kurakin, I. Goodfellow, and J. Sohl-Dickstein, Adversarial examples that fool both computer vision and time-limited humans, in *Advances in Neural Information Processing Systems*, pp. 3910–3920, 2018.

Fiser, J., C. Chiu, and M. Weliky, Small modulation of ongoing cortical dynamics by sensory input during natural vision, *Nature*, *431*(7008), 573, 2004.

Fiser, J., P. Berkes, G. Orbán, and M. Lengyel, Statistically optimal perception and learning: from behavior to neural representations, *Trends in cognitive sciences*, *14*(3), 119–130, 2010.

Fisher, C. K., A. M. Smith, and J. R. Walsh, Boltzmann encoded adversarial machines, *arXiv preprint arXiv:1804.08682*, 2018.

Frémaux, N., and W. Gerstner, Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules, *Frontiers in neural circuits*, *9*, 85, 2016.

Frenkel, C., M. Lefebvre, and D. Bol, Learning without feedback: Direct random target projection as a feedback-alignment algorithm with layerwise feedforward training, *arXiv preprint arXiv:1909.01311*, 2019.

Friedmann, S., J. Schemmel, A. Grübl, A. Hartel, M. Hock, and K. Meier, Demonstrating hybrid learning in a flexible neuromorphic hardware system, *IEEE transactions on biomedical circuits and systems*, *11*(1), 128–142, 2016.

Frith, C., and R. Dolan, The role of the prefrontal cortex in higher cognitive functions, *Cognitive brain research*, *5*(1-2), 175–181, 1996.

Fuhrmann, G., I. Segev, H. Markram, and M. Tsodyks, Coding of temporal information by activity-dependent synapses, *Journal of neurophysiology*, *87*(1), 140–148, 2002.

Fukushima, K., Neocognitron: A hierarchical neural network capable of visual pattern recognition, *Neural networks*, *1*(2), 119–130, 1988.

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

Gerstner, W., Spiking neurons, *Tech. rep.*, MIT-press, 1998.

Gerstner, W., and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*, Cambridge university press, 2002.

Gerstner, W., and R. Naud, How good are neuron models?, *Science*, *326*(5951), 379–380, 2009.

Gewaltig, M.-O., and M. Diesmann, Nest (neural simulation tool), *Scholarpedia*, *2*(4), 1430, 2007.

Gidon, A., T. A. Zolnik, P. Fidzinski, F. Bolduan, A. Papoutsi, P. Poirazi, M. Holtkamp, I. Vida, and M. E. Larkum, Dendritic action potentials and computation in human layer 2/3 cortical neurons, *Science*, *367*(6473), 83–87, 2020.

Gollisch, T., and M. Meister, Rapid neural coding in the retina with relative spike latencies, *science*, *319*(5866), 1108–1111, 2008.

Golowasch, J., and F. Nadim, *Capacitance, Membrane*, pp. 1–5, Springer New York, New York, NY, doi:10.1007/978-1-4614-7320-6_32-1, 2013.

Göltz, J., et al., Fast and deep neuromorphic learning with time-to-first-spike coding, *arXiv preprint arXiv:1912.11443*, 2019.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Großkinsky, M., Neural sampling with linear feedback shift registers as a source of noise, Bachelor thesis, Heidelberg University, 2016.

Guerguiev, J., T. P. Lillicrap, and B. A. Richards, Towards deep learning with segregated dendrites, *ELife*, *6*, e22,901, 2017.

Gürtler, N., A markovian model of lif networks, Master's thesis, Universität Heidelberg, 2018.

Gütig, R., and H. Sompolinsky, The tempotron: a neuron that learns spike timing–based decisions, *Nature neuroscience*, *9*(3), 420, 2006.

Hamerly, R., L. Bernstein, A. Sludds, M. Soljačić, and D. Englund, Large-scale optical neural networks based on photoelectric multiplication, *Physical Review X*, *9*(2), 021,032, 2019.

Harris, J. J., R. Jolivet, and D. Attwell, Synaptic energy use and supply, *Neuron*, *75*(5), 762–777, 2012.

Hassabis, D., D. Kumaran, C. Summerfield, and M. Botvinick, Neuroscience-inspired artificial intelligence, *Neuron*, *95*(2), 245–258, 2017.

Hastings, W. K., Monte carlo sampling methods using markov chains and their applications, 1970.

Häunggi, P., and P. Jung, Colored noise in dynamical systems, *Advances in chemical physics*, *89*, 239–326, 1994.

He, K., X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hebb, D., The organization of behavior, 1949.

Hennessy, J. L., and D. A. Patterson, *Computer architecture: a quantitative approach (fifth edition)*, Elsevier, 2012.

Henry, G., P. Bishop, R. Tupper, and B. Dreher, Orientation specificity and response variability of cells in the striate cortex, *Vision research*, *13*(9), 1771–1779, 1973.

Hinton, G., How to do backpropagation in a brain, URL: https://www.cs.toronto.edu/~hinton/backpropincortex2007.pdf, workshop talk, NeurIPS, 2007.

Hinton, G. E., Training products of experts by minimizing contrastive divergence, *Neural computation*, *14*(8), 1771–1800, 2002.

Hinton, G. E., A practical guide to training restricted boltzmann machines, in *Neural networks: Tricks of the trade*, pp. 599–619, Springer, 2012.

Hinton, G. E., and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *science*, *313*(5786), 504–507, 2006.

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

Hochreiter, S., and J. Schmidhuber, Long short-term memory, *Neural computation*, *9*(8), 1735–1780, 1997.

Hodgkin, A. L., and A. F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, *The Journal of physiology*, *117*(4), 500–544, 1952.

Holden, D., B. C. Duong, S. Datta, and D. Nowrouzezahrai, Subspace neural physics: fast data-driven interactive simulation, in *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, p. 6, ACM, 2019.

Hopfield, J. J., Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the national academy of sciences*, *79*(8), 2554–2558, 1982.

Hoyer, P. O., and A. Hyvärinen, Interpreting neural response variability as monte carlo sampling of the posterior, in *Advances in neural information processing systems*, pp. 293–300, 2003.

Hubel, D. H., and T. N. Wiesel, Receptive fields of single neurones in the cat's striate cortex, *The Journal of physiology*, *148*(3), 574–591, 1959.

Hubel, D. H., and T. N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, *The Journal of physiology*, *160*(1), 106–154, 1962.

Huh, D., and T. J. Sejnowski, Gradient descent for spiking neural networks, in *Advances in Neural Information Processing Systems 31*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, pp. 1433–1443, Curran Associates, Inc., 2018.

Illing, B., W. Gerstner, and J. Brea, Biologically plausible deep learning–but how far can we go with shallow networks?, *Neural Networks*, 2019.

Indiveri, G., and S.-C. Liu, Memory and information processing in neuromorphic systems, *Proceedings of the IEEE*, *103*(8), 1379–1397, 2015.

Indiveri, G., E. Chicca, and R. Douglas, A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity, *IEEE transactions on neural networks*, *17*(1), 211–221, 2006.

Indiveri, G., et al., Neuromorphic silicon neuron circuits, *Frontiers in neuroscience*, *5*, 73, 2011.

Ivakhnenko, A. G., Polynomial theory of complex systems, *IEEE transactions on Systems, Man, and Cybernetics*, (4), 364–378, 1971.

Izhikevich, E. M., Which model to use for cortical spiking neurons?, *IEEE transactions on neural networks*, *15*(5), 1063–1070, 2004.

Jaderberg, M., W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, Decoupled neural interfaces using synthetic gradients, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1627–1635, JMLR. org, 2017.

Jia, Y., et al., Transfer learning from speaker verification to multispeaker text-to-speech synthesis, in *Advances in neural information processing systems*, pp. 4480–4490, 2018.

Johns, P., *Clinical Neuroscience E-Book: An Illustrated Colour Text*, Elsevier Health Sciences, 2014.

Jordan, J., M. A. Petrovici, O. Breitwieser, J. Schemmel, K. Meier, M. Diesmann, and T. Tetzlaff, Deterministic networks for probabilistic computing, *Scientific reports*, *9*(1), 1–17, 2019a.

Jordan, J., J. Sacramento, M. A. Petrovici, and W. Senn, Error-driven learning supports bayes-optimal multisensory integration via conductance-based dendrites, in *Cosyne Abstracts 2019*, Lisbon, 2019b.

Kaiser, J., H. Mostafa, and E. Neftci, Synaptic plasticity dynamics for deep continuous local learning, *arXiv preprint arXiv:1811.10766*, 2018.

Kamyshanska, H., and R. Memisevic, The potential energy of an autoencoder, *IEEE transactions on pattern analysis and machine intelligence*, *37*(6), 1261–1273, 2014.

Keller, G. B., and T. D. Mrsic-Flogel, Predictive processing: a canonical cortical computation, *Neuron*, *100*(2), 424–435, 2018.

Khaligh-Razavi, S.-M., and N. Kriegeskorte, Deep supervised, but not unsupervised, models may explain it cortical representation, *PLoS computational biology*, *10*(11), 2014.

Kheradpisheh, S. R., and T. Masquelier, S4nn: temporal backpropagation for spiking neural networks with one spike per neuron, *arXiv preprint arXiv:1910.09495*, 2019.

Kheradpisheh, S. R., M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, Stdp-based spiking deep convolutional neural networks for object recognition, *Neural Networks*, *99*, 56–67, 2018.

Khintchine, A., Korrelationstheorie der stationären stochastischen prozesse, *Mathematische Annalen*, *109*(1), 604–615, 1934.

Kingma, D. P., and M. Welling, Auto-encoding variational bayes, *arXiv preprint arXiv:1312.6114*, 2013.

Kleider, M., Neuron circuit characterization in a neuromorphic system, Ph.D. thesis, 2017.

Knill, D. C., and W. Richards, *Perception as Bayesian inference*, Cambridge University Press, 1996.

Koke, C., Device variability in synapses of neuromorphic circuits, Ph.D. thesis, 2017.

Köndgen, H., C. Geisler, S. Fusi, X.-J. Wang, H.-R. Lüscher, and M. Giugliano, The dynamical response properties of neocortical neurons to temporally modulated noisy inputs in vitro, *Cerebral cortex*, *18*(9), 2086–2097, 2008.

Korcsak-Gorzo, A., Simulated tempering in spiking neural networks, Master's thesis, Heidelberg University, 2017.

Körding, K., Decision theory: what" should" the nervous system do?, *Science*, *318*(5850), 606–610, 2007.

Krizhevsky, A., G. Hinton, et al., Learning multiple layers of features from tiny images, *Tech. rep.*, Citeseer, 2009.

Krizhevsky, A., I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Krotov, D., and J. J. Hopfield, Unsupervised learning by competing hidden units, *Proceedings of the National Academy of Sciences*, *116*(16), 7723–7731, 2019.

Kullback, S., and R. A. Leibler, On information and sufficiency, *The annals of mathematical statistics*, *22*(1), 79–86, 1951.

Kumar, A., S. Schrader, A. Aertsen, and S. Rotter, The high-conductance state of cortical networks, *Neural computation*, *20*(1), 1–43, 2008.

Kungl, A. F., Robust learning algorithms for neural networks, Ph.D. thesis, Heidelberg University — Kirchhoff Institute for Physics, 2020 - in prep.

Kungl, A. F., D. Dold, O. Riedler, W. Senn, and M. A. Petrovici, Deep reinforcement learning in a time-continuous model, *Bernstein Conference*, doi:10.12751/nncn.bc2019.0168, 2019a.

Kungl, A. F., et al., Accelerated physical emulation of bayesian inference in spiking neural networks, *Frontiers in Neuroscience*, *13*, 1201, 2019b.

Kuśmierz, Ł., T. Isomura, and T. Toyoizumi, Learning with three factors: modulating hebbian plasticity with errors, *Current opinion in neurobiology*, *46*, 170–177, 2017.

La Camera, G., A. Rauch, D. Thurbon, H.-R. Luscher, W. Senn, and S. Fusi, Multiple time scales of temporal response in pyramidal and fast spiking cortical neurons, *Journal of neurophysiology*, *96*(6), 3448–3464, 2006.

Landau, L., and E. Lifshitz, Classical mechanics, 1959.

Lapique, L., Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization, *J Physiol Pathol Gen*, *9*, 620–635, 1907.

Larkum, M., A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex, *Trends in neurosciences*, *36*(3), 141–151, 2013.

Larkum, M. E., J. J. Zhu, and B. Sakmann, A new cellular mechanism for coupling inputs arriving at different cortical layers, *Nature*, *398*(6725), 338, 1999.

Larkum, M. E., W. Senn, and H.-R. Lüscher, Top-down dendritic input increases the gain of layer 5 pyramidal neurons, *Cerebral cortex*, *14*(10), 1059–1070, 2004.

LeCun, Y., D. Touresky, G. Hinton, and T. Sejnowski, A theoretical framework for back-propagation, in *Proceedings of the 1988 connectionist models summer school*, vol. 1, pp. 21–28, CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, *86*(11), 2278–2324, 1998.

LeCun, Y., Y. Bengio, and G. Hinton, Deep learning, *nature*, *521*(7553), 436–444, 2015.

Ledergerber, D., and M. E. Larkum, Properties of layer 6 pyramidal neuron apical dendrites, *Journal of Neuroscience*, *30*(39), 13,031–13,044, 2010.

Lee, J. H., T. Delbruck, and M. Pfeiffer, Training deep spiking neural networks using back-propagation, *Frontiers in neuroscience*, *10*, 508, 2016.

Lee, J.-H., et al., Review of candidate devices for neuromorphic applications, in *ESSDERC 2019-49th European Solid-State Device Research Conference (ESSDERC)*, pp. 22–27, IEEE, 2019.

Leinweber, M., D. R. Ward, J. M. Sobczak, A. Attinger, and G. B. Keller, A sensorimotor circuit in mouse cortex for visual flow predictions, *Neuron*, *95*(6), 1420–1432, 2017.

Leng, L., R. Martel, O. Breitwieser, I. Bytschok, W. Senn, J. Schemmel, K. Meier, and M. A. Petrovici, Spiking neurons with short-term synaptic plasticity form superior generative networks, *Scientific reports*, *8*(1), 10,651, 2018.

Leshno, M., V. Y. Lin, A. Pinkus, and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, *Neural networks*, *6*(6), 861–867, 1993.

Li, H. L., and M. C. Van Rossum, Energy efficient synaptic plasticity, *BioRxiv*, p. 714055, 2019.

Li, Y., J. Bradshaw, and Y. Sharma, Are generative classifiers more robust to adversarial attacks?, *arXiv preprint arXiv:1802.06552*, 2018.

Lillicrap, T. P., and K. P. Kording, What does it mean to understand a neural network?, *arXiv preprint arXiv:1907.06374*, 2019.

Lillicrap, T. P., and A. Santoro, Backpropagation through time and the brain, *Current opinion in neurobiology*, *55*, 82–89, 2019.

Lillicrap, T. P., D. Cownden, D. B. Tweed, and C. J. Akerman, Random synaptic feedback weights support error backpropagation for deep learning, *Nature communications*, *7*, 13,276, 2016.

Linnainmaa, S., The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors, *Master's Thesis (in Finnish), Univ. Helsinki*, pp. 6–7, 1970.

Liu, Q., M. Allamanis, M. Brockschmidt, and A. Gaunt, Constrained graph variational autoencoders for molecule design, in *Advances in Neural Information Processing Systems*, pp. 7795–7804, 2018.

Liu, S.-C., J. Kramer, G. Indiveri, T. Delbra, R. Douglas, et al., *Analog VLSI: circuits and principles*, MIT press, 2002.

Logothetis, N. K., and J. D. Schall, Neuronal correlates of subjective visual perception, *Science*, *245*(4919), 761–763, 1989.

Ma, W. J., J. M. Beck, P. E. Latham, and A. Pouget, Bayesian inference with probabilistic population codes, *Nature neuroscience*, *9*(11), 1432, 2006.

Maass, W., Networks of spiking neurons: the third generation of neural network models, *Neural networks*, *10*(9), 1659–1671, 1997.

Maass, W., Searching for principles of brain computation, *Current Opinion in Behavioral Sciences*, *11*, 81–92, 2016.

Maaten, L. v. d., and G. Hinton, Visualizing data using t-sne, *Journal of Machine Learning Research*, *9*(Nov), 2579–2605, 2008.

Mainen, Z. F., and T. J. Sejnowski, Reliability of spike timing in neocortical neurons, *Science*, *268*(5216), 1503, 1995.

Markram, H., M. Toledo-Rodriguez, Y. Wang, A. Gupta, G. Silberberg, and C. Wu, Interneurons of the neocortical inhibitory system, *Nature reviews neuroscience*, *5*(10), 793, 2004.

Markram, H., W. Gerstner, and P. J. Sjöström, Spike-timing-dependent plasticity: a comprehensive overview, *Frontiers in synaptic neuroscience*, *4*, 2, 2012.

Markram, H., et al., Reconstruction and simulation of neocortical microcircuitry, *Cell*, *163*(2), 456–492, 2015.

Marr, D., Vision: A computational investigation into the human representation and processing of visual information, *San Francisco: W. H. Freeman*, 1982.

Mayr, C., S. Hoeppner, and S. Furber, Spinnaker 2: A 10 million core processor system for brain simulation and machine learning, *arXiv preprint arXiv:1911.02385*, 2019.

McCloskey, M., and N. J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in *Psychology of learning and motivation*, vol. 24, pp. 109–165, Elsevier, 1989.

McKee, S. A., et al., Reflections on the memory wall., in *Conf. Computing Frontiers*, p. 162, 2004.

Mead, C., Neuromorphic electronic systems, *Proceedings of the IEEE*, *78*(10), 1629–1636, 1990.

Mehring, C., U. Hehl, M. Kubo, M. Diesmann, and A. Aertsen, Activity dynamics and propagation of synchronous spiking in locally connected random networks, *Biological cybernetics*, *88*(5), 395–408, 2003.

Meunier, D., R. Lambiotte, and E. T. Bullmore, Modular and hierarchically modular organization of brain networks, *Frontiers in neuroscience*, *4*, 200, 2010.

Mikolov, T., A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato, Learning longer memory in recurrent neural networks, *arXiv preprint arXiv:1412.7753*, 2014.

Miller, D. L., Reconfigurable systems: a potential solution to the von neumann bottleneck, 2011.

Mink, J. W., R. J. Blumenschine, and D. B. Adams, Ratio of central nervous system to body metabolism in vertebrates: its constancy and functional basis, *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, *241*(3), R203–R212, 1981.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, Playing atari with deep reinforcement learning, *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., et al., Human-level control through deep reinforcement learning, *Nature*, *518*(7540), 529–533, 2015.

Moskovitz, T. H., A. Litwin-Kumar, and L. Abbott, Feedback alignment in deep convolutional networks, *arXiv preprint arXiv:1812.06488*, 2018.

Mostafa, H., Supervised learning based on temporal coding in spiking neural networks, *IEEE transactions on neural networks and learning systems*, *29*(7), 3227–3235, 2017.

Moustafa, A. A., R. D. McMullan, B. Rostron, D. H. Hewedi, and H. H. Haladjian, The thalamus as a relay station and gatekeeper: relevance to brain disorders, *Reviews in the neurosciences*, *28*(2), 203–218, 2017.

Muldoon, S. F., E. W. Bridgeford, and D. S. Bassett, Small-world propensity and weighted brain networks, *Scientific reports*, *6*, 22,057, 2016.

Murase, M., Linear feedback shift register, uS Patent 5,090,035, 1992.

Natschläger, T., W. Maass, and H. Markram, The" liquid computer": A novel strategy for real-time computing on time series, *Special issue on Foundations of Information Processing of TELEMATIK*, *8*(ARTICLE), 39–43, 2002.

Neftci, E., S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, Event-driven contrastive divergence for spiking neuromorphic systems, *Frontiers in neuroscience*, *7*, 272, 2014.

Neftci, E. O., B. U. Pedroni, S. Joshi, M. Al-Shedivat, and G. Cauwenberghs, Stochastic synapses enable efficient brain-inspired learning machines, *Frontiers in neuroscience*, *10*, 241, 2016.

Ng, A., What artificial intelligence can and can't do right now, *Harvard Business Review*, *9*, 2016.

Ng, A. Y., Feature selection, l 1 vs. l 2 regularization, and rotational invariance, in *Proceedings of the twenty-first international conference on Machine learning*, p. 78, ACM, 2004.

Nicola, W., and C. Clopath, The dance of the interneurons: How inhibition facilitates fast compressible and reversible learning in hippocampus, *BioRxiv*, p. 318303, 2018.

Nielsen, M. A., and I. Chuang, Quantum computation and quantum information, 2002.

Nøkland, A., Direct feedback alignment provides learning in deep neural networks, in *Advances in neural information processing systems*, pp. 1037–1045, 2016.

O'Connor, P., and M. Welling, Deep spiking networks, *arXiv preprint arXiv:1602.08323*, 2016.

Oh, J.-H., and H. S. Seung, Learning generative models with the up propagation algorithm, in *Advances in Neural Information Processing Systems*, pp. 605–611, 1998.

Orbán, G., P. Berkes, J. Fiser, and M. Lengyel, Neural variability and sampling-based probabilistic representations in the visual cortex, *Neuron*, *92*(2), 530–543, 2016.

Oster, M., R. Douglas, and S.-C. Liu, Computation with spikes in a winner-take-all network, *Neural computation*, *21*(9), 2437–2465, 2009.

Palmer, S. E., O. Marre, M. J. Berry, and W. Bialek, Predictive information in a sensory population, *Proceedings of the National Academy of Sciences*, *112*(22), 6908–6913, 2015.

Pecevski, D., L. Buesing, and W. Maass, Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons, *PLoS computational biology*, *7*(12), e1002,294, 2011.

Pei, J., et al., Towards artificial general intelligence with hybrid tianjic chip architecture, *Nature*, *572*(7767), 106–111, 2019.

Peters, A., Number of neurons and synapses in primary visual cortex, in *Cerebral cortex*, pp. 267–294, Springer, 1987.

Petreanu, L., T. Mao, S. M. Sternson, and K. Svoboda, The subcellular organization of neocortical excitatory connections, *Nature*, *457*(7233), 1142, 2009.

Petrovici, M. A., *Form Versus Function: Theory and Models for Neuronal Substrates*, Springer, 2016.

Petrovici, M. A., J. Bill, I. Bytschok, J. Schemmel, and K. Meier, Stochastic inference with spiking neurons in the high-conductance state, *Phys. Rev. E*, *94*, 042,312, doi:10.1103/PhysRevE.94.042312, 2016.

Pfeiffer, M., and T. Pfeil, Deep learning with spiking neurons: opportunities and challenges, *Frontiers in neuroscience*, *12*, 2018.

Pfister, J.-P., P. Dayan, and M. Lengyel, Synapses with short-term plasticity are optimal estimators of presynaptic membrane potentials, *Nature neuroscience*, *13*(10), 1271, 2010.

Pineda, F. J., Generalization of back-propagation to recurrent neural networks, *Physical review letters*, *59*(19), 2229, 1987.

Pontes-Filho, S., and M. Liwicki, Bidirectional learning for robust neural networks, *arXiv preprint arXiv:1805.08006*, 2018.

Pozzi, I., S. Bohté, and P. Roelfsema, A biologically plausible learning rule for deep learning in the brain, *arXiv preprint arXiv:1811.01768*, 2018.

Probst, D., M. A. Petrovici, I. Bytschok, J. Bill, D. Pecevski, J. Schemmel, and K. Meier, Probabilistic inference in discrete spaces can be implemented into networks of lif neurons, *Frontiers in computational neuroscience*, *9*, 2015.

Rao, R. P., and D. H. Ballard, Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects, *Nature neuroscience*, *2*(1), 79, 1999.

Rauch, A., G. La Camera, H.-R. Lüscher, W. Senn, and S. Fusi, Neocortical pyramidal cells respond as integrate-and-fire neurons to in vivo–like input currents, *Journal of neurophysiology*, *90*(3), 1598–1612, 2003.

Richards, B. A., et al., A deep learning framework for neuroscience, *Nature neuroscience*, *22*(11), 1761–1770, 2019.

Riley, H. N., The von neumann architecture of computer systems, *Computer Science Department, California State Polytechnic University*, 1987.

Roberts, A., and B. M. Bush, *Neurones without impulses: their significance for vertebrate and invertebrate nervous systems*, vol. 6, Cambridge University Press, 1981.

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

Roth, M., Predictive stochastic inference: From abstract models to neuromorphic implementation, Bachelor thesis, Heidelberg University, 2014.

Roy, K., A. Jaiswal, and P. Panda, Towards spike-based machine intelligence with neuromorphic computing, *Nature*, *575*(7784), 607–617, 2019.

Rudy, B., G. Fishell, S. Lee, and J. Hjerling-Leffler, Three groups of interneurons account for nearly 100% of neocortical gabaergic neurons, *Developmental neurobiology*, *71*(1), 45–61, 2011.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, *nature*, *323*(6088), 533–536, 1986.

Sacramento, J., R. P. Costa, Y. Bengio, and W. Senn, Dendritic cortical microcircuits approximate the backpropagation algorithm, in *Advances in Neural Information Processing Systems*, pp. 8721–8732, 2018.

Salakhutdinov, R., Learning deep boltzmann machines using adaptive mcmc, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 943–950, 2010.

Salakhutdinov, R., and G. Hinton, Deep boltzmann machines, in *Artificial Intelligence and Statistics*, pp. 448–455, 2009.

Sarpeshkar, R., Analog versus digital: extrapolating from electronics to neurobiology, *Neural computation*, *10*(7), 1601–1638, 1998.

Scellier, B., and Y. Bengio, Equilibrium propagation: Bridging the gap between energy-based models and backpropagation, *Frontiers in computational neuroscience*, *11*, 24, 2017.

Scellier, B., and Y. Bengio, Equivalence of equilibrium propagation and recurrent backpropagation, *Neural computation*, *31*(2), 312–329, 2019.

Schemmel, J., J. Fieres, and K. Meier, Wafer-scale integration of analog neural networks, in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pp. 431–438, IEEE, 2008.

Schemmel, J., D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, A wafer-scale neuromorphic hardware system for large-scale neural modeling, in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 1947–1950, IEEE, 2010.

Schiller, J., G. Major, H. J. Koester, and Y. Schiller, Nmda spikes in basal dendrites of cortical pyramidal neurons, *Nature*, *404*(6775), 285, 2000.

Schiller, P. H., B. L. Finlay, and S. F. Volman, Short-term response variability of monkey striate neurons, *Brain research*, *105*(2), 347–349, 1976.

Schmidhuber, J., Deep learning in neural networks: An overview, *Neural networks*, *61*, 85–117, 2015.

Schmitt, S., et al., Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system, in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2227–2234, IEEE, 2017.

Schrimpf, M., et al., Brain-score: Which artificial neural network for object recognition is most brain-like?, *BioRxiv*, p. 407007, 2018.

Schuman, C. D., T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, A survey of neuromorphic computing and neural networks in hardware, *arXiv preprint arXiv:1705.06963*, 2017.

Sejnowski, T. J., *The deep learning revolution*, MIT Press, 2018.

Sejnowski, T. J., The unreasonable effectiveness of deep learning in artificial intelligence, *Proceedings of the National Academy of Sciences*, 2020.

Seung, H. S., Learning continuous attractors in recurrent networks, in *Advances in neural information processing systems*, pp. 654–660, 1998.

Shadlen, M. N., and W. T. Newsome, The variable discharge of cortical neurons: implications for connectivity, computation, and information coding, *Journal of neuroscience*, *18*(10), 3870–3896, 1998.

Shai, A. S., C. A. Anastassiou, M. E. Larkum, and C. Koch, Physiology of layer 5 pyramidal neurons in mouse primary visual cortex: coincidence detection through bursting, *PLoS computational biology*, *11*(3), 2015.

Silver, D., et al., Mastering the game of go with deep neural networks and tree search, *nature*, *529*(7587), 484, 2016.

Silver, D., et al., Mastering the game of go without human knowledge, *Nature*, *550*(7676), 354, 2017.

Silver, D., et al., A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, *Science*, *362*(6419), 1140–1144, 2018.

Sjöström, J., and W. Gerstner, Spike-timing dependent plasticity, *Spike-timing dependent plasticity*, *35*(0), 0–0, 2010.

Sjöström, P. J., and M. Häusser, A cooperative switch determines the sign of synaptic plasticity in distal dendrites of neocortical pyramidal neurons, *Neuron*, *51*(2), 227–238, 2006.

Sjöström, P. J., G. G. Turrigiano, and S. B. Nelson, Rate, timing, and cooperativity jointly determine cortical synaptic plasticity, *Neuron*, *32*(6), 1149–1164, 2001.

Snowden, R. J., S. Treue, and R. A. Andersen, The response of neurons in areas v1 and mt of the alert rhesus monkey to moving random dot patterns, *Experimental Brain Research*, *88*(2), 389–400, 1992.

Song, S., P. J. Sjöström, M. Reigl, S. Nelson, and D. B. Chklovskii, Highly nonrandom features of synaptic connectivity in local cortical circuits, *PLoS biology*, *3*(3), e68, 2005.

Spicher, D., C. Clopath, and W. Senn, Functional stdp-like plasticity under somato-dendritic prediction error learning, *Bernstein Conference*, doi:10.12751/nncn.bc2016.0208, 2016.

Spicher, D., C. Clopath, and W. Senn, Robust predictive plasticity in dendrites: From a computational principle to experimental data and back, human Brain Project Subproject 9 Fürberg Workshop, Fürberg, 2018.

Spicher, D., C. Clopath, and W. Senn, Robust predictive plasticity in dendrites: From a computational principle to experimental data, *D9.4.1 CDP5-Biological deep learning: Results for SGA2 Year 1 (M12)*, 2019.

Spoerer, C. J., P. McClure, and N. Kriegeskorte, Recurrent convolutional neural networks: a better model of biological object recognition, *Frontiers in psychology*, *8*, 1551, 2017.

Spruston, N., Pyramidal neurons: dendritic structure and synaptic integration, *Nature Reviews Neuroscience*, *9*(3), 206, 2008.

Srowig, A., J.-P. Loock, K. Meier, J. Schemmel, H. Eisenreich, G. Ellguth, and R. Schüffny, Analog floating gate memory in a 0.18 μm single-poly cmos process, *Internal FACETS documentation*, 2007.

Sutskever, I., O. Vinyals, and Q. Le, Sequence to sequence learning with neural networks, *Advances in NIPS*, 2014.

Sutton, R. S., and A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, Intriguing properties of neural networks, *arXiv preprint arXiv:1312.6199*, 2013.

Takahashi, N., T. G. Oertner, P. Hegemann, and M. E. Larkum, Active cortical dendrites modulate perception, *Science*, *354*(6319), 1587–1590, 2016.

Tang, H., M. Schrimpf, W. Lotter, C. Moerman, A. Paredes, J. O. Caro, W. Hardesty, D. Cox, and G. Kreiman, Recurrent computations for visual pattern completion, *Proceedings of the National Academy of Sciences*, *115*(35), 8835–8840, 2018.

Teeter, C., et al., Generalized leaky integrate-and-fire models classify multiple neuron types, *Nature communications*, *9*(1), 709, 2018.

Thorpe, S., and J. Gautrais, Rank order coding, in *Computational neuroscience*, pp. 113–118, Springer, 1998.

Thorpe, S., D. Fize, and C. Marlot, Speed of processing in the human visual system, *nature*, *381*(6582), 520, 1996.

Thorpe, S., A. Delorme, and R. Van Rullen, Spike-based strategies for rapid processing, *Neural networks*, *14*(6-7), 715–725, 2001.

Thrun, S., and L. Pratt, *Learning to learn*, Springer Science & Business Media, 2012.

Tieleman, T., Training restricted boltzmann machines using approximations to the likelihood gradient, in *Proceedings of the 25th international conference on Machine learning*, pp. 1064–1071, ACM, 2008.

Tolhurst, D. J., J. A. Movshon, and A. F. Dean, The statistical reliability of signals in single neurons in cat and monkey visual cortex, *Vision research*, *23*(8), 775–785, 1983.

Tsodyks, M., T. Kenet, A. Grinvald, and A. Arieli, Linking spontaneous activity of single cortical neurons and the underlying functional architecture, *Science*, *286*(5446), 1943–1946, 1999.

Tsodyks, M. V., and H. Markram, The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability, *Proceedings of the National Academy of Sciences*, *94*(2), 719–723, 1997.

Turing, A. M., On computable numbers, with an application to the entscheidungsproblem, *Proceedings of the London mathematical society*, *2*(1), 230–265, 1937.

Turrigiano, G., Homeostatic synaptic plasticity: local and global mechanisms for stabilizing neuronal function, *Cold Spring Harbor perspectives in biology*, *4*(1), a005,736, 2012.

Uhlenbeck, G. E., and L. S. Ornstein, On the theory of the brownian motion, *Physical review*, *36*(5), 823, 1930.

Urban-Ciecko, J., and A. L. Barth, Somatostatin-expressing neurons in cortical networks, *Nature Reviews Neuroscience*, *17*(7), 401, 2016.

Urbanczik, R., and W. Senn, Learning by the dendritic prediction of somatic spiking, *Neuron*, *81*(3), 521–528, 2014.

Van Rullen, R., R. Guyonneau, and S. J. Thorpe, Spike times make sense, *Trends in neurosciences*, *28*(1), 1–4, 2005.

Van Vreeswijk, C., and H. Sompolinsky, Chaos in neuronal networks with balanced excitatory and inhibitory activity, *Science*, *274*(5293), 1724–1726, 1996.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is all you need, in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Vinyals, O., et al., Grandmaster level in starcraft ii using multi-agent reinforcement learning, *Nature*, pp. 1–5, 2019.

Vogels, R., W. Spileers, and G. A. Orban, The response variability of striate cortical neurons in the behaving monkey, *Experimental brain research*, *77*(2), 432–436, 1989.

Vogels, T. P., K. Rajan, and L. F. Abbott, Neural network dynamics, *Annu. Rev. Neurosci.*, *28*, 357–376, 2005.

Vogels, T. P., H. Sprekeler, F. Zenke, C. Clopath, and W. Gerstner, Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks, *Science*, *334*(6062), 1569–1573, 2011.

Von Neumann, J., First draft of a report on the edvac (republished 1993), *IEEE Annals of the History of Computing*, *15*(4), 27–75, 1945.

Von Neumann, J., and R. Kurzweil, *The computer and the brain (unfinished book, republished 2012)*, Yale University Press, 1958.

Vreeswijk, C. v., and H. Sompolinsky, Chaotic balanced state in a model of cortical circuits, *Neural computation*, *10*(6), 1321–1371, 1998.

Wainwright, M. J., and E. P. Simoncelli, Scale mixtures of gaussians and the statistics of natural images, in *Advances in neural information processing systems*, pp. 855–861, 2000.

Waldrop, M. M., The chips are down for moore's law, *Nature News*, *530*(7589), 144, 2016.

Wang, J., H. He, and D. V. Prokhorov, A folded neural network autoencoder for dimensionality reduction, *Procedia Computer Science*, *13*, 120–127, 2012.

Wang, J. X., Z. Kurth-Nelson, D. Kumaran, D. Tirumala, H. Soyer, J. Z. Leibo, D. Hassabis, and M. Botvinick, Prefrontal cortex as a meta-reinforcement learning system, *Nature neuroscience*, *21*(6), 860–868, 2018.

Wang, S. S.-H., J. R. Shultz, M. J. Burish, K. H. Harrison, P. R. Hof, L. C. Towns, M. W. Wagers, and K. D. Wyatt, Functional trade-offs in white matter axonal scaling, *Journal of Neuroscience*, *28*(15), 4047–4056, 2008.

Werbos, P. J., Applications of advances in nonlinear sensitivity analysis, in *System modeling and optimization*, pp. 762–770, Springer, 1982.

Werbos, P. J., et al., Backpropagation through time: what it does and how to do it, *Proceedings of the IEEE*, *78*(10), 1550–1560, 1990.

Whittington, J. C., and R. Bogacz, An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity, *Neural computation*, *29*(5), 1229–1262, 2017.

Whittington, J. C., and R. Bogacz, Theories of error back-propagation in the brain, *Trends in cognitive sciences*, 2019.

Wiener, N., Generalized harmonic analysis, *Acta mathematica*, *55*(1), 117–258, 1930.

Williams, R. J., and D. Zipser, A learning algorithm for continually running fully recurrent neural networks, *Neural computation*, *1*(2), 270–280, 1989.

Wilmes, K. A., and C. Clopath, Inhibitory microcircuits for top-down plasticity of sensory representations, *Nature communications*, *10*(1), 1–10, 2019.

Wu, J., Y. Chua, M. Zhang, Q. Yang, G. Li, and H. Li, Deep spiking neural network with spike count based learning rule, *arXiv preprint arXiv:1902.05705*, 2019a.

Wu, J., Y. Chua, M. Zhang, Q. Yang, G. Li, and H. Li, Deep spiking neural network with spike count based learning rule, *CoRR*, *abs/1902.05705*, 2019b.

Wulf, W. A., and S. A. McKee, Hitting the memory wall: implications of the obvious, *ACM SIGARCH computer architecture news*, *23*(1), 20–24, 1995.

Wunderlich, T., et al., Demonstrating advantages of neuromorphic computation: a pilot study, *Frontiers in Neuroscience*, *13*, 260, 2019.

Wyatte, D., T. Curran, and R. O'Reilly, The limits of feedforward vision: Recurrent processing promotes robust object recognition when objects are degraded, *Journal of Cognitive Neuroscience*, *24*(11), 2248–2261, 2012.

Xia, Q., and J. J. Yang, Memristive crossbar arrays for brain-inspired computing, *Nature materials*, *18*(4), 309–323, 2019.

Xie, X., and H. S. Seung, Equivalence of backpropagation and contrastive hebbian learning in a layered network, *Neural computation*, *15*(2), 441–454, 2003.

Yamins, D. L., and J. J. DiCarlo, Using goal-driven deep learning models to understand sensory cortex, *Nature neuroscience*, *19*(3), 356, 2016.

Yamins, D. L., H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo, Performance-optimized hierarchical models predict neural responses in higher visual cortex, *Proceedings of the National Academy of Sciences*, *111*(23), 8619–8624, 2014.

Yao, P., H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, Fully hardware-implemented memristor convolutional neural network, *Nature*, (577), 641–646, 2020.

Yarom, Y., and J. Hounsgaard, Voltage fluctuations in neurons: signal or noise?, *Physiological reviews*, *91*(3), 917–929, 2011.

Zador, A., Impact of synaptic unreliability on the information transmitted by spiking neurons, *Journal of Neurophysiology*, *79*(3), 1219–1229, 1998.

Zeldenrust, F., W. J. Wadman, and B. Englitz, Neural coding with bursts—current state and future perspectives, *Frontiers in computational neuroscience*, *12*, 48, 2018.

Zenk, M., Spatio-temporal predictions with spiking neural networks, Master's thesis, Universität Heidelberg, 2018.

Zenke, F., and S. Ganguli, Superspike: Supervised learning in multilayer spiking neural networks, *Neural computation*, *30*(6), 1514–1541, 2018.

Zhong, G., X. Ling, and L.-N. Wang, From shallow feature learning to deep learning: Benefits from the width and depth of deep architectures, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(1), e1255, 2019.

Zhu, J.-Y., P. Krähenbühl, E. Shechtman, and A. A. Efros, Generative visual manipulation on the natural image manifold, in *European Conference on Computer Vision*, pp. 597–613, Springer, 2016.

Zoschke, K., M. Güttler, L. Böttcher, A. Grübl, D. Husmann, J. Schemmel, K. Meier, and O. Ehrmann, Full wafer redistribution and wafer embedding as key technologies for a multi-scale neuromorphic hardware cluster, in *2017 IEEE 19th Electronics Packaging Technology Conference (EPTC)*, pp. 1–8, IEEE, 2017.

Zylberberg, A., S. Dehaene, P. R. Roelfsema, and M. Sigman, The human turing machine: a neural framework for mental programs, *Trends in cognitive sciences*, *15*(7), 293–300, 2011.

Notation

Throughout Chapter 3, bold font is used to denote vectors $\boldsymbol{u}$ and matrices $\boldsymbol{w}$. Normal font with indices are used for vector or matrix components, e.g., the membrane potential of neuron j in a network is denoted by u_j. In the main text, if not otherwise stated, upper indices denote samples, i.e., $\boldsymbol{y}^i$ is the i'th sample of state vector $\boldsymbol{y}$. Boltzmann parameters are given as $\boldsymbol{W}$ and $\boldsymbol{b}$ and the corresponding biological parameters as $\boldsymbol{w}$ and $\boldsymbol{E_l}$. For correlation functions, we use $\mathcal{C}$ and for correlation coefficients ρ. $\delta(\cdot)$ is the Dirac delta distribution.

In Chapter 5, we dropped the bold notation for vectors and matrices. Lower indices denote the area number, i.e., u_j is a vector containing the membrane potentials of all neurons in area j. Similarly, W_j is a matrix containing all weights of neurons projecting from area $j-1$ into area j. In case of generative weights, G_j are all weights projecting from area $j+1$ into area j. A bar generally denotes low-pass filtering with time constant τ, e.g., $\bar{r}$ is the low-pass of r. In parts of this chapter, 'P', 'I' and 'd' indices are used to mark **P**yramidal, **I**nterneuron and **d**endritic variables. A is used to denote actions, E for energy functions and C for cost functions. Feedback weights are given by B, forward and lateral weights by W, generative weights by G – with corresponding upper indices.

In general, $\bar{r}(t) = \varphi\left(u(t)\right)$ is calculated using the activation function φ depending on the current membrane potential values $u(t)$.

Acronyms

ACF autocorrelation function.

AdEx adaptive exponential leaky integrate-and-fire.

AGI artificial general intelligence.

AI artificial intelligence.

backprop backpropagation-of-errors algorithm.

BPTT backpropagation through time.

CC correlation coefficient.

CMOS complementary metal-oxide-semiconductor.

CoBa conductance-based.

CuBa current-based.

FMP free membrane potential.

FPGA field-programmable gate arrays.

GABA γ-aminobutyric acid.

HCS high-conductance state.

iEEG intracranial electroencephalography.

LIF leaky integrate-and-fire.

MCMC Markov chain Monte Carlo.

MLE maximum likelihood estimates.

MLP multilayer perceptron.

OU Ornstein-Uhlenbeck.

PSP postsynaptic potential.

RMS root mean square.

SST somatostatin-expressing.

STDP spike time dependent plasticity.

t-SNE t-distributed stochastic neighbor embedding.

TSO Tsodyks-Markram.

V visual.

Index

Abstract neural network, 5
Action potential, 9
AdEx model, 10
Adversarial attack, 7
Apical dendrite, 10, 76
Apical error representation, 78
Artificial general intelligence, 6
Autocorrelation function, 42, 112
Autoencoder, 83
Axon, 8, 10
Axon hillock, 11

Backprop. action potential, 11, 68, 96
Balanced network, 20
Basal dendrite, 10, 68
Bayesian brain hypothesis, 21
Bayesian inference, 21
Bi-stable perception, 24
Boltzmann machine, 27
Bottleneck area, 83
BrainScaleS–1, 16, 55, 94, 121
BrainScaleS–2, 94

Calibration scheme, 47
Catastrophic forgetting, 73
Colored noise, 106
Conductance-based synapses, 9
Convex gating, 84
Correlation coefficient, 43
Cortical areas, 13, 64
Cortical layers, 10
Cost function, 6, 64, 67, 82, 89
Cost minimization, 6, 64, 68, 82
CPU and GPU power consumption, 14

Credit assignment problem, 62
Cross-correlation function, 113
Current-based synapses, 9

Decoder, 84
Discounted future voltage, 72
Dorsal pathway, 13
Dreaming, 29, 50

Effective membrane potential, 109
Encoder, 84
Ergodicity, 22
Error backpropagation, 6, 64, 69, 74, 80
Error backpropagation through time, 6
Euler-Lagrange equation, 72, 126
Excitatory synapse, 9

Fixed-pattern noise, 15
Free membrane potential distr., 105

GABA, 9, 12
Gaussian Scale Mixtures, 24
Gibbs sampling, 27
Glutamate, 9
Gradient-ascent with momentum, 34
Greedy areawise training, 60

Hebbian learning, 49, 62, 96
HICANN chip, 16, 121
Hidden neurons, 6, 30
High-conductance state, 26
Hodgkin-Huxley model, 72, 125
Human brain power consumption, 14

Inhibitory synapse, 9
Interneurons, 12, 76

L2 regularization, 34
Label neurons, 6, 67
Lagrange multiplier, 66
Latent neurons, 84
Latent space arithmetics, 88
Lateral microcircuit, 12, 78, 82
Leaky integrate-and-fire model, 9
Leaky integrator, 8, 73
Least action principle, 72, 76, 78, 82
Logistic activation function, 26
Loihi, 14, 94
Lookahead operator, 72

Markov Chain Monte Carlo, 22
Marr's levels of analysis, 3, 58, 63
Martinotti cells, 12
Maximum likelihood learning, 29
Metropolis-Hastings algorithm, 23
Mismatch energy, 67
Mixing problem, 36
Mixing synapses, 38
MLE estimate, 69
Multilayer perceptron, 6
Myelination, 10

Neural code, 67, 81
Neural sampling, 24
Neuromorphic hardware, 2, 14, 55
Neuronal response variability, 19
Neurotransmitters, 9

Ornstein-Uhlenbeck process, 25, 88

Partition function, 22
Persistent contrastive divergence, 34
Plasticity scheme, 47
Plateau potentials, 11
Poisson noise, 20, 25, 106
Postsynaptic potential, 106
Pre- and postsynaptic, 9
Prediction error, 67, 69, 74, 80, 82
Predictive coding, 3, 66, 94
Predictive voltage, 72
Probabilistic population codes, 24
Proposal distribution, 23

Pyramidal neurons, 10, 68

Receptive field, 35
Reconstruction error, 88
Refractory period, 9
Reinforcement learning, 40, 92
Renewing synapses, 37
Restricted Boltzmann machine, 30
Reversal potential, 9

Self-consistent ensemble, 48
Self-predictive state, 78
Shallow learning, 62
Short-term plasticity, 36
Simulated tempering, 36
Soma, 9, 11, 68
Spike, 1, 9, 17
Spike-based sampling, 25, 28
Spontaneous activity encodes prior, 24
SST interneurons, 12
STDP, 61, 68, 96
Synaptic noise, 20
Synthetic gradients, 17, 66, 80

t-SNE, 52
Target nudging, 69, 71, 76, 77
Transition probability, 22
Truncated error backpropagation, 130
Tsodyks-Markram model, 36

Universal function approximators, 63
Unsupervised learning, 83

Variational autoencoder, 88
Ventral pathway, 13
Visible neurons, 6, 30
Von Neumann bottleneck, 13

Wafer-scale system, 16
Wake-sleep learning rule, 29, 60
Winner-nudges-all circuit, 92