Connection weight changes and learning dynamics in models of neural networks

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
Christian Klos
aus
Groß-Gerau

Bonn, 2021

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Raoul-Martin Memmesheimer
2. Gutachter:	Prof. Dr. Dr. h.c. Ulf-G. Meißner
Tag der Promotion:	04.04.2022
Erscheinungsjahr:	2022

Abstract

The brain can be considered as a complex system of interacting neurons. As many other complex systems, neural circuits can be understood as networks of linked model units. The dynamics of the model units are often on their own rather simple. However, many interesting phenomena emerge from their interaction. In the case of neural network models, these interactions are typically mediated by weighted connections. The weights specify the strength of the coupling between the simple dynamical systems that model the neurons. In the brain, the connection sites between neurons are called synapses and are subject to a realm of plasticity mechanisms that affect their properties. Yet, the roles that synaptic plasticity plays for the functioning of neural circuits are in general poorly understood.

In this thesis, we use neural network models to numerically and analytically study four aspects of biologically inspired forms of connection weight changes. First, synaptic plasticity can effectively change the dynamics of neural networks and thus underlies many types of learning. We develop a novel scheme which uses weight changes to even endow neural networks with the ability to learn with static weights. Such dynamical learning is faster and less laborious than weight learning and thus has a high potential for applications in physics, biology and engineering. We illustrate our scheme by constructing networks that can dynamically learn dynamics ranging from simple oscillations to chaotic dynamical systems. Further, we analyze the underlying network mechanisms using dynamical systems theory. Second, recent results indicate that seemingly random weight changes are a ubiquitous phenomenon in the brain. A learning method called weight perturbation, which performs a random search in weight space, could be the cause for such plasticity. However, it is widely considered to perform poorly due to the high dimensionality of the weight space. By taking the temporal extension and the typically low dimensionality of neural dynamics into account, we show numerically and analytically that it performs much better than expected in biologically realistic settings. Third, while weight changes can allow the learning of new tasks, they may also interfere with previously acquired memories that are stored in the weights. For example, weight changes may destroy strongly connected assemblies of neurons that represent an associative memory. We show how this can be avoided in a model where noisy weight changes only lead to switches of neurons between different assemblies. These noise-induced transitions between meta-stable states can be tracked by the network, thus avoiding the forgetting of the memory. To further elucidate the underlying network mechanisms, we construct a random walk model of the weight dynamics. Fourth, synaptic plasticity is affected by neurological diseases such as epilepsy. Based on experimental data, we construct a model of a network motif that is potentially important for the spread of epileptic seizures. In doing so, we determine how short-term synaptic plasticity, which affects the synapses of the motif, and other network properties change in epilepsy. In addition, we predict how these changes influence the spread of epilepsy-associated activity.

List of publications

[1] **C. Klos**, Y. F. Kalle Kossio, S. Goedeke, A. Gilra and R.-M. Memmesheimer
Dynamical Learning of Dynamics
Phys. Rev. Lett. **125** (2020) 088103

[2] P. Züge, **C. Klos** and R.-M. Memmesheimer
Weight perturbation learning outperforms node perturbation on broad classes of temporally extended tasks
bioRxiv (2021):2021.10.04.463055

[3] Y. F. Kalle Kossio, S. Goedeke*, **C. Klos*** and R.-M. Memmesheimer
(* equal contribution)
Drifting assemblies for persistent memory: Neuron transitions and unsupervised compensation
PNAS **118** (2021) e2023832118

[4] L. Pothmann*, **C. Klos***, O. Braganza*, S. Schmidt, O. Horno, R.-M. Memmesheimer, H. Beck
(* equal contribution)
Altered Dynamics of Canonical Feedback Inhibition Predicts Increased Burst Transmission in Chronic Epilepsy
Journal of Neuroscience **39** (2019) 8998–9012

Contents

CHAPTER 1

Introduction

The brain is the seat of intelligence in higher animals. It underlies perception, learning, reasoning and complex behavior. Its capabilities, especially the generality and flexibility of human intelligence, remain largely unmatched by artificial intelligence. Further, many severe diseases, such as epilepsy and Alzheimer's, affect the brain. This has motivated great research efforts to understand its functioning. Today, we know that the basic functional units of the brain are cells called neurons, which communicate with each other via electrical impulses called spikes or action potentials [5, 6]. The mostly unidirectional connection sites between neurons are termed synapses and its properties determine the effect a presynaptic neuron can exert on a postsynaptic neuron. Furthermore, almost all neurons are either purely excitatory or inhibitory. Excitatory neurons facilitate and inhibitory neurons suppress the spiking of the neurons they project to. Importantly, the neural networks in the brain are not static. This holds in particular for the connections between neurons. Individual synapses can appear or disappear and their strength is variable. On the other hand, neurons are not newly created after birth apart from a few rare exceptions. However, the number of neurons declines with age. Further, neurons can be lost and their properties altered as a result of injury or disease. Due to the vast complexity of the brain, ranging from the intricate molecular machinery at a single synapse to the interplay of up to billions of neurons [7] organized in plastic neural networks, it has become clear that a full understanding of the brain cannot come from the biological sciences alone but necessitates tools from theoretical disciplines such as physics, computer science and mathematics.

An important and widely used tool are networks [8]. Networks consist of a set of units, called nodes or vertices, that are joined together. Mostly, the links are between pairs of nodes and are called edges or simply connections. Many different systems, for example power grids, the World Wide Web, social networks and neural circuits, can be represented as networks. In the case of neural circuits, nodes correspond to individual or groups of neurons and the connections between them to synapses [9, 10]. In most neural networks[1], individual units exhibit rather simple dynamics and the connections between them are unidirectional and weighted, i.e. a real number, termed the connection weight, is assigned to each directed connection and determines the coupling strength between the neurons it joins together. In other words, neural networks are, usually nonlinear, dynamical systems of, typically many, coupled differential equations. An important motivation for the use of neural networks is the assumption that the brain's information processing and dynamical properties largely emerge from the interaction of neurons. This is a defining feature of all complex systems [11]. Hence, one often

[1] For simplicity, we mostly omit explicitly mentioning that we refer to a model of a biological entity and not the entity itself.

omits many intricate biological details of neurons, which simplifies simulations and enables analytical investigations (e.g. [12–15]). Neural networks come in many different variations: They differ, for example, in how neuronal activity is modeled, as discrete spikes or continuous spike rates, or in the dynamics of the neuronal connections, which can be static or plastic. Apart from being models of the neural circuits in the brain, neural networks form the basis of a majority of the modern approaches in machine learning [16, 17]. Interestingly, insights from machine learning research have also influenced neuroscience [18, 19]. For example, it has been shown that neural networks originally developed for image classification can quite accurately explain the responses of neurons in the visual system of the brain [20]. Thus, neural networks have proven numerous times that they are helpful and arguably necessary to understand the workings of the brain.

In this thesis, we use neural network models to study different forms of synaptic plasticity and their roles in the functioning of neural circuits. Synaptic plasticity refers to changes of synaptic properties and is a ubiquitous phenomenon in the brain [5, 6, 9, 10]. Much experimental and theoretical research on synaptic plasticity has focused on activity-dependent forms of it. They are activity-dependent in the sense that the synaptic changes depend on the spiking activity of pre- and postsynaptic neurons. Often, one primarily considers modifications of the strength of a synapse, which corresponds to the connection weight in network models. Activity-dependent synaptic plasticity is commonly assumed to underlie learning in the brain. An important example is spike-timing dependent plasticity (STDP) [21], which modifies synaptic strengths based on the relative timing of spikes of pre- and postsynaptic neurons. Besides activity-dependent synaptic plasticity, spontaneous synaptic plasticity, i.e. synaptic changes that are independent of neuronal activity, can be observed in the brain. Recent work has shown that such plasticity can be of similar magnitude as activity-dependent plasticity [22–24]. Its function, however, is much less clear. Furthermore, neurological diseases can be associated with changes in the number and properties of both neurons and synapses. Taken together, synaptic plasticity has a strong influence on the functioning of the brain. Yet, many open questions remain. The goal of this thesis is to extend our understanding on the workings, functions and consequences of biologically inspired forms of weight changes in neural networks with the help of methods from physics. In the following, we outline the specific questions we address.

In Chapter 3, we consider the role of connection weight changes for learning with little availability of time or experience. In biology, such learning is potentially necessary to quickly attain new movements [25] or to store temporal sequences in short-term memory [26]. Learning via synaptic plasticity is hard to reconcile with such quick learning, however, as experiments on STDP show that it can take several minutes for synaptic strengths to change [27]. Further, a recent experiment in monkeys indicates quick learning without weight changes [28]. Simply speaking, neural network learning can often be thought of as follows: Distinguished output neurons, which, for example, control a muscle, have to exhibit appropriate dynamics for the network to achieve the task at hand. During learning, the network receives some form of feedback on its task performance. The learning mechanism uses this feedback to change the network dynamics such that the output exhibits the appropriate, possibly input dependent, dynamics. After the feedback has stopped, the network should still be able to achieve the task. Usually, the feedback is used to change the connection weights. However, as neural networks can exhibit very general dynamics [29], it is also possible to include a learning algorithm in the dynamics of a network with static weights [30, 31]. To differentiate such learning from learning via weight modifications, we refer to it as dynamical learning. Such dynamical learning is potentially very fast. Yet, it has only recently attracted attention as a biologically plausible learning mechanism [32, 33]. In addition, previous approaches on dynamical learning suffer from problems with the stability

of the learned network dynamics, which they circumvent by continuously providing the network with feedback on how it performs [33–46]. This substantially restricts the applicability of these approaches, however, as a central goal of learning is to make such feedback unnecessary. Hence, we develop and analyze a novel scheme to perform dynamical learning that stabilizes the network dynamics after learning without the necessity of continuous target feedback. To illustrate our scheme, we consider a standard model for neural networks, which is related to spin glasses and exhibits rich dynamics [13]. We show that a low-rank correction of its weights endows it with the ability to learn to generate required target dynamics. In particular, our scheme can be used to quickly learn output dynamics ranging from simple oscillations to chaotic dynamical systems. Furthermore, we carefully analyze the underlying network mechanisms using dynamical systems theory. Besides providing a novel candidate mechanism for learning in the brain, our approach may also be useful for other applications in physics, such as the prediction of chaotic dynamics [47, 48] and physical reservoir computing [49–53]. Briefly, physical reservoir computing is a computational framework for temporal data processing using high-dimensional systems that can consist of a wide range of physical substrates such as photons and electron spins.

In Chapter 4, we turn to activity-independent weight changes and the question if they have a function. They may be a result of weight perturbation (WP) [54, 55], which is an often overlooked learning mechanism. WP and the closely related node perturbation (NP) [56, 57] use random perturbations of connection weights and neuronal activity, respectively, for learning. Their underlying idea is to correlate the perturbations with the change in performance of the perturbed neural network compared to the unperturbed network or an estimate of it. Then the weights are updated such that they reproduce or contrast the perturbations depending on the sign and magnitude of the performance difference. However, WP has been widely disregarded as a potential learning mechanism as it is considered to perform badly, in particular relative to NP [19, 56–62]. The underlying reasoning is based on the fact that weights drastically outnumber nodes in typical neural networks. Thus, the perturbation dimensions or, equivalently, the dimension of the search space, is much larger for WP compared to NP, which supposedly leads to slower learning for WP. We show that this argument ceases to hold when considering that tasks are extended in time and that they are low-dimensional, two properties that are common in biologically relevant settings. Specifically, we derive analytical expressions for the error dynamics of WP and NP for simple, time-extended tasks with the help of Wick's theorem, which is commonly used in field theories. We find that WP performs much better than expected, even outperforming NP in biologically relevant situations. We also show numerically that the analytical results largely extend to more complex networks and tasks. Thus, our results suggest learning via WP as a potential function of activity-independent weight changes.

In Chapter 5, we consider the effects of continuously ongoing synaptic plasticity. The basic variant of STDP is constantly changing synaptic weights even if there is no obvious task to be learned. The same holds for spontaneous synaptic plasticity. As memories are thought to be stored in the connection weights, this poses the question of how they can persist. Similarly, recent experimental observations show that neural representations change over time, i.e. that memories are represented by different ensembles of neurons at different points in time [63–65]. Such ensembles of neurons form the basis of a standard memory model, where they are termed neural assemblies [10, 66]. Assemblies are characterized by large intra- and small inter-assembly connection weights. So far, it has been assumed that they are static, i.e., that they consist of the same neurons at different points in time [67]. We show that, given appropriate plasticity mechanisms, neurons can switch between assemblies, thereby giving rise to the observed representational drift. Importantly, the plasticity

mechanisms together with the network activity also keep the assemblies consistently connected to the same in- and output neurons, thus preventing the forgetting of memories. From a physics perspective, the weight dynamics of a single neuron can be considered to be governed by a potential, whose minima correspond to the different assemblies in the network. In previous models, these minima would be stable, while in our model they are meta-stable with noise-induced transitions between the minima [68]. To demonstrate the viability of our approach, we use network simulations. To further elucidate the network mechanisms underlying the neuron switching, we construct a simplified random walk model of the weight dynamics and approximate it by a diffusion process. This also allows us to semi-analytically compute the stationary probability density of the weights. In addition, it allows us to show that the noise of the weight changes without including their drift already suffices to generate meta-stable states. This phenomenon is known as noise-induced multistability, which can also be observed in other systems such as electrical oscillations and foraging behavior [69–72].

Finally, in Chapter 6, we study how short-term synaptic plasticity and other neuron properties are affected in epilepsy. Epilepsy is defined to be the predisposition to have epileptic seizures, which result from abnormal excessive or synchronous neuronal activity [73, 74]. It exists in many different variants, one of which being temporal lobe epilepsy. In temporal lobe epilepsy, seizures originate from one of the subregions of the medial temporal lobe of the brain and subsequently spread to large parts of the brain. Temporal lobe epilepsy accounts for about 25 % of all cases of epilepsy [74]. Yet, the underlying network mechanisms are not well understood. In physiological conditions, inhibition is crucial for controlling excitatory activity, especially during oscillatory activity [75]. As epilepsy is characterized by pathological high-frequency oscillations, it could be that altered inhibition plays a significant role in epilepsy. Previous work indicates that, for example, the cell number [76] and the number of excitatory neuron-targeting synapses [77] of some types of inhibitory neurons is indeed reduced. However, the effect of these and other epilepsy-associated changes on network activity, especially during seizures, is unclear. This holds in particular for the functioning of feedback inhibition, i.e. local recurrent loops between excitatory and inhibitory neurons, and synaptic plasticity that acts on fast timescales and thus may be important for the spread of seizures. Hence, we investigate how the dynamics of excitatory-inhibitory feedback-loops are affected in epilepsy. Therefore, we consider the CA1 region of the hippocampus, which is part of the temporal lobe and is important for the spread of seizures (see Introduction in ref. [78]). Based on experimental data obtained from epileptic animals, we construct a model of the feedback-loops. The connection weights in our model are additionally modulated by fast variables that capture the effect of short-term synaptic plasticity. This allows us to estimate neuronal and synaptic properties that were not directly attainable in the experiment. Importantly, it also allows us to predict how the feedback-loops respond to external stimulation. In particular, we probe the network model with stimuli representing input from upstream regions. We find that the epilepsy-induced changes indeed appear to facilitate the spread of epileptic bursts. Thus, altered feedback inhibition might be an important factor for the spreading of epileptic activity.

This thesis is structured as follows. In Chapter 2, we summarize fundamental and recent findings from neurobiology that are relevant for this thesis, provide a brief introduction into neural network modeling and introduce important terminology. Thereafter, we show how weight learning can be used to construct networks that learn dynamically in Chapter 3. In Chapter 4, we present our analysis of perturbation-based learning methods. Afterwards, in Chapter 5, we show how weight changes due to noisy spiking give rise to drifting assemblies while preserving memories. In Chapter 6, we present our model for altered feedback inhibition in epilepsy together with the underlying experimental results. Finally, in Chapter 7, we summarize our results and give an outlook.

CHAPTER 2

Foundations

In this chapter, we summarize foundations of and recent findings from neurobiology and neural network modeling that underlie this thesis and introduce important notations. We start with some of the fundamental findings of neurobiology, primarily following the textbooks by Kandel et al. [5] and Bear et al. [6]. Specifically, we describe the functioning of neurons, how they communicate with each other via synapses, how synaptic connections change over time and the organization of connectivity and activity in populations of neurons. As the properties of all of these aspects vary across species, brain regions and neuron type, we focus on the mammalian cortex and the most common neurons therein. The cortex underlies the highest cognitive functions [5, 6]. Most of the described findings extend at least qualitatively to other species and brain regions, however. Thereafter, we introduce some of the mathematical tools used for neural network modeling and learning, primarily following the textbooks by Dayan and Abbott [9] and Gerstner et al. [10]. We describe spike- and rate-based point neuron models, current-based synapse models, different plasticity mechanisms, some fundamental network models and different types of learning. These tools will be extensively used in the remaining chapters.

2.1 Neurobiology

2.1.1 Neurons

Neurons are specialized cells that underlie the transmission and processing of spikes in the brain. Most types of neurons consist of three parts: Dendrites, which are branch-like processes that receive inputs from other neurons, mostly at small protrusions called spines, and route them to the soma. The soma or cell body, which integrates the incoming signals over space and time and generates spikes. Finally, the axon, which is a projection that conducts the spikes to other neurons or muscles.

There exist many different types of neurons. Generally, they can be divided into excitatory and inhibitory neurons. Excitatory neurons constitute most of the principal neurons, whose axons often project to brain regions other than the region where their somas are located. Inhibitory neurons polarize their targets. They constitute most of the interneurons, whose axons are restricted to the brain region where the neuron is located [79]. The most common type of excitatory neurons in the cortex are pyramidal cells, which possess a characteristic pyramidal shape (Fig. 2.1a). Inhibitory neurons are much more diverse. They can be classified based on their morphology, the expression of

certain proteins or other molecules and their firing patterns [80]. A prominent inhibitory cell type are basket cells. They possess a distinctive branch-like axon, which encloses the somas of pyramidal cells (Fig. 2.1b). In the hippocampal region CA1, they have a strong impact on the activity of excitatory neurons [75, 81], likely because of their axonal structure and their abundance [82]. In Chapter 6, we examine their role in epilepsy. Other examples for inhibitory neurons are bistratified cells, trilaminar cells and oriens-lacunosum moleculares (OLM) cells (Fig. 2.1c) [80]. These neurons also occur in CA1 and target the dendrites of pyramidal cells. Their effect on the spiking of their target neurons is more indirect by controlling the excitatory input arriving at the dendrites [79].

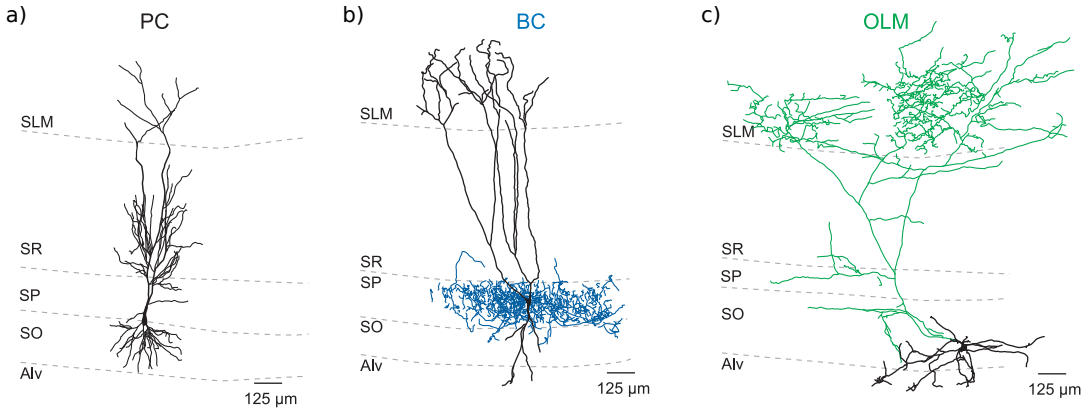

Figure 2.1: Morphological reconstructions of different cell types in the CA1 of rats.
Specifically, a pyramidal cell (PC, a), basket cell (BC, b) and OLM cell (c) is shown. For the BC and OLM cell, axons are colored in blue and green, respectively, and dendrites in black. SLM: stratum lacunosum-moleculare, SR: stratum radiatum, SP: stratum pyramidiale, SO: stratum oriens, Alv: Alveus. Figure adapted from ref. [4], see also Chapter 6.

Neurons are electrically charged, i.e., there is a potential difference between the in- and outside of the cell membrane. This membrane potential results from concentration gradients of a range of different ions, which are established and maintained by ion channels and pumps. For example, the ions responsible for spiking are sodium and potassium cations. At rest, the membrane potential is about –65 mV. Neurons generate a spike if the membrane potential at the axon hillock, the part of the soma at which the axon starts, raises above a threshold, which typically lies about 10 mV above the resting potential. Specifically, a thresholds crossing leads to the opening of fast, voltage-dependent channels for sodium ions. The following influx of sodium ions depolarizes the cell even more, leading to a positive feedback process as more and more voltage-dependent sodium channels open. It is eventually stopped by delayed, temporary closing of the sodium channels and the opening of slower ion channels for potassium ions. The latter leads to an outflux of potassium ions, which hyperpolarizes the neuron, typically even below its resting potential. In summary, this process leads to an about 100 mV large and a few ms long voltage deflection that constitutes a spike, which is also known as an action potential. Directly after a neuron has spiked, the generation of another spike is virtually impossible as the sodium channels stay closed for a few ms, This period is called the absolute refractory period and is followed by the tens of ms long relative refractory period during which spiking is still less likely. Spikes have several important properties. First, they are relatively short and large, which renders them clearly distinct from any background fluctuations. Second, they are stereotypical all-or-none processes, i.e.,

all spikes of the same neuron type have near-identical time-courses. Third, they are the only type of voltage deflection that can travel along the axon over long distances to other neurons, because they are actively regenerated.

2.1.2 Synapses

Synapses are the connection sites of the axon of the presynaptic neuron and the dendrites, soma or, on rare occasions, axon of the postsynaptic neuron. Their properties largely determine the effect that an incoming spike has on the postsynaptic neuron. In addition, they are plastic and their strength is believed to store memories. There exist two types of synapses, electrical and chemical synapses. As chemical synapses are much more common in the cortex, we focus on them and refer to them only as synapse from here on.

Synapses consist of three parts: an axonal or presynaptic terminal, a specialized region of the postsynaptic neuron and an about 20 nm to 40 nm wide synaptic cleft in between (Fig. 2.2a). The transmission of signals via synapses is mediated by chemical messengers called neurotransmitters. For excitatory neurons, the most common neurotransmitter is glutamate, while for inhibitory neurons it is γ-aminobutyric acid (GABA). In the resting state of a synapse, the neurotransmitters are assembled in synaptic vesicles in the axonal terminals. A significant fraction of all vesicles cluster at the active zone of a presynaptic terminal. These vesicles constitute the readily releasable pool (RRP) of vesicles. When a spike arrives at an axonal terminal, voltage-dependent ion channels at the active zone open, which leads to an influx of calcium ions, which in turn triggers the fusing of the vesicles in the RRP with the cell membrane and, thus, the release of neurotransmitters into the synaptic cleft. This process, called exocytosis, is stochastic. It may even happen that a spike arrival at the synapse leads to barely any release of neurotransmitters, in which case one speaks of a synaptic failure. Such synaptic unreliability may underlie the perturbation-based learning methods that we study in Chapter 4 (see also [83]). The released neurotransmitters then bind to receptors at the postysnaptic membrane, which, either directly or indirectly, lead to the opening of ion channels and the influx of ions such as sodium ions. Eventually, the neurotransmitters unbind from the receptors and are transported back to the presynaptic terminal. If the presynaptic neuron is excitatory, the ion influx into the postsynaptic neuron is an excitatory postsynaptic current (EPSC) leading to an excitatory postsynaptic potential (EPSP, depolarizing voltage deflection). Accordingly, if the presynaptic neuron is inhibitory, the influx is an inhibitory postsynaptic current (IPSC) leading to an inhibitory postsynaptic potential (IPSP, hyperpolarizing voltage deflection). When the synapse is positioned at a dendrite, the EPSP or IPSP then travels to the soma, where all incoming PSPs are integrated. Mostly, PSPs deteriorate while traveling to the soma, meaning synapses located at or close to it have a stronger effect on spike generation.

2.1.3 Synaptic plasticity

Many properties of synapses are not static but change over time. Such plasticity mechanisms underlie learning and short- and long-term memory, are important to keep neuronal activity in a healthy regime, but are partly also without a known function. Among the different forms of plasticity, activity-dependent synaptic plasticity stands out as it is most widely studied and as it is considered to underlie many forms of learning and memory formation [85]. In the following, we present classical forms of activity-dependent synaptic plasticity mechanisms together with more recent results on

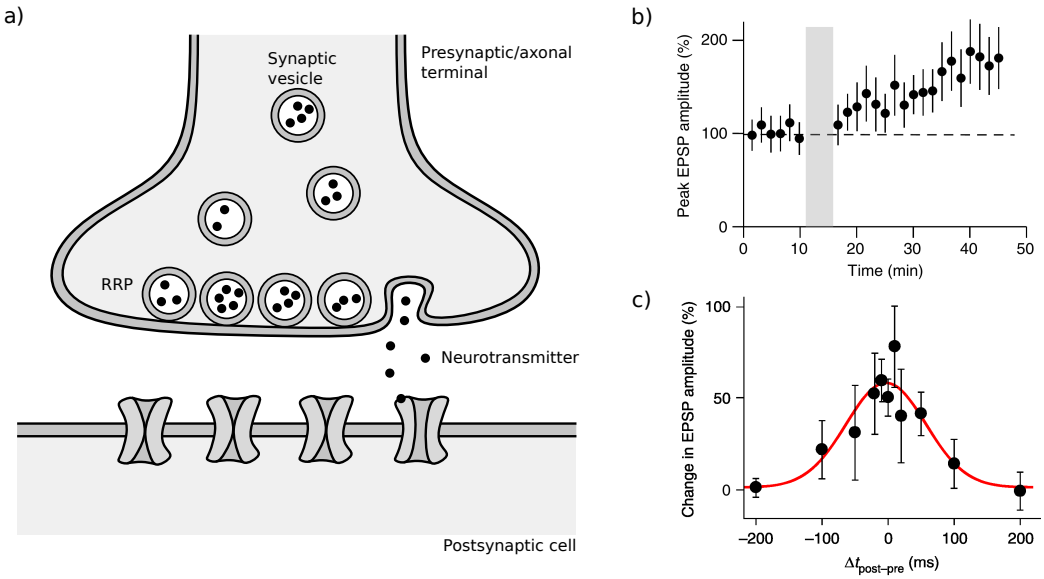

Figure 2.2: Synapses and STDP.
(a) Schematic of a synapse. Some synaptic vesicles gather in the readily releasable pool (RRP) in the presynaptic terminal. If an action potential arrives at the synapse, exocytosis is triggered, i.e. vesicles fuse with the membrane and release neurotransmitters into the synaptic cleft, as depicted on the right. The released neurotransmitters then bind to receptors on the postsynaptic membrane, which causes the opening of ion channels.
(b,c) STDP at synapses between pyramidal cells in the CA3 of rats.
(b) Repetitive spike pairings (gray bar) induce an increase of the EPSP amplitude. Note that it takes about 20 min to 30 min after the spike pairings have finished until the EPSP change is completed. Spike pairs consist of a presynaptic spike followed by a postsynaptic spike with a delay of $\Delta t_{\text{post-pre}} = 10\,\text{ms}$.
(c) Change in EPSP amplitude as a function of the time difference between post- and presynaptic spike $\Delta t_{\text{post-pre}}$. In contrast to the classical time window, where the synapse is potentiated for $\Delta t_{\text{post-pre}} > 0\,\text{ms}$ and depressed for $\Delta t_{\text{post-pre}} < 0\,\text{ms}$, here it is symmetric.
Panels (b,c) adapted from ref. [84].

spontaneous, activity-independent synaptic plasticity.

Short-term plasticity (STP) describes changes in synaptic efficacy that last on the order of hundreds to thousands of ms [86]. It depends on the history of presynaptic activity and comes in two variations. Short-term depression (STD) refers to a temporary reduction of PSP size. It is caused by the depletion of vesicles from the RRP in the axonal terminal after the arrival of one or more spikes at the synapse. If an additional spike arrives at the synapse before the RRP is refilled, there are fewer neurotransmitters available to be released into the synaptic cleft and, thus, to cause a PSP. Short-term facilitation (STF) refers to a temporary increase in PSP size. It is caused by residual calcium available in the axonal terminal resulting from previous spike arrivals. If an additional spike arrives at the synapse, the probability for the fusing of vesicles with the membrane is increased due to the higher level of available calcium. Which of STD or STF dominates depends on the types of pre- and postsynaptic neuron and the brain region. STP is thought to be important for, e.g., motor control, short-term memory and the control of population activity. In Chapter 6, we study how it is changed in epilepsy.

Long-term plasticity describes changes in synaptic efficacy that last from tens of minutes to days or even longer. As for STP, one distinguishes between long-term potentiation (LTP) and depression (LTD).

They result from, for example, the creation or removal of receptors at the postsynaptic membrane and occur in different variations. One such variation is spike-timing dependent plasticity (STDP), which refers to changes in the synaptic efficacy that depend on the precise timing of pre- and postsynaptic spikes [21, 87]. Often it is assumed that it only depends on the time difference between doublets of post- and presynaptic spikes, though it can also depend on the timing of spike triplets and other factors. The relationship between change in synaptic efficacy and spike-time difference is called the STDP or learning window. The learning window is typically largest for small time differences and decays within tens of ms. Often, the learning window is positive (negative) if the postsynaptic neuron spikes after (before) the presynaptic neuron. However, the properties of STDP depend on synapse type and brain region. In CA3, for example, the learning window for connections between excitatory neurons is symmetric (Fig. 2.2b,c) [84]. STDP thus allows neurons to learn about correlations in their input and is considered to be one of the primary mechanisms underlying learning.

Another type of long-term plasticity is synaptic scaling [88]. Synaptic scaling is triggered by changes in a neuron's average firing rate and regulates the total number of its synaptic receptors to counteract the rate changes. In addition, there is evidence that the total strength of all outgoing synapses is similarly scaled [89].

Long-term plasticity has traditionally been studied indirectly by observing how the amplitude of PSPs change after external stimulation of the pre- and postsynaptic neuron [21]. More recently, however, some experimentalists have begun to take a more direct approach and imaged dendritic spines, which indicate the existence of a synapse and whose size appears to correlate well with synaptic efficacy [22, 23, 90]. These experiments have revealed that synapses can appear newly and disappear completely, a process called structural plasticity. They also found that structural plasticity as well as the fluctuations in spine size, and thus synaptic efficacy, are ongoing even if neural activity is prohibited [22, 24]. What the function of this activity-independent plasticity is and how it can be reconciled with the proven relationship between activity-dependent synaptic plasticity and memory formation is an open research question (see Chapters 4 and 5 for possible answers to these questions).

2.1.4 Populations of neurons

So far, we considered individual neurons and synapses, but much of the capabilities of neural networks are thought to arise through their interactions. Here, we summarize the high-level organization of the cortex, general principles of neural network connectivity and common patterns of neural population activity.

The cortex is the outer mantle of the cerebrum, which makes up large parts of the brain. It consists of a few layers, defined by the neuron types in them and their connectivity, and can be divided into two parts: the neocortex, usually and from now on only referred to as cortex, and the allocortex. The neocortex, which only exists in mammals, is the evolutionary youngest part of the brain and seat of our highest cognitive functions. Although it takes over diverse functions and receives input from almost all sensory organs, it has a rather uniform structure. About 80 % of all neurons in the cortex are excitatory and 20 % are inhibitory. Their connectivity is characterized by a high degree of recurrence, i.e., starting from a single neuron there are many loops of connected neurons that end up at the starting neuron. Furthermore, each neuron receives input from thousands of others. Each excitatory neuron connects to about 10 % of all other excitatory neurons in its local neighborhood. The cortex can be further subdivided into regions with specialized function. As mentioned above, excitatory neurons in the cortex are principal cells and make connections both within and across these different regions,

while inhibitory neurons are interneurons and make only local connections.

The allocortex is evolutionary older than the cortex. It consists of the olfactory system and the hippocampus. The latter is necessary for the formation of new memories, though if memories stay hippocampus-dependent or are fully conveyed to the cortex for long-term storage is a matter of ongoing debate [74, 91]. In addition, it is crucial for spatial navigation, with recent results indicating that it is also important for navigation in non-spatial behaviorally relevant dimensions. Finally, the hippocampus is involved in some of the most common neuronal diseases such as epilepsy and Alzheimer's. The structure of the hippocampus is less uniform than that of the cortex and differs substantially between its subparts dentate gyrus (DG), cornu Amonis regions 1–3 (CA1–3) and subiculum. They are connected to each other in a directed manner: the in- and output region of the cortex that connects to the hippocampus is the entorhinal cortex. The entorhinal cortex projects to all subregions of the hippocampus via the perforant path. The DG connects to CA3 via the mossy fibers, CA3 connects to CA1 via the Schaffer collaterals, CA1 then connects to the Subiculum, which connects back to the entorhinal cortex. As CA1, and to a lesser extent CA3, are important for parts of this thesis, we explain here their structure in more detail. All CA subregions have a very similar laminar structure (Fig. 2.1). The central layer is the pyramidal cell layer, which contains the somas of pyramidal cells and some interneuron types, especially basket cells. Below it is the stratum oriens and above it are the stratum radiatum and the stratum lacunosum-moleculare. These layers contain the dendrites of the pyramidal cells as well as diverse types of interneurons. For example, the somas of OLM cells are located in the stratum oriens and its axonal terminals are mostly located in the stratum lacunosum-moleculare.

Cortical and hippocampal networks exhibit a range of different activity patterns. In awake, behaving animals, they generally fire asynchronously and irregular, i.e., neurons in a population spike at different times and the time between spikes of individual neurons is highly random [92–94]. In fact, their interspike interval distribution decays approximately exponentially for times larger than the refractory period, as for Poisson processes with homogenous rate. Typically, neurons spike a few times per second with inhibitory neurons often having a larger firing rate than excitatory neurons. On top of that and in other brain states, for example during phases of sleep, cortical and in particular hippocampal networks exhibit oscillations in different frequency bands ranging from 0.05 Hz to 500 Hz [95]. An example for such oscillations are sharp wave/ripples (SPW/Rs), which originate in CA3, from where they spread via CA1 to the cortex [75]. SPW/Rs are characterized by highly active, localized neuronal populations (sharp wave) with high-frequency oscillations of about 140 Hz to 200 Hz on top. They last about 100 ms and are important for memory consolidation and planning of future actions. While the precise mechanisms for their generation are a matter of ongoing research, they likely are a result of the recurrent interaction between pyramidal and basket cells [81].

Furthermore, some neurological diseases give rise to salient activity patterns. Epilepsy, for example, is characterized by seizures, which are pathological high-frequency oscillations entraining often large parts of the brain [73, 74]. In addition, temporal lobe epilepsy is accompanied by pathological SPW/Rs, which are typically stronger and have higher frequencies than their physiological counterpart [96–99].

2.1.5 Neural representations

We end this section with a few general observations on how behaviorally relevant stimuli and memories are represented by populations of neurons. Recordings from a variety of different brain regions have shown that neuronal representations, more precisely the manifolds in neural state space occupied by

neuronal activity for some task, have a low intrinsic as well as embedding dimension [100–104]. This means that most of the variability of neuronal activity is restricted to a low-dimensional subspace of the full neuronal state space. Remarkably, it appears that brain regions specified for a certain computation make it possible for downstream regions to read out results of this computation with a linear readout [100, 101]. For example, in a cycling task that demanded the animal to keep track of the number of cycles, the number of cycles could be read out linearly from the supplementary motor area [105]. The low dimensionality of neuronal representations is also in accordance with a classical and experimentally verified idea for the storage of memories by means of neuronal assemblies [66]. Assemblies are groups of neurons that collectively encode a well-specified concept, such as an apple. They are associative in the sense that an increased activity of only part of an assembly, maybe because only a part of an apple is visible, will activate all neurons in the assembly. Recent results, however, indicate that the individual neurons that make up an assembly change over time [63–65], posing the question on how the memory is not forgotten. In Chapter 5, we provide a possible solution based on compensatory learning induced by STDP and synaptic scaling.

2.2 Neural network modeling

2.2.1 Single-neuron and synapse models

There exist many different models for single neurons differing in their complexity and in how accurately they capture the dynamics of real neurons. One distinguishes between single-compartment (or point neuron) models and multi-compartment models. Single-compartment models largely ignore the morphological structure of neurons and describe the membrane potential of a neuron by a single variable. Multi-compartment models use multiple variables to describe the potential of the neuron in its different sections. Further one distinguishes between spiking neurons, whose outputs are spikes, and rate neurons, whose outputs are spike rates. Thus, spiking neurons communicate via discrete signals while rate neurons communicate via continuous signals. Similarly, synapses can be modeled with various degrees of complexity, for example, synapse models can exclude or include plasticity. One typically chooses the neuron and synapse model based on how detailed the underlying biological mechanisms needs to be modeled, whether analytical calculations should be performed and how many computational resources are available for simulations. In this thesis, we use rather simple spiking and rate-based point neuron models, because we are mostly interested in the dynamics and learning capabilities arising from the interactions of neurons.

Integrate-and-fire neurons

Integrate-and-fire (IF) neurons are probably the most widely used spiking neuron models and underlie the model of Chapter 5. They model the subthreshold dynamics of the membrane potential at the soma, denoted with $V(t)$, as a leaky integration of the input, while the spiking is modeled as a separate mechanism. The simplest type of IF neuron model is the leaky integrate-and-fire (LIF) neuron. It assumes that a neuron can be considered as a simple RC-circuit: Cations accumulate at the outside surface and anions at the inside surface of the cell membrane, resembling a capacitor C. The ion channels in the membrane lead to a leak current, which can be modeled with a resistor R in parallel to the capacitance. The resting potential is held up by a battery with potential V_{rest} in line with the

resistor. Thus, the subthreshold dynamics of $V(t)$, except for the refractory period, are given by

$$\tau_m \frac{dV}{dt}(t) = V_{\text{rest}} - V(t) + R\tilde{I}(t), \tag{2.1}$$

where $\tau_m = RC$ is the membrane time constant and $\tilde{I}(t)$ is a potential input current stemming from other neurons or an external source. In this chapter, Chapter 6 and as in many research articles, we define $I(t) = R\tilde{I}(t)$ and, for simplicity, call it input current as well. When $V(t)$ reaches a threshold V_Θ, the neuron fires a spike, and $V(t)$ is reset to a reset potential V_0, where it is fixated for the duration of the refractory period τ_{ref} (Fig. 2.3a). Given that spikes are very short and have a stereotypical form, it is customary to represent them by Dirac δ-functions. A neuron's spike sequence, also called spike train, can thus be represented by a sum of δ-functions: $S_f(t) = \sum_{t_f} \delta(t - t_f)$, where the t_f are the spike or firing times, i.e. the times of threshold crossings. The LIF neuron is the most widely used integrate-and-fire neuron, as its simplicity allows for long simulations, some analytical tractability (e.g. [15]), and easy interpretability of simulation results. However, it neglects the nonlinear parts of the membrane potential dynamics of real neurons, which are especially important near the threshold and are caused by the voltage-dependent ion channels. Nonlinear IF models attempt to capture these parts by replacing the leak term $V_{\text{rest}} - V(t)$ in Eq. (2.1) with a nonlinear function $f(V(t))$. Widely used nonlinear IF models are quadratic or exponential IF models.

Current-based synapses

As for neurons, there exist different models for synapses, i.e., for the effect that spikes of presynaptic neurons have on the membrane potential of a postsynaptic neuron. Throughout this thesis, we employ current-based synapses, for which presynaptic spikes increment the input current $I(t)$. Another important synapse type are conductance-based synapses, for which presynaptic spikes increment the conductance of modeled ion channels at the synapse. For a single, current-based synapse, at which a spike train $S_s(t) = \sum_{t_s} \delta(t - t_s)$ arrives, the input current at the postsynaptic neuron obeys

$$\tau_s \frac{dI}{dt}(t) = -I(t) + \tau_s w_s S_s(t). \tag{2.2}$$

Here, τ_s is the synaptic timescale and w_s the synaptic or connection weight. Integrating Eq. (2.2) and inserting the result into Eq. (2.1) yields $I(t) = \sum_{t_s} \text{PSC}(t; t_s)$ and, assuming $V(t)$ stays below V_Θ, $V(t) = \sum_{t_s} \text{PSP}(t; t_s) + V_{\text{rest}}$, where

$$\text{PSC}(t; t_s) = w_s \exp\left(-\frac{t - t_s}{\tau_s}\right) \Theta(t - t_s) \tag{2.3}$$

is the postsynaptic current and

$$\text{PSP}(t; t_s) = \frac{w_s \tau_s}{\tau_m - \tau_s} \left(\exp\left(-\frac{t - t_s}{\tau_m}\right) - \exp\left(-\frac{t - t_s}{\tau_s}\right)\right) \Theta(t - t_s) \tag{2.4}$$

is the postsynaptic potential in response to a presynaptic spike at t_s (Fig. 2.3a). In Chapter 5, we specify w_s in terms of the peak PSP $\hat{w}_s$ a spike evokes. If the PSP is a double exponential function as in Eq. (2.4), $\hat{w}_s = \max_t \text{PSP}(t; t_s) = \left(\frac{\tau_s}{\tau_m}\right)^{\frac{\tau_m}{\tau_m - \tau_s}} w_s$.

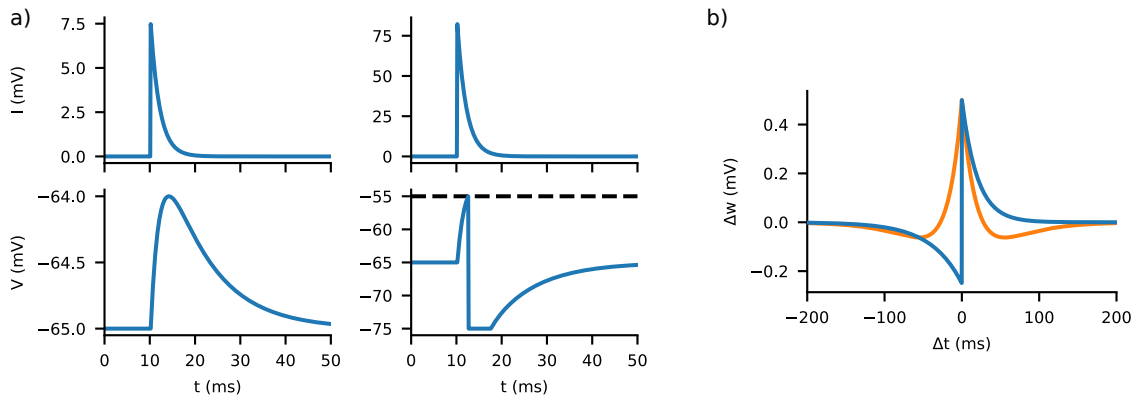

Figure 2.3: LIF neuron dynamics and STDP.
a) Dynamics of a LIF neuron in response to a single input spike. Left column shows the PSC (top) and membrane potential (bottom), i.e. the PSP plus the resting potential, in response to a single input spike assuming a connection weight of $\hat{w}_s = 1\,\text{mV}$. Right column shows the PSC (top) and membrane potential (bottom) in response to a single input spike assuming a connection weight of $\hat{w}_s = 11\,\text{mV}$. The weight is set to a very large value to drive the membrane potential across the threshold (dashed black line). Thus, the LIF neuron spikes and its membrane potential is reset to the reset potential, at which it is held constant during the refractory period. Parameters: $V_{\text{rest}} = -65\,\text{mV}$, $V_\Theta = -55\,\text{mV}$, $V_0 = -75\,\text{mV}$, $\tau_m = 10\,\text{ms}$, $\tau_s = 2\,\text{ms}$, $\tau_{\text{ref}} = 5\,\text{ms}$.
b) Typical STDP window functions for asymmetric STDP (blue) and symmetric STDP (orange). Parameters: $\eta_{\text{LTP}} = 0.5\,\text{mV}$ ($\eta_{\text{LTP}} = 1\,\text{mV}$ for symmetric STDP), $\eta_{\text{LTD}} = -0.25\,\text{mV}$ ($\eta_{\text{LTD}} = -0.5\,\text{mV}$ for symmetric STDP), $\tau_{\text{LTP}} = 20\,\text{ms}$, $\tau_{\text{LTD}} = 40\,\text{ms}$.

Short-term plasticity

In Chapter 6, we model STP with the phenomenological model by Markram and Tsodyks [86, 106]. It introduces two new variables: the fraction of available neurotransmitters in the RRP, $x(t)$, and the fraction of available neurotransmitters that is ready to be used, $u(t)$, i.e., the release probability of the available vesicles in the RRP. Hence, $x(t_s^-)u(t_s^-)$, where $f(t^-) = \lim_{s \to t^-} f(s)$, is the fraction of all possible neurotransmitters in the RRP that is released upon spike arrival at the synapse at t_s. The input current therefore obeys

$$\tau_s \frac{dI}{dt}(t) = -I(t) + \tau_s w_s x(t^-)u(t^-)S_s(t). \tag{2.5}$$

The depletion of neurotransmitters in the RRP after a spike arrival is modeled as a stepwise decrease of $x(t)$ by the released fraction of neurotransmitters, $x(t_s^-)u(t_s^-)$. The RRP is then gradually refilled, i.e. $x(t)$ grows back to its baseline value of one. Thus, $x(t)$ obeys

$$\tau_{\text{RRP}} \frac{dx}{dt}(t) = 1 - x(t) - \tau_{\text{RRP}} x(t^-)u(t^-)S_s(t), \tag{2.6}$$

where τ_{RRP} is the depression time constant. Eq. (2.6) models STD. On the other hand, the influx of calcium ions into the axonal terminal after a spike arrival is modeled by a stepwise increase of the

release probability $u(t)$, which afterwards decreases exponentially back to its baseline value:

$$\tau_{\text{fac}} \frac{du}{dt}(t) = u_0 - u(t) + \tau_{\text{fac}} u_f (1 - u(t^-)) S_s(t). \tag{2.7}$$

Here, τ_{fac} is the facilitation time constant, u_0 is the asymptotic release fraction and u_f determines the increase of $u(t)$ after a spike arrival. Both u_0 and u_f are restricted to lie between 0 and 1, which together with the factor $(1 - u(t^-))$ ensures that $0 \leq u(t) \leq 1$ and $0 \leq x(t) \leq 1$. Eq. (2.7) models STF. Note that in the original STP model by Markram and Tsodyks [86, 106], u_f is equal to u_0, which is not always justifiable from experimental data. Like for synapses without STP, one often gives w_s in terms of the peak PSP a spike evokes assuming $x(t) = 1$ and $u(t) = u_0$: $\hat{w}_s = \left(\frac{\tau_s}{\tau_m}\right)^{\frac{\tau_m}{\tau_m - \tau_s}} \frac{w_s}{u_0}$.

Long-term plasticity

If one models long-term plasticity, synaptic weights become time-dependent. Since such weight changes are typically happening either slowly or stepwise, one often does not make the time-dependence explicit, i.e. one writes w_s instead of $w_s(t)$. Models for long-term plasticity are much more diverse compared to models for STP, which is why we here only mention two commonly used models for STDP and synaptic scaling that are relevant for Chapter 5. First, we consider pair-based STDP with all-to-all spike interaction [21]. For this mechanism all pairs of pre- and postsynaptic spikes lead to a stepwise change of the weight by $\Delta w(\Delta t)$, where $\Delta w(\cdot)$ is the STDP window function and Δt is the time difference of the spike pair. Specifically, if the presynaptic (postsynaptic) neuron spikes, the sum of these weight updates of pairs involving the current spike and all previous spikes of the postsynaptic (presynaptic) neuron is added to the weight. Thus, the weight changes according to

$$\frac{dw_s}{dt}(t) = S_s(t) \left(\int_{-\infty}^{t} ds \Delta w(s - t) S_f(s) \right) + S_f(t) \left(\int_{-\infty}^{t} ds \Delta w(t - s) S_s(s) \right). \tag{2.8}$$

For asymmetric STDP (Fig. 2.3b blue line), the STDP window function is typically

$$\Delta w(\Delta t) = \eta_{\text{LTP}} \exp\left(-\frac{|\Delta t|}{\tau_{\text{LTP}}}\right) \Theta(\Delta t) + \eta_{\text{LTD}} \exp\left(-\frac{|\Delta t|}{\tau_{\text{LTD}}}\right) \Theta(-\Delta t). \tag{2.9}$$

For symmetric STDP (Fig. 2.3b orange line), which we employ in Chapter 5, it is often

$$\Delta w(\Delta t) = \eta_{\text{LTP}} \exp\left(-\frac{|\Delta t|}{\tau_{\text{LTP}}}\right) + \eta_{\text{LTD}} \exp\left(-\frac{|\Delta t|}{\tau_{\text{LTD}}}\right). \tag{2.10}$$

Here, $\eta_{\text{LTP}} \geq 0$ and $\eta_{\text{LTD}} \leq 0$ determine the window amplitudes and τ_{LTP} and τ_{LTD} the timescales of potentiation and depression. In network simulations, using such STDP rules without restrictions usually produce instabilities. This is because strongly connected neurons exhibit more correlated spiking, which leads to further strengthening of their connection weights and ultimately unbounded growth of the weights. Another problem is the potential loss of selective responsiveness to different presynaptic neurons or, more generally, different input spike patterns due to independent growth of all synapses. A solution to the first problem is the introduction of bounds for w_s. A solution to the second problem, which partly also solves the first problem, is provided by homeostatic plasticity mechanisms that introduce synaptic competition. A particularly important example of such a mechanism is weight

normalization, which is a simple model of synaptic scaling as observed in the cortex. It ensures that the sum of the weights of all incoming or, less often, outgoing connections of a neuron stays at least approximately equal to some value w_{sum} at all times. This is achieved by scaling the weights by w_{sum} divided by the sum of all incoming or outgoing connection weights. Such scaling may be applied in regular intervals, typically ranging from seconds to minutes, or after each weight update. Note, however, that it is a matter of debate if such timescales, which are likely necessary to ensure the stability of networks with realistic STDP rules, are not too fast to be considered biologically plausible [107, 108]. In the model of Chapter 5, we perform input and output normalization after each weight update.

Rate neurons

Rate neurons, which we use in Chapters 3, 4 and 6, output and communicate with each other via continuous firing rates. Compared to spiking neurons, this usually makes them simpler to simulate and more amenable to theoretical analysis [9]. In addition, they are advantageous for population-averaged simulations of groups of neurons as performed in Chapter 6. Specifically, to simplify network simulations one often represents neurons whose activities are sufficiently correlated with a single model unit. Averaging the rates of the summarized neurons and representing the average by a single rate neuron is straightforward. How actual spikes of the summarized neurons could be averaged is much less clear, however. In particular, due to the voltage reset and refractory period after a spike, a single spike of a spiking model unit would already be similar to the synchronous spiking of all summarized neurons. On the other hand, rate neurons cannot account for spike timing correlations, which may be crucial for the workings of populations of neurons in the brain.

In rate neurons, neuronal spike trains $S(t) = \sum_{t_s} \delta(t - t_s)$ are replaced by firing rates $r(t)$. Performing this replacement in Eq. (2.2), one gets the input current for a single synapse,

$$\tau_s \frac{dI}{dt}(t) = -I(t) + \tau_s w_s r_s(t), \tag{2.11}$$

where $r_s(t)$ is the rate of the presynaptic neuron. To get the rate of the postsynaptic neuron, there exist different possibilities. First, one can compute the membrane potential $V(t)$ using the subthreshold dynamics of IF neurons (Eq. (2.1) for LIF neurons) and subsequently compute the rate as $r(t) = f(V(t))$, where $f(V)$ is a nonlinear function. We use this approach in parts of Chapter 6, where we additionally subtract $\tau_m V(t) r(t)$ from the right-hand side of Eq. (2.1) to account for the voltage resets after spikes. $f(V)$ is typically a rectified linear ($f(V) = \max(0, \alpha(V - V_\Theta))$ with $\alpha > 0$) or softplus ($f(V) = \hat{r} \log\left(1 + \exp\left(\frac{V - V_\Theta}{V_w}\right)\right)$ with $\hat{r}, V_w > 0$) function.

Second, one can compute the rate directly from the input current as $r(t) = \sigma(I(t))$, where $\sigma(I)$ is a nonlinear, often sigmoidal function. This approach is used in Chapters 3, 4 and 6. Neglecting the potentially important voltage dynamics can be justified in situations where the input current varies less rapidly than the membrane potential would. There are several reasons for using this approach: The frequency-current curve (f-I curve), i.e. the rate of a neuron in response to a constant input current, is an often measured relation in single-cell experiments. It saturates for large input currents due to the refractoriness of neurons, which motivates the use of sigmoidal functions for the computation of the spike rate. Further, it includes fewer parameters and is amenable to analytical considerations. In particular, it is popular for more abstract studies of neuronal network dynamics (Chapters 3 and 4),

where it is usually used with the following notation with dimensionless quantities:

$$\tau \frac{dx}{dt}(t) = -x(t) + w r_s(t), \tag{2.12}$$

$$r(t) = \sigma(x(t)). \tag{2.13}$$

Most neuron models employed in machine learning are technically also rate neurons. However, they are only distantly related to real neurons because of their high level of abstraction.

2.2.2 Neural network models

Models of single neurons can be connected to form neural network models. While the properties of individual neurons and synapses even of the same type vary in the brain, one usually models at most a few neuron types each possessing the same parameters. To model the input from multiple presynaptic neurons, one typically assumes that the input currents stemming from different presynaptic neurons sum linearly, in line with having multiple input currents in the equivalent circuit used to derive LIF neurons. Thus, for, e.g., spiking neurons with current-based synapses described by Eq. (2.2), the total input current from a population of presynaptic neurons to a postsynaptic neuron obeys

$$\tau_s \frac{dI_i}{dt}(t) = -I_i(t) + \tau_s \sum_j w_{ij} S_j(t). \tag{2.14}$$

Here, i indexes the postsynaptic and j the presynaptic neurons, w_{ij} is the weight between neurons j and i and $S_j(t)$ is the spiketrain of neuron j. For simplicity, we assume that there is no conduction delay of the spike transmission. Besides neuron, synapse and plasticity models, network connectivity is an important, distinguishing feature of neural networks. Connectivity refers to the existence and non-existence of connections but also the distribution of weights of existing connections. Important connectivity types are feedforward and recurrent connectivity. Feedforward networks consist of ordered layers of neurons, where all neurons in each layer only project to neurons in subsequent layers. Thus, external input is successively transported through all layers until it reaches the output layer. Unlike feedforward networks but as cortical networks, recurrent networks (RNNs) possess directed cycles, i.e. external input reverberates in them. In the following, we briefly introduce some of the network models used in this thesis.

Spiking excitatory-inhibitory networks and the balanced state

Many spiking models of cortical networks share the following structure and components, which capture important properties of real neural networks: They consist of two populations of LIF neurons, N_E excitatory neurons, whose outgoing weights are positive, and N_I inhibitory neurons, whose outgoing weights are negative. N_E is typically on the order of tens to tens of thousands of neurons, while N_I is roughly around 5 % to 30 % of that. The connectivity between all neurons is characterized by a high degree of recurrence. Often, network neurons additionally receive external input, which should model stimuli or input from other, not explicitly modeled, brain regions. Such input can be a constant current, randomly generated spike trains or current or voltage noise.

Depending on parameter values and connectivity, such networks can exhibit one of four types of network activity: Synchronous regular, synchronous irregular, asynchronous regular or asynchronous

irregular activity [10, 15]. Here, regularity refers to how regular individual neurons spike and synchrony to how synchronous the spiking is on a network level. As mentioned above, in their ground state, cortical neurons spike asynchronously and irregular. The question of how such activity can arise used to be a problem due to the following observations. The irregular spiking of neurons means they are driven by fluctuations in their input. However, they receive thousands of inputs from other neurons, which spike irregularly and are only weakly correlated. On first sight, input fluctuations should therefore average out and thus lead to regular spiking. A possible solution to this conundrum is what is called the balanced state [14, 15, 94, 109]. It posits that excitatory input is balanced by inhibitory input such that the average input vanishes and synaptic weights can be large enough to yield voltage fluctuations that generate spikes. In Chapter 5, we use networks with an approximate balance between excitation and inhibition.

Standard networks of rate neurons

In Chapter 3 and partly also Chapter 4, we use networks of simple rate neurons. Such networks serve as the basis of many studies concerned with more general properties of neural network dynamics and computations. For a network consisting of N neurons, the activation vector $x(t)$ obeys

$$\tau \frac{dx}{dt}(t) = -x(t) + Ar(t), \tag{2.15}$$

$$r(t) = \sigma(x(t)). \tag{2.16}$$

Here, the activation function $\sigma(x)$ is applied element-wise and A is the matrix of connection weights, weight matrix in short. Often the network neurons are augmented with additional input, output and feedback units (see below and Chapters 3 and 4). The entries of the weight matrix may be learned but are also often chosen randomly. In a prominent scheme [13], one sets the connection probability to p, i.e. an individual weight is zero with probability $(1 - p)$ or if it corresponds to a self-connection, and then draws the weights of existing connections from a Gaussian distribution with mean 0 and variance $\frac{g^2}{pN}$, where g is the so-called gain parameter. In the limit of large N, the eigenvalues of such a weight matrix are distributed in the complex plane on a disk that has radius g and that is centered at the origin. If $\sigma(x) = \tanh(x)$, the dynamics of such networks undergo a phase transition from fixed-point ($g < 1$) to chaotic ($g > 1$) behavior [13] (Fig. 2.4). In the chaotic regime, such RNNs provide rich dynamics. For example, different timescales are present in the dynamics, which can also be observed in the cortex. The RNNs do not, however, capture the separation in excitatory and inhibitory neurons (but see, e.g., ref. [110] for an alternative construction scheme for A including this separation). In addition, they allow for negative rates, which may be interpreted to represent the difference of the actual firing rate compared to some baseline rate. They further underlie a rather novel learning scheme called reservoir computing, which we will describe below.

Perceptrons

In large parts of Chapter 4, we consider perceptrons [16]. Perceptrons are feedforward neural networks and started the overwhelming success of neural networks in machine learning. In contrast to the previous examples, they neglect the temporal dynamics of neurons. Thus, their connection to real neural networks is rather abstract. Generally, a multi-layer perceptron consists of L layers of neurons,

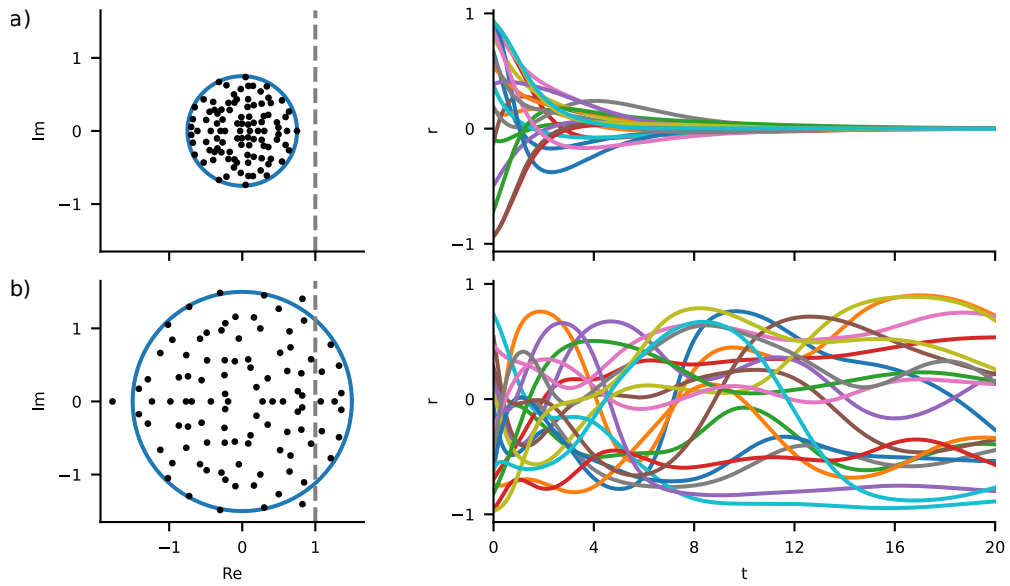

Figure 2.4: Dynamics in networks of rate neurons given by Eqs. (2.15) and (2.16) with $\tanh(x)$-nonlinearity. Such networks can exhibit fixed-point (a, $g = 0.75$) and chaotic (b, $g = 1.5$) behavior, depending on the eigenvalues of the weight matrix. The eigenvalues lie mostly inside a circle with radius g (left, blue circle). If the real part of an eigenvalue gets greater than one (left, gray dashed), the origin ($x = 0$) becomes linearly unstable and the dynamics transition from fixed-point to chaotic behavior (right, rates of twenty randomly selected network neurons). Parameters: $N = 100$, $p = 0.1$, $\tau = 1$.

which themselves are sometimes called perceptrons, plus an input layer. The rate or activation vector of all neurons in layer l, $l = 1, \ldots, L$, is given by

$$r^{(l)} = \sigma^{(l)}\left(w^{(l)} r^{(l-1)} + b^{(l)}\right), \tag{2.17}$$

where $w^{(l)}$ is the weight matrix, $b^{(l)}$ the vector of biases and $\sigma^{(l)}(\cdot)$ the activation function (applied element-wise) of layer l. $r^{(0)}$ is the input vector and $r^{(L)}$ the output vector. After learning the weight matrices, multi-layer perceptrons can be used for function-approximation and classification problems.

2.2.3 Learning in neural networks

Learning means shaping neuronal dynamics such that a network, or sometimes a single neuron, solves a given task. A typical task is that a network augmented with external input and output units should generate a certain target output depending on the input. Modifying connection weights is by far the most widely used way of shaping neuronal dynamics. Besides this weight learning, learning can be implemented, e.g., via modification of neuronal gains [111] or purely dynamical without changing parameters (see Chapter 3).

Learning can be differentiated into three categories. First, unsupervised learning shapes dynamics without any feedback on how well the network performs. STDP may act as such a learning mechanism and can be used to form assemblies and the auto-associative memories they encode. Second, supervised learning shapes dynamics based on precise feedback on how well the network performs. Specifically,

the learning rule has access to the exact target output. Third, reinforcement learning shapes dynamics based on delayed feedback in the form of reward or punishment. Thus, the network learns via trial-and-error. In the following, we briefly introduce the learning paradigms of gradient descent and reservoir computing, which are used Chapter 4 and Chapter 3, respectively.

Gradient descent

The most widely used training algorithm in machine learning is gradient descent. It can be used for all three types of learning but is most often employed for supervised learning. It updates connection weights at discrete time points by adding $\Delta w_{ij} = -\frac{\partial E}{\partial w_{ij}}$, where E is the error, to the weights. Its success in machine learning has led to the belief that the brain also implements it in some way [19]. However, it is hard to reconcile with restrictions imposed by the working of real neural networks in the brain. Specifically, gradient descent assumes that E is differentiable with respect to the weights, but the spiking leads to discontinuities. In addition, biologically plausible weight learning rules require locality, i.e. weight changes can only depend on quantities that are locally available at the synapse. Backpropagation, the most common algorithm to compute the gradients, fulfills the locality requirement by propagating the error from the output node back to network nodes [112]. However, this backward pass has to be done with the same weights as the forward pass meaning the weights have to be symmetric ($w_{ij} = w_{ji}$). Also, the backward pass should not alter neural activity arising during the forward pass and, in recurrent networks, the backward pass is computed backwards in time. Recently, many ideas, such as surrogate gradients for spiking networks, the replacement of symmetric by random weights and elaborate synapse models, have been developed to circumvent these problems (e.g. [19, 113–116]).

Reservoir computing

Another more recent approach for learning of time-dependent tasks is reservoir computing [117, 118]. Reservoir computing is a general computing paradigm specified by only a few simple rules: the input stimulates a reservoir, the reservoir's state includes a high-dimensional nonlinear transformation of its input history, the output is a linear readout of the reservoir, learning happens (almost) exclusively by adjusting the linear readout. The idea is that due to the nonlinear expansion of the input, the dynamics of the target output arise naturally in linear dimensions of the reservoir, which can be easily found during the learning of the linear readout. This idea also underlies kernel methods in machine learning and can be more formally justified by Cover's theorem, which, roughly, states that two sets of random vectors are more likely to be linearly separable if they are nonlinearly cast into a high-dimensional space [16, 119]. The reservoir can in principle be of any kind fulfilling the above stated properties. It can be a physical substrate such as photons or even a bucket of water [49], but also an RNN, for which reservoir computing was originally introduced under the name of liquid-state machines for spiking networks [120] and under the name of echo-state networks for rate networks [121, 122] based on previous work [123, 124].

Here we focus on rate networks in an example supervised learning setting, where the task is to first learn and then autonomously generate a trajectory that is periodic or generated from some dynamical system. The reservoir is an RNN of the type introduced in Eqs. (2.15) and (2.16). It consists of N neurons with connection probability p and $w_{ij} \sim \mathcal{N}(0, \frac{g^2}{pN})$ for existing connections and an additional,

M-dimensional output $z(t)$. The complete network evolves according to the following equations:

$$\tau \frac{dx}{dt}(t) = \begin{cases} -x(t) + Ar(t) + w_z \tilde{z}(t) & \text{(open loop)}, \\ -x(t) + Ar(t) + w_z z(t) & \text{(closed loop)}, \end{cases} \tag{2.18}$$

$$r(t) = \tanh(x(t)), \tag{2.19}$$

$$z(t) = o_z r(t). \tag{2.20}$$

Here, $\tilde{z}(t)$ is the target available for learning, o_z is the matrix of output weights, which is to be learned and w_z is the matrix of feedback weights. The network can operate in two modes: in the open loop setting the target drives the reservoir and in the closed loop setting the reservoir receives output feedback. The feedback of target or output is necessary for the reservoir to "echo" the target or output enabling it to predict how the target signal evolves. For learning, one can use, for example, the ridge regression error function

$$E(o_z) = \sum_{k=1}^{n} \|z(t_k) - \tilde{z}(t_k)\|_2^2 + \alpha \|o_z\|_2^2, \tag{2.21}$$

where k indexes the recording time steps, which are usually also the simulation time steps. n is the number of recording time steps during learning. The term $\alpha \|o_z\|_2^2$, where $\|\cdot\|_2$ is the Frobenius norm and $\alpha > 0$ the so-called regularization factor, regularizes the solution by biasing the output weights towards small values.

The open loop setting is used when learning happens offline, i.e. o_z is modified only after the reservoir has been driven by the target for some time. Minimizing Eq. (2.21) yields

$$o_z^T = \left(R^T R + \alpha I\right)^{-1} R^T \tilde{z}, \tag{2.22}$$

where R is the $n \times N$ design matrix, whose kth row is equal to $r^T(t_k)$ and $\tilde{z}$ is the $n \times M$-dimensional matrix of target outputs, whose kth row is equal to $\tilde{z}^T(t_k)$. A prerequisite for the offline learning to work is that the network fulfills the so-called echo-state property, which essentially states that the influence of previous input, which is in this example the target feedback, and previous reservoir states on future reservoir states should gradually vanish with time. Without input, the echo-state property is fulfilled if $g < 1$. Although external input can stabilize the dynamics for $g > 1$, in practice one often initializes the reservoir with $g < 1$ for offline learning. After learning the feedback loop is closed (closed loop setting) and the network should continue to generate the target output.

Another possibility is online learning of the output weights, with $o_z(k)$ being the current output weight matrix at t_k. A particularly successful way to do so is FORCE (first-order reduced and controlled error) learning [125]. It uses the closed loop setting already during learning, but ensures that $z(t)$ always stays close to $\tilde{z}(t)$ by means of fast and large updates to $o_z(k)$. Thus, the dynamics of the reservoir stay close to the dynamics in the open loop setting and allows the learning to converge. Nevertheless, the feedback contains small errors, which may sample previously unstable directions in state space. The learning procedure can then stabilize these directions. Because the small errors are intrinsically generated, this procedure of stabilizing output dynamics is more effective than injecting external noise into the target feedback in the case of offline learning. Furthermore, FORCE learning allows using reservoirs with $g > 1$, i.e. with chaotic activity in the absence of input, as the strong

weight updates rapidly lead to output that suppresses the chaos. One can use different weight-update algorithms for FORCE learning. A natural candidate is recursive least-squares, which is an online version of ridge regression. It can be derived from Eq. (2.22) with the help of the Woodbury matrix identity and updates the output weights according to:

$$o_z^T(k) = o_z^T(k-1) - g(k)(z(t_k) - \tilde{z}(t_k)), \tag{2.23}$$

$$g(k) = \frac{P(k-1)r(t_k)}{1 + r^T(t_k)P(k-1)r(t_k)}, \tag{2.24}$$

$$P(k) = \left(I - g(k)r^T(t_k)\right) P(k-1). \tag{2.25}$$

Here, $g(k)$ is the vector of learning rates for the N output weights. $P(k) = \left(\sum_{j=1}^{k} r(t_j)r^T(t_j) + \alpha I\right)^{-1}$ is a running estimate of the inverse of the sum of the rate correlation matrix and the regularization term. It is initialized as $P(0) = \frac{I}{\alpha}$ with α^{-1} determining the initial learning rates.

Compared to gradient descent, reservoir computing is much less computationally expensive and allows faster learning, because linear regression is a fairly simple procedure. However, as computational resources got significantly cheaper during the last decades, gradient descent has proven to be more effective in real-world applications. This might change in the future as recent research on reservoir computing tries to take advantage of the fact that the reservoir need not be changed during learning and can be of any physical substrate [49]. Photonic reservoirs, for example, are in principle much faster than silicon-based computers and have a low power consumption [50]. Another reason to study reservoirs are their relation to cortical neural networks. Cortical networks exhibit rich dynamics, which is advantageous for reservoir computing. For example, widely different timescales are present, ranging form a few ms for spikes to seconds for STP. Furthermore, as mentioned above linear outputs often suffice to read out task-relevant information from cortical networks [100, 101]. But there are also some complications with using standard reservoir computing as a model for learning in cortical networks. First, synaptic weights are, at least when one considers timescales of minutes or more, not static but plastic. Recent studies have shown, however, that using biologically plausible, unsupervised plasticity rules for connections between reservoir neurons can increase network performance [126]. Second, the well-performing learning algorithms for the output weights are not biologically plausible, because connections are updated with information that is not locally available to them. Furthermore, FORCE learning requires unrealistically strong and fast weight updates.

CHAPTER 3

Dynamical learning of dynamics

This chapter consists of the following published article:

[1] **C. Klos**, Y. F. Kalle Kossio, S. Goedeke, A. Gilra and R.-M. Memmesheimer
Dynamical Learning of Dynamics
Phys. Rev. Lett. **125** (2020) 088103
© 2020 American Physical Society

The following sections contain the complete article with minor editorial changes. My contributions were the development and simulation of the models for tasks (i-iv and vi) (Figs. 3.1 to 3.3 except Fig. 3.3c,d) including the corresponding quantification of learning performance (Section 3.A.3, except Fig. 3.9), the analysis of the learning speed (Section 3.A.5), the investigation of the robustness of learning (Section 3.A.6), the construction of a model for generalization (Section 3.A.7) and the pretraining with weight perturbation (Section 3.A.8). Further, I wrote large parts of the article and the supplemental material/appendix (Section 3.A) and helped with the analysis of the underlying network mechanisms (Figs. 3.4 and 3.11).

3.1 Introduction

Humans and animals can learn wide varieties of tasks. The predominant paradigm assumes that their neural networks achieve this by slow adaptation of connection weights between neurons [9, 10]. Neurobiological experiments, however, also indicate fast learning with static weights [28]. Our study addresses how neural networks may quickly learn to generate required output dynamics without weight-learning.

The goal of neural network learning is ultimately to appropriately change the activity of the output neurons of the network. In supervised learning, it should match a target and continue doing so during subsequent testing; in reinforcement learning, it should maximize a sparsely given reward (see Section 2.2.3). In our study, the networks adapt their weights during a pretraining phase [127–129] such that thereafter with static weights they achieve supervised learning of desired outputs, by adapting only their dynamics (dynamical learning). Adapting the static network's weights during pretraining is thus a kind of meta-learning or learning-to-learn. There is a recent spurt of interest in learning-to-learn [128,

129], focusing mainly on learning of reinforcement learning [32, 130, 131]. Studies on learning of supervised dynamical learning showed prediction of a time series at the current time step given the preceding step's target [33–35, 37–43] and control of a system along a time-varying target [36, 44–46]. The studies assume that a target is present during testing to avoid unlearning. This limits applicability and renders the dynamics necessarily non-autonomous; it is conceptually problematic for supervised settings and at odds with the common concept of teacher-free recall.

We therefore develop a scheme for fast supervised dynamical learning and subsequent teacher-free generation of long-term dynamics. We consider models for biological recurrent (reciprocally connected) neural networks, where leaky rate neurons interact in continuous time [9, 10]. Such models are amenable to learning, computation and phase space analysis [9, 10, 125, 132, 133]. After appropriate pretraining using the reservoir computing scheme (where only the weights to output neurons are trained (see Sections 2.2.3 and 3.A.1 and refs. [120, 122, 125])), all weights are fixed. The networks can nevertheless learn to generate new, desired dynamics. Furthermore, they continue to generate them in self-contained manner during subsequent testing. We illustrate this with a variety of trajectories and dynamical systems and analyze the underlying mechanisms.

3.2 Network model

We use recurrent neural networks, where each neuron (or neuronal subpopulation) i, $i = 1, ..., N$, with N between 500 and 3000 depending on the task (Section 3.A.2), is characterized by an activation variable $x_i(t)$ and communicates with other neurons via its firing rate $r_i(t)$, a nonlinear function of $x_i(t)$ [9, 10] (see Section 2.2.2). In isolation $x_i(t)$ decays to zero with a time constant τ_i. This combines the decay times of membrane potential and synaptic currents. The network has two outputs, which can be interpreted as linear neurons: signal $z_k(t)$, $k = 1, ..., N_z$, and context $c_l(t)$, $l = 1, ..., N_c$ (Fig. 3.1). After learning, $z(t)$ generates the desired dynamics while $c(t)$ indexes it. They are continually fed back to the network, allowing their autonomous generation [122]. The networks are temporarily also informed about their signal's difference from its target $\tilde{z}(t)$ by an error input $\varepsilon(t) = z(t) - \tilde{z}(t)$. Taken together, for constant weights the network dynamics are given by

$$\begin{aligned} \tau\dot{x}(t) &= -x(t) + Ar(t) + w_z z(t) + w_c c(t) \\ &\quad + w_\varepsilon \varepsilon(t) + w_u u(t), \\ z(t) &= o_z r(t),\ c(t) = o_c r(t), \end{aligned} \tag{3.1}$$

with recurrent weights A, the diagonal matrix of time constants τ, signal and context output weights o_z, o_c, feedback weights w_z, w_c, input weights w_ε, w_u and a drive $u(t)$ absent for most tasks. We choose $r_i(t) = \tanh(x_i(t) + b_i)$ [122, 125, 134]; offsets b_i are drawn from a uniform distribution between -0.2 and 0.2 and break the $x \to -x$ symmetry without input. Unless mentioned otherwise, we set $\tau_i = 1$ fixing the overall time scale. Recurrent weights A_{ij} are set to zero with probability $1 - p$ ($p = 0.1$ or $p = 0.2$ depending on the task). Nonzero weights are drawn from a Gaussian distribution with mean 0 and variance $\frac{g^2}{pN}$, where $g = 1.5$ [125]. $w_{z,ij}$, $w_{c,ij}$ and $w_{\varepsilon,ij}$, $w_{u,ij}$ are drawn from a uniform distribution between $-\tilde{w}$ and $\tilde{w}$ ($\tilde{w} = 1$ or $\tilde{w} = 2$).

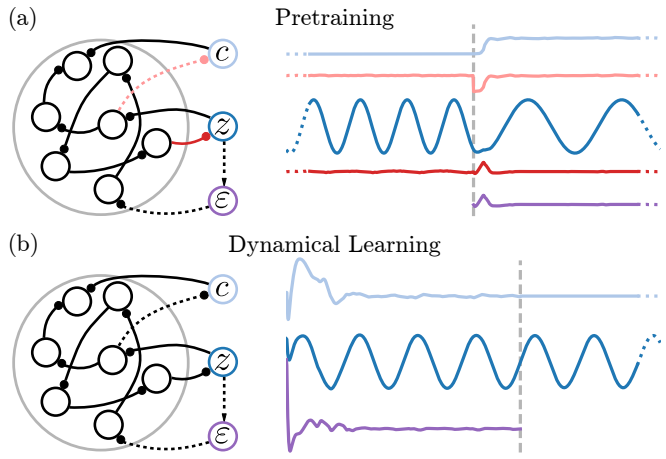

Figure 3.1: Pretraining and learning.
(a) During pretraining, the output weights (left, different reds) of the network are adapted using the output errors $\varepsilon(t) = z(t) - \tilde{z}(t)$ (right, red) and $c(t) - \tilde{c}(t)$ (light red), such that $z(t)$ (blue, different scale for clarity) and $c(t)$ (light blue) match their targets. Different members of the target family are weight-learned in the training periods (dashed vertical). At their beginnings, $\varepsilon(t)$ is fed also as input (purple).
(b) Dynamical learning. The output weights are now fixed. The network receives the signal error $\varepsilon(t)$ as input (purple). It adapts its dynamics to generate $z(t) \approx \tilde{z}(t)$ (blue). During testing, an error is no longer provided and $c(t)$ is fixed to its previous average (right, dashed vertical, left, dashed weights). $z(t)$ continues to approximate $\tilde{z}(t)$.

3.2.1 Pretraining

The aim of our pretraining (Fig. 3.1a) is twofold. First, it should enable the resulting static networks to learn signals of a specific class given only the error input $\varepsilon(t)$. Second, after removing the error input the static networks should be able to continue to generate the desired dynamics. Therefore, the networks have to learn to minimize $\varepsilon(t)$ and, as explained in the Analysis section, to associate unique contexts with the different target dynamics.

To achieve this, we present different trajectories $\tilde{z}(t)$ of the target class to the networks, together with associated, straightforwardly chosen constant indices $\tilde{c}$. The output weights $o_{z,ij}$ and $o_{c,ij}$ learn online according to the FORCE rule (see Sections 2.2.3 and 3.A.1 and ref. [125]) to minimize the output errors $\varepsilon(t)$ and $c(t) - \tilde{c}$. In short, they are modified using the supervised recursive least-squares algorithm with high learning rate. This provides a least-squares optimal regularized solution for the output weights given the past network states and targets [135]. Signals and indices are presented for a time t_{wlearn} (30000 or 50000) as a continuous, randomly repeating sequence of training periods of duration t_{stay} (between 200 and 1000). During each training period's first part, a network receives $\varepsilon(t)$ as input. Because of the various last states of the previous learning periods, it thus learns to approach $\tilde{z}(t)$ from a broad range of initial conditions given this input. In most of our tasks, after a time $t_{\text{fb}} = 100$, when $z(t)$ is close to $\tilde{z}(t)$, $\varepsilon(t)$ is switched off and $c(t)$ is fixed to its constant target, matching the testing paradigm. This often helps the network to learn generation of $z(t) \approx \tilde{z}(t)$ without error input.

3.2.2 Dynamical learning and testing

The weights now remain static and the error input teaches the network new tasks of the pretrained target class (Fig. 3.1b), i.e. the networks dynamically learn to generate $z(t) \approx \tilde{z}(t)$ for previously unseen $\tilde{z}(t)$. The learning time t_{learn} (between 50 and 200) is short, a few characteristic timescales of the target dynamics (Section 3.A.2). $c(t)$ is moderately fluctuating.

Thereafter the test phase begins, where no more teacher is present ($w_\varepsilon \to 0$). In weight-learning paradigms, during such phases the weights are fixed [110, 120, 122, 125, 136]. We likewise fix $c(t)$ to a temporally constant value, an average of previously assumed ones, $c(t) = \bar{c}$. This may be interpreted as indicating that the context is unchanged and the same signal is still desired. We find in our applications, that the network dynamics continue to generate a close-to-desired signal $z(t)$, establishing the successful dynamical learning of the task.

3.3 Applications

We illustrate our approach by learning a variety of trajectories (tasks (i-iv)) and dynamical systems (tasks (v,vi)). First, we consider a family $\tilde{z}(t;k)$ of target trajectories, parameterized by k. The networks are pretrained on a few of them, where the context target $\tilde{c}$ is a linear function of k. Thereafter the networks dynamically learn to generate a previously unseen trajectory as output and perpetuate it during testing. We start with the simple, instructive target family of oscillations with different periods (task (i)): $\tilde{z}(t;T) = 5\sin(\frac{2\pi}{T}t)$. We use three different teacher trajectories for pretraining, with $T = 10, 15, 20$. After pretraining, our networks can precisely dynamically learn oscillations with unseen periods within and slightly beyond the pretrained ones (Fig. 3.2a,b, see Section 3.A.3 for further detail and analysis of the learning performance of all tasks). Next, in (ii), we generalize (i) to higher order Fourier series. Specifically, we consider the target family of superpositions of two random Fourier series with weighting factor λ: $\tilde{z}(t;\lambda) = (1-\lambda)\tilde{z}_1(t;\lambda) + \lambda\tilde{z}_2(t;\lambda)$. Here, $\tilde{z}_l(t;\lambda)$, $l = 1, 2$, are Fourier series of order O and period $T(\lambda) = (1-\lambda)T_1 + \lambda T_2$. T_l and the Fourier coefficients are drawn randomly. We use seven different teacher trajectories for pretraining, with weighting factors distributed equidistantly between 0 and 1. After pretraining, we test the dynamical learning for thirteen weighting factors also distributed equidistantly between 0 and 1. To quantify the learning performance, we determine the fraction of these targets that can be successfully learned (RMSE below given threshold (0.4) and below RMSE between signal and (other) pretrained targets). We find that networks of increasing size can learn Fourier series with increasing order (Fig. 3.2c,d). Networks with 3000 neurons learn Fourier series of order 10 with a median fraction of successes of close to 90%. Hence, very general periodic functions can be learned. The highest producible frequency is limited by the available neuronal time scales τ_i. We thus expect that larger networks containing smaller τ_i can learn even higher order targets.

To check if our approach also works for a target family with more than one parameter and multidimensional trajectories, we consider in (iii) a superposition of sines with different amplitude and period (consequently $k, \tilde{c}$ are two-dimensional vectors) and in (iv) a set of fixed points along a curve in three-dimensional space. We find that, after pretraining, our networks are able to dynamically learn unseen members of these target families with multidimensional context or signal, as shown in Fig. 3.3a,b for example trajectories.

Second, we consider a family $\dot{\tilde{z}}(t) = F(\tilde{z}(t), u(t); k)$ of target dynamical systems. The networks are pretrained on a few representative systems. Thereafter, an unseen one is dynamically learned. Learning

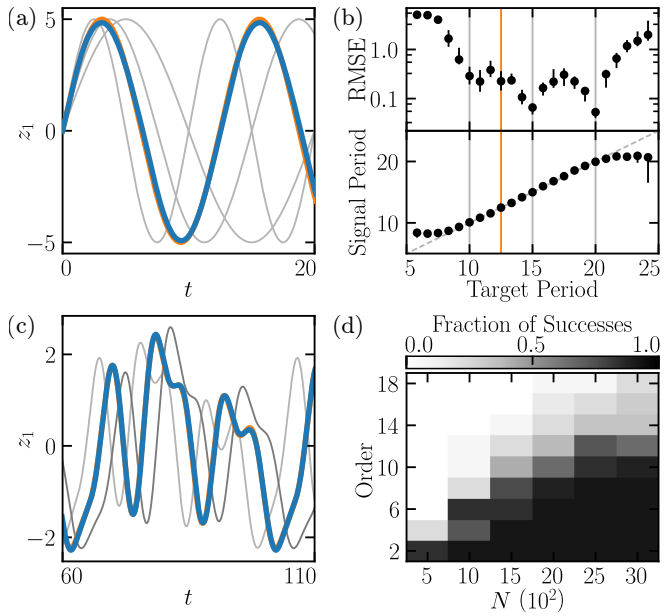

Figure 3.2: Dynamical learning of periodic trajectories.
(a,b) Testing after dynamical learning of sinusoids. (a) Signal (blue) matches the example testing target (orange, mostly covered by signal) well. Pretraining targets (gray traces) are clearly distinct. (b) For many different targets the root-mean-square error (RMSE) between signal and target is low (top) and the signal's period tracks the target's period well (bottom). Gray and orange verticals indicate periods of pretrained targets and target in (a). Dots show median value and errorbars interquartile range, using 10 network instances.
(c,d) Testing after dynamical learning of Fourier series. (c) Like (a), for a random Fourier series with $O = 6$. Only the two closest pretrained dynamics are displayed for clarity. (d) Learning success for different network sizes and orders of the Fourier series. Color encodes median fraction of success, using 40 network instances and random Fourier series.

is in both phases based on imitation of trajectories. However, in contrast to tasks (i-iv) the networks now need to generate unseen output trajectories during testing. To demonstrate dynamical learning of a driven system, we consider task (v) of approximating the trajectory of an overdamped pendulum with drive $u(t)$ and different masses m: $\dot{\tilde{z}}(t) = F(\tilde{z}(t), u(t); m)$. During pretraining and dynamical learning, we use low-pass filtered white noise as drive (Fig. 3.3c, left of dashed vertical). During testing, we use a triangular wave (Fig. 3.3c, right of dashed vertical). As our networks nevertheless generate the correct qualitatively different signal (Fig. 3.3c,d), they must have learned the underlying vector field $F(\tilde{z}, u; m)$. (v) also shows that learning goes beyond interpolation of trajectories (compare blue and gray traces in Fig. 3.3d). Finally, in task (vi) we show dynamical learning of chaotic dynamics, considering autonomous Lorenz systems with different dissipation parameter β of the z-variable. For chaotic dynamics, even trajectories of similar systems quickly diverge. The aim in this task is thus only to generate during testing signals of the same type as the trajectories of the target system. We test this by comparing the limit sets of the dynamics and the tent-map relation between subsequent maxima of the z-coordinate (Fig. 3.3e,f). The reproduction of the tent-map relation further shows that our approach can generate not explicitly trained quantitative dynamical features. We note that the networks also dynamically learn the fixed point convergence of some of the targets in the considered parameter space, even though they were pretrained on chaotic dynamics only (Section 3.A.3).

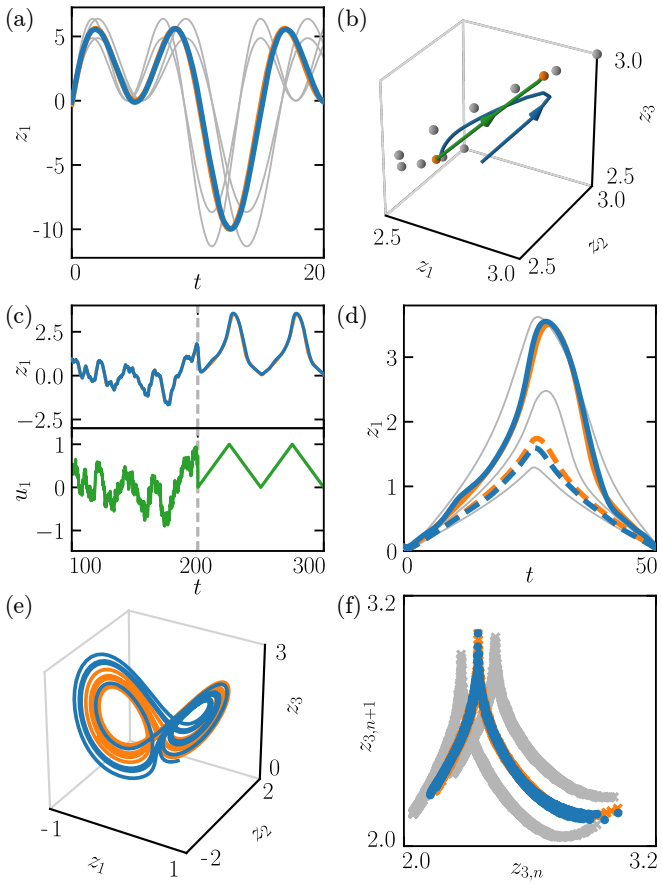

Figure 3.3: Dynamical learning of different tasks.
Testing phase after dynamical learning of an example (a) two-parametric superposition of sines, (b) fixed point, (c,d) driven overdamped pendulum, and (e,f) Lorenz system. (a-d) Signals (blue) match testing targets (orange, mostly covered by signal) well. Pretraining targets (gray traces or spheres) are clearly distinct. (a) displays only the four closest pretrained dynamics for clarity. (b) Signal transients (blue, green) of subsequent dynamical learning of two targets (orange spheres). (c) Signal, target and drive (green) during dynamical learning and subsequent testing (dashed vertical). (d) Dynamically learned approximations of two different pendulums (continuous, dashed), driven by the same triangular input. (e) Limit sets of signal (blue) and target (orange). (f) Tent maps of signal (blue) and dynamical (orange) and pretrained targets (gray).

3.4 Analysis

In the following we analyze the different parts of our network learning and its applicability. One interpretation of the pretraining phase is that the network learns a negative feedback loop, which reduces the error $\varepsilon(t)$. For another interpretation, we split $\varepsilon(t)$ and regroup the z-dependent part of Eq. (3.1) as $(w_z + w_\varepsilon)z(t) - w_\varepsilon\tilde{z}(t)$: feeding back $\varepsilon(t)$ is equivalent to adding a teacher drive $\tilde{z}(t)$, except for a specific change in the feedback weights w_z. For the z-output alone the network thus weight-learns an autoencoder $\tilde{z}(t) \to z(t)$. This is usually an easy task for reservoir networks [137]. To simultaneously learn the constant output $c(t) = \tilde{c}$, the network has to choose an appropriate o_c orthogonal to the subspaces in which the different $z(t)$-driving r-dynamics take place. Orthogonal directions are available in sufficiently large networks, since the subspaces are low-dimensional [138].

After the correct z-dynamics are assumed, we have $\varepsilon(t) \approx 0$. Since remaining fluctuations in $\varepsilon(t)$ could stabilize the dynamics, we usually include ensuing learning phases with $w_\varepsilon \to 0$ and $c(t) = \tilde{c}$. These teach the network to generate the correct dynamics in stable manner under conditions similar to testing. To analyze the principles underlying dynamical learning and testing, we consider task (i). The similarity of the network and learning setups suggests that the same principles underlie all our tasks. We additionally confirm this for (vi) (Section 3.A.4). Viewing the network dynamics in the space of firing rates r, we choose new coordinates with first axis along o_c and the principal components of the dynamics orthogonal to o_c. The dynamics are then given by $c(t) = o_c r(t), r_{\mathrm{PC1}}(t), r_{\mathrm{PC2}}(t), \ldots$ (Fig. 3.4). We focus on the first three coordinates, which describe large parts of the dynamics and output generation.

We find that during dynamical learning, the error feedback drives the dynamics towards an orbit that is shifted in c but similar to pretrained ones. The network therewith generalizes the pretrained reaching and generation of orbits together with corresponding, near-constant $c(t)$, while $\varepsilon(t)$ is fed in. We note that the combination of current state and error input is important (see Fig. 3.4a for $w_\varepsilon \to 0$ and a mismatched $\tilde{z}(t) = \tilde{z}(t_0)$ for $t > t_0$).

During testing, the network generalizes the pretrained characteristics that feeding back $w_c\tilde{c}$ leads to $c(t) \approx \tilde{c}$. Clamping $w_c c(t)$ to $w_c\bar{c}$ thus results in an approximate restriction of $r(t)$ to an $N-1$-dimensional hyperplane with $c(t) = o_c r(t) \approx \bar{c}$ (Fig. 3.4b). The resulting trajectory is for task (i) a stable periodic orbit that generates the desired signal, because the vector field projected to the $c(t) = \bar{c}$-hyperplane is similar to the vector field projected to the $c(t) = \tilde{c}$-hyperplanes embedding nearby pretrained periodic orbits (Fig. 3.4c).

3.5 Discussion

We have introduced a scheme how neural networks can quickly learn dynamics without changing their weights and without requiring a teacher during testing. It relies on a weight-learned mutual association, quasi an entanglement, between contexts and targets. This enables the latter to fix the former during dynamical learning and vice versa during testing.

Previous approaches to supervised dynamical learning with continuous signal space required a form of the teaching signal also during testing. They further differ in network architecture, learning algorithm, task and/or assumption of discrete time from ours (Section 3.A.9 and refs. [33–46, 139–141]). In networks with external input unseen, interpolating input can lead to interpolating dynamics (Section 3.A.7 and ref. [142]). In contrast, our networks *learn* new dynamics, by imitation.

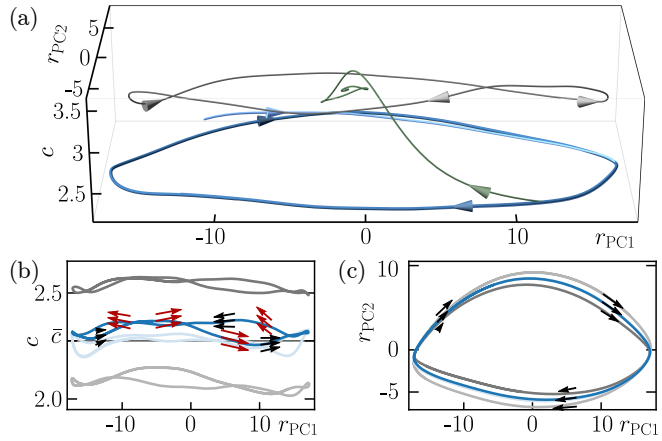

Figure 3.4: Network dynamics during dynamical learning (a) and testing (b,c) of task (i), in c, r_{PC1}, r_{PC2}-coordinates.
(a) During dynamical learning, the error input drives the network to a periodic orbit (light blue trajectory) and keeps it there (blue). Without input, the dynamics converge to a stable orbit (gray) whose signal approximates a pretrained one. Freezing $\tilde{z}(t) = \tilde{z}(t_0)$ drives the dynamics to a fixed point off the orbit (green).
(b) During testing, the assumed orbit (blue) in the c-$r_{PC,1}$-plane is similar to the error driven one (light blue, closest pretrained orbits with $c(t)$ fixed to their $\bar{c}$: gray). The constant feedback $\bar{c}$ prevents the dynamics to leave the region where $c(t) \approx \bar{c}$, compare $\dot{r}(r)$ (black vectors, r on/nearby trajectory) with $\dot{r}(r)$ for variable feedback $c(t)$ (red vectors).
(c) All four orbits are similar in the $r_{PC,1}$-$r_{PC,2}$-plane. The dynamically learned orbit has an attracting projected vector field (black vectors) like the pretrained orbits.

Our scheme is conceptually independent of the network and weight-learning model. The pretraining implements a form of structure learning, i.e. learning of the structure underlying a task family [25, 129]. Animals and humans employ it frequently, but little is known about its neurobiology. We thus realize it by a simple reservoir computing scheme with FORCE learning [125]. We checked that we can use biologically more plausible weight perturbation learning for a simple fixed point learning task (Section 3.A.8).

Dynamical learning is biologically plausible: it is naturally local, causal and does not require fast synaptic weight updates. Continuous supervision could be generated by an inverse model [143] and might be replaceable by a sparse, partial signal. Our dynamical learning is fast (Section 3.A.5) (cf. also [32, 34, 35, 38, 39, 140]). Even for more complicated tasks convergence requires only a few multiples of a characteristic time scale of the dynamics. Further, we find robustness against changes in network and task parameters (Section 3.A.6). The above points suggest a high potential of our scheme for applications in biology, physics and engineering such as neuromorphic computing and the prediction of chaotic systems (Section 3.A.9).

3.A Appendix

3.A.1 Reservoir computing and FORCE learning

Reservoir computing (see also Section 2.2.3) has been introduced several times at different levels of elaborateness and in different flavors, in machine learning and in neuroscience [120, 122–124]. A reservoir computer consists of a high-dimensional, nonlinear dynamical system, the reservoir or liquid, and a comparably simple readout. The reservoir, often a recurrent neural network, "echoes" the input in a complicated, nonlinear way; it acts like a random filter bank with finite memory as each of its units generates a nonlinearly filtered version of the current input and its recent past while forgetting more remote inputs [120, 122, 123, 144]. The simple, often linear readout can then be weight-trained to extract the desired results while the reservoir remains static. Only a fraction of the neural network weights are therefore used for task-related adaptation.

We use a reservoir computing scheme for pretraining. The output weights of our networks, the weights o_z and o_c to z and c, learn online according to the FORCE rule [125], which is well suited for reservoir computers with output feedback [134]. This is because it assumes fast learning of the output weights with a powerful algorithm and thereby ensures that the output and thus the feedback input always match the desired ones up to a small error. The recurrent network is thus largely driven by the correct feedback signals and generates appropriate dynamics already during training. The remaining fluctuations are intrinsically generated and therefore efficiently immunize the system against fluctuations that will occur during testing, leading to dynamically stable generation of desired dynamics. The output weights are trained using the supervised recursive least-squares algorithm. This higher order algorithm provides a least-squares optimal, usually regularized result given the past network states and the targets. Concretely, the version used in ref. [125] and in this chapter starts the recursion with $o_{z,ij}(0)$ and $o_{c,ij}(0)$ for the signal and context output weights and with an $N \times N$ matrix $P(0) = \alpha^{-1} I$, where I is the identity matrix and α^{-1} acts as a learning rate parameter. In learning step n at time t_n the output weights $o_z(n)$ and $o_c(n)$ and the matrix $P(n)$ are recursively updated via

$$o_{z,ij}(n) = o_{z,ij}(n-1) - g_j(n)\varepsilon_i(t_n), \tag{3.2}$$

$$o_{c,ij}(n) = o_{c,ij}(n-1) - g_j(n)e_i(t_n), \tag{3.3}$$

$$P(n) = (I - g(n)r^T(t_n))P(n-1), \tag{3.4}$$

where T denotes transposition, $r(t_n)$ the outputs of the neurons at time t_n and $\varepsilon(t) = z(t) - \tilde{z}(t)$ and $e(t) = c(t) - \tilde{c}(t)$ the errors of the signal and the context. $g(n) = \left(1 + r^T(t_n)P(n-1)r(t_n)\right)^{-1} P(n-1)r(t_n)$ specifies the learning rates of $o_{z,ij}$ and $o_{c,ij}$. They depend on the presynaptic neuron j and on the dynamical history of the entire reservoir, which renders the algorithm causal but non-local. The recursion ensures that $o_z(n)$ and $o_c(n)$ minimize the "ridge regression" error functions

$$E_{z,i}(n) = \sum_{k=1}^{n} \left(\sum_j o_{z,ij}(n) r_j(t_k) - \tilde{z}_i(t_k) \right)^2 + \alpha \sum_{j=1}^{N} \left(o_{z,ij}(n) - o_{z,ij}(0) \right)^2, \tag{3.5}$$

$$E_{c,i}(n) = \sum_{k=1}^{n} \left(\sum_j o_{c,ij}(n) r_j(t_k) - \tilde{c}_i(t_k) \right)^2 + \alpha \sum_{j=1}^{N} \left(o_{c,ij}(n) - o_{c,ij}(0) \right)^2, \tag{3.6}$$

i.e. the individual signal and context errors are kept small with weights that ideally do not deviate far from the initial ones (weight regularization) [135]. The non-locality and the assumed fast weight changes are considered biologically implausible [60, 125].

3.A.2 Additional detail on the applications

In the following, we detail the parameters, setups and targets used in the different applications. We denote the duration of the pretraining phase by t_{wlearn}. Each training period (individual target presentation) in it lasts for t_{stay}. If not mentioned otherwise, in the beginning of each period until t_{fb} the network receives error input $\varepsilon(t)$ and the context signal evolves freely. Thereafter, $w_\varepsilon \to 0$ and $c(t)$ is fixed to its target value. The intervals between updates of the output weights have length dt for task (ii) and random lengths with an average of 0.5 for the other tasks [145]. The parameter of the FORCE rule is $\alpha = 1$. Dynamical learning lasts for t_{learn}. During dynamical learning, we determine $\bar{c}$ by averaging the context signal with an exponentially forgetting kernel ($\tau_{\text{forget}} = 50$ for task (v) and $\tau_{\text{forget}} = 5$ for the other tasks). Testing lasts for t_{test}.

In all applications, recurrent weights A_{ij} are set to zero with probability $1 - p$. Nonzero weights are drawn from a Gaussian distribution with mean 0 and variance $\frac{g^2}{pN}$, where $g = 1.5$ [125]. We draw the feedback weights $w_{z,ij}$, $w_{c,ij}$ and the input weights $w_{\varepsilon,ij}$, $w_{u,ij}$ from a uniform distribution between $-\tilde{w}$ and $\tilde{w}$, set all initial output weights $o_{z,ij}(0)$ and $o_{c,ij}(0)$ to 0 and draw the biases b_i from a uniform distribution between -0.2 and 0.2. The number of external inputs is N_u. We use the standard Euler method for our simulations, with an integration time step of $dt = 0.1$, except for Figs. 3.4 and 3.11, where we use $dt = 0.01$ and $dt = 0.025$, respectively. We implement the model using Python and NumPy [146] (see [147] for example code for task (i)).

Further settings in the individual tasks are as follows:

Task (i): $N = 500, N_z = 1, N_c = 1, N_u = 0, p = 0.1, \tilde{w} = 1, t_{\text{stay}} = 500, t_{\text{fb}} = 100, t_{\text{wlearn}} = 50000, t_{\text{learn}} = 50, t_{\text{test}} = 5000$. The network learns to generate sinusoidal oscillations with period T. The family of target trajectories is $\tilde{z}(t;T) = 5\sin(\frac{2\pi}{T}t)$. We use three different teacher trajectories for pretraining, with periods $T = 10, 15, 20$ and corresponding context targets $\tilde{c} = 2, 2.5, 3$. The target of dynamical learning in Fig. 3.2a and Fig. 3.12 has $T = 12.5$.

Task (ii): $N = 500$–$3000, N_z = 1, N_c = 1, N_u = 0, p = 0.1, \tilde{w} = 1, t_{\text{stay}} = 500, t_{\text{fb}} = 100, t_{\text{wlearn}} = 50000, t_{\text{learn}} = 100, t_{\text{test}} = 500$. We do not update the output weights during a time interval of 20 at the beginning of each training period. The network learns to generate a superposition of two Fourier series with weighting factor λ. The family of target trajectories is $\tilde{z}(t;\lambda) = (1-\lambda)\tilde{z}_1(t;\lambda) + \lambda\tilde{z}_2(t;\lambda)$ with $\tilde{z}_l(t) = \frac{1}{C_l}(\frac{\tilde{a}_{l,0}}{2} + \sum_{o=1}^{O}\tilde{a}_{l,o}\sin(\frac{2\pi o}{T(\lambda)}t + \tilde{\varphi}_{l,o}))$, $l = 1, 2$, and $T(\lambda) = (1-\lambda)T_1 + \lambda T_2$. We draw the $\tilde{a}_{l,0}$, $\tilde{a}_{l,o}$, $\tilde{\varphi}_{l,o}$ and T_l from uniform distributions between -10 and 10, 0 and 10, 0 and 2π, and 20 and 50, respectively. C_l is drawn from a uniform distribution to normalize the maximal value of $|\tilde{z}_l(t)|$ to a random value between 3 and 7. We use seven different teacher trajectories for pretraining, with weighting factors λ distributed equidistantly between 0 and 1. The corresponding context targets are distributed equidistantly between 2 and 3. The target of dynamical learning in Fig. 3.2c has $N = 2000, O = 6, \lambda = \frac{7}{12}$.

Task (iii): $N = 1000, N_z = 1, N_c = 2, N_u = 0, p = 0.2, \tilde{w} = 1, t_{\text{stay}} = 500, t_{\text{fb}} = 100, t_{\text{wlearn}} = 50000, t_{\text{test}} = 1000$. The network learns to generate a superposition of sinusoidal oscillations with amplitude a and period T. The family of target trajectories is $\tilde{z}(t;a,T) = a\left(\sin(\frac{2\pi}{T}t) + \cos(\frac{4\pi}{T}t)\right)$. We use sixteen different teacher trajectories for pretraining, with four amplitudes a distributed equidistantly

between 3 and 7 and four periods T distributed equidistantly between 10 and 20. The corresponding context targets are distributed equidistantly between 2 and 3 for both parameters. The target of dynamical learning in Fig. 3.3a and Fig. 3.12 has $a = 5$ and $T = 15$.

Task (iv): $N = 500, N_z = 3, N_c = 1, N_u = 0, p = 0.1, \tilde{w} = 1, t_{\text{stay}} = 200, t_{\text{fb}} = 100, t_{\text{wlearn}} = 50000, t_{\text{learn}} = 50, t_{\text{test}} = 1000$. The network learns to generate a constant output positioned on a curve in three-dimensional space parameterized by s. The family of target trajectories (fixed points) is $\tilde{z}(t; s) = \left(\frac{s^3}{2} + s_{\text{off}}, 2(s - \frac{1}{2})^2 + s_{\text{off}}, \frac{s}{2} + s_{\text{off}}\right)$, where the offset $s_{\text{off}} = 2.5$ ensures that the network feedback is strong enough to entrain the reservoir network. We use ten different teacher trajectories for pretraining with parameters s chosen between 0 and 1 such that the corresponding $\tilde{z}(t; s)$ lie equidistantly on the target curve $\{\tilde{z}(t; s) | s \in [0, 1]\}$. The corresponding context targets are distributed equidistantly between 2 and 3. The targets of dynamical learning in Fig. 3.3b have $s = 0.10$ and $s = 0.92$.

Task (v): $N = 1000, N_z = 1, N_c = 1, N_u = 1, p = 0.2, \tilde{w} = 2, t_{\text{stay}} = 1000, t_{\text{wlearn}} = 30000, t_{\text{learn}} = 200, t_{\text{test}} = 500$. We choose τ_i from a uniform distribution between 0.3 and 2.5. During pretraining, we always provide error input $\varepsilon(t)$ to the network and do not fix $c(t)$, i.e. $t_{\text{fb}} = t_{\text{stay}} = 1000$. The network learns to predict the angle of a driven overdamped pendulum with mass m. The family of target dynamical systems is given by $\dot{\tilde{z}}(t) = F(\tilde{z}(t), u(t); m) = -m \sin(\tilde{z}(t)) + u(t) - \exp((\tilde{z}(t) - 0.65\pi)/0.65\pi) + \exp(-(\tilde{z}(t) + 0.65\pi)/0.65\pi)$. The last two terms provide a soft barrier preventing the pendulum from undergoing full rotations. During pretraining and dynamical learning, the pendulum is driven by low-pass filtered white noise $\dot{u}_{\text{wlearn}}(t) = -u_{\text{wlearn}}(t) + 0.2dW/dt$ (see Fig. 3.9b), which allows a comprehensive sampling of the pendulum's dynamics. During testing the pendulum is driven by a triangular wave with unit amplitude and period T= 50. We use three different teacher dynamical systems for pretraining, with $m = 0.5, 1.0, 1.5$ and corresponding context targets $\tilde{c} = 0.7, 0.95, 1.2$. The targets of dynamical learning in Fig. 3.3c,d have $m = 0.8$ (continuous trace) and $m = 1.2$ (dashed trace).

Task (vi): $N = 1000, N_z = 3, N_c = 1, N_u = 0, p = 0.1, \tilde{w} = 2, t_{\text{stay}} = 1000, t_{\text{fb}} = 100, t_{\text{wlearn}} = 50000, t_{\text{learn}} = 50, t_{\text{test}} = 10000$. The network learns a Lorenz system with dissipation parameter β. During pretraining, we always provide error input $\varepsilon(t)$ to the network, but fix $c(t)$ after t_{fb}. The family of target dynamical systems is given by $\dot{\tilde{z}}(t) = F(\tilde{z}(t); \beta) = F_{\text{Lorenz}}(C_{\text{Lorenz}}\tilde{z}(t); \beta)/(C_{\text{Lorenz}}\tau_{\text{Lorenz}})$, where $C_{\text{Lorenz}} = 40$ and $\tau_{\text{Lorenz}} = 20$ determine the spatial and temporal scale of the dynamics and $F_{\text{Lorenz}}(x(t); \beta) = (\sigma(x_2 - x_1), x_1(\rho - x_3) - x_2, x_1x_2 - \beta x_3)$ is the vector field of the standard Lorenz system, with $\sigma = 10$ and $\rho = 70$. We use four teacher dynamical systems for pretraining, with parameters β distributed equidistantly between 2 and 6 and corresponding context targets distributed equidistantly between 2 and 3. The target of dynamical learning in Fig. 3.3e,f and Fig. 3.12 has $\beta = 4$.

3.A.3 Quantification of learning performance

To quantify the performance of our model, we measure for each application the errors between signal outputs and targets during testing, for different network instances and targets. Except for task (vi), we compute the testing error as the root-mean-square error between signal output and target during a period of length 50 in the middle of the testing phase. The measure is chosen to ignore phase shifts that occur over long testing times, as they are unavoidable in periodic autonomous dynamics (tasks (i,ii)), due to the accumulation of small errors in the period.

Task (i)

Fig. 3.5a shows the testing error for the learning of sinusoidal oscillations. It is small for targets with periods within and slightly beyond the range spanned and interspersed by pretrained targets. Fig. 3.5b shows the good agreement between the periods of the output signals and the targets. We determine the periods from the maxima of the output signals' power spectra, after discarding the initial interval of length 100 of the testing phase to allow for equilibration.

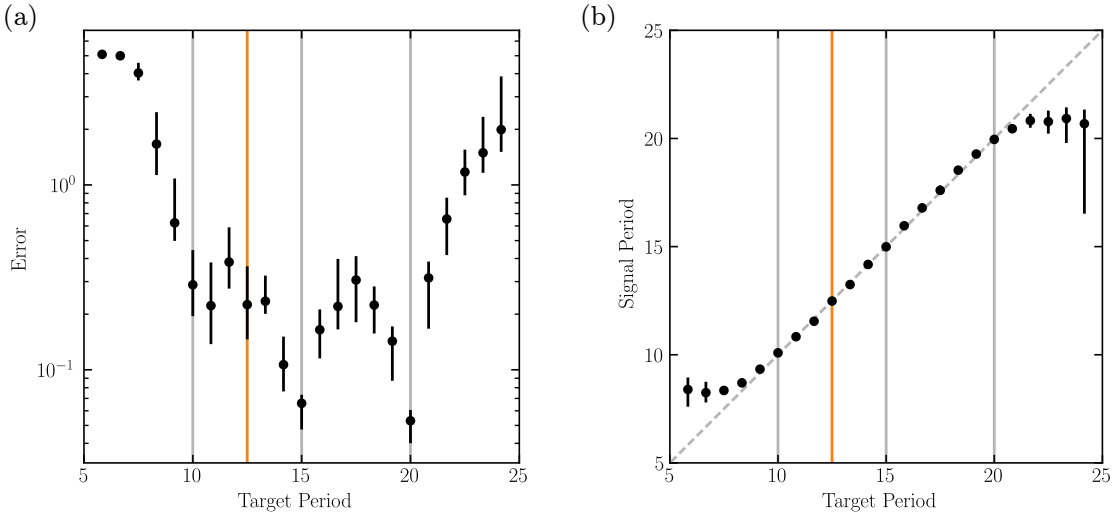

Figure 3.5: Quality of dynamical learning of the sinusoidal oscillations in task (i).
(a) Testing error between signal output and target and (b) period of the signal output, as a function of the period of the target. Vertical gray lines indicate the periods of the pretrained targets and vertical orange lines indicate the period of the target used in Fig. 3.2a. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances.

Task (ii)

Fig. 3.6 shows the testing error for four different combinations of network size N and order O together with the signal and target in time and frequency domain for example instances of task (ii), i.e., for specific realizations of the random Fourier series described above. The testing error is low within the range of pretrained weighting factors, especially for the targets used during pretraining.

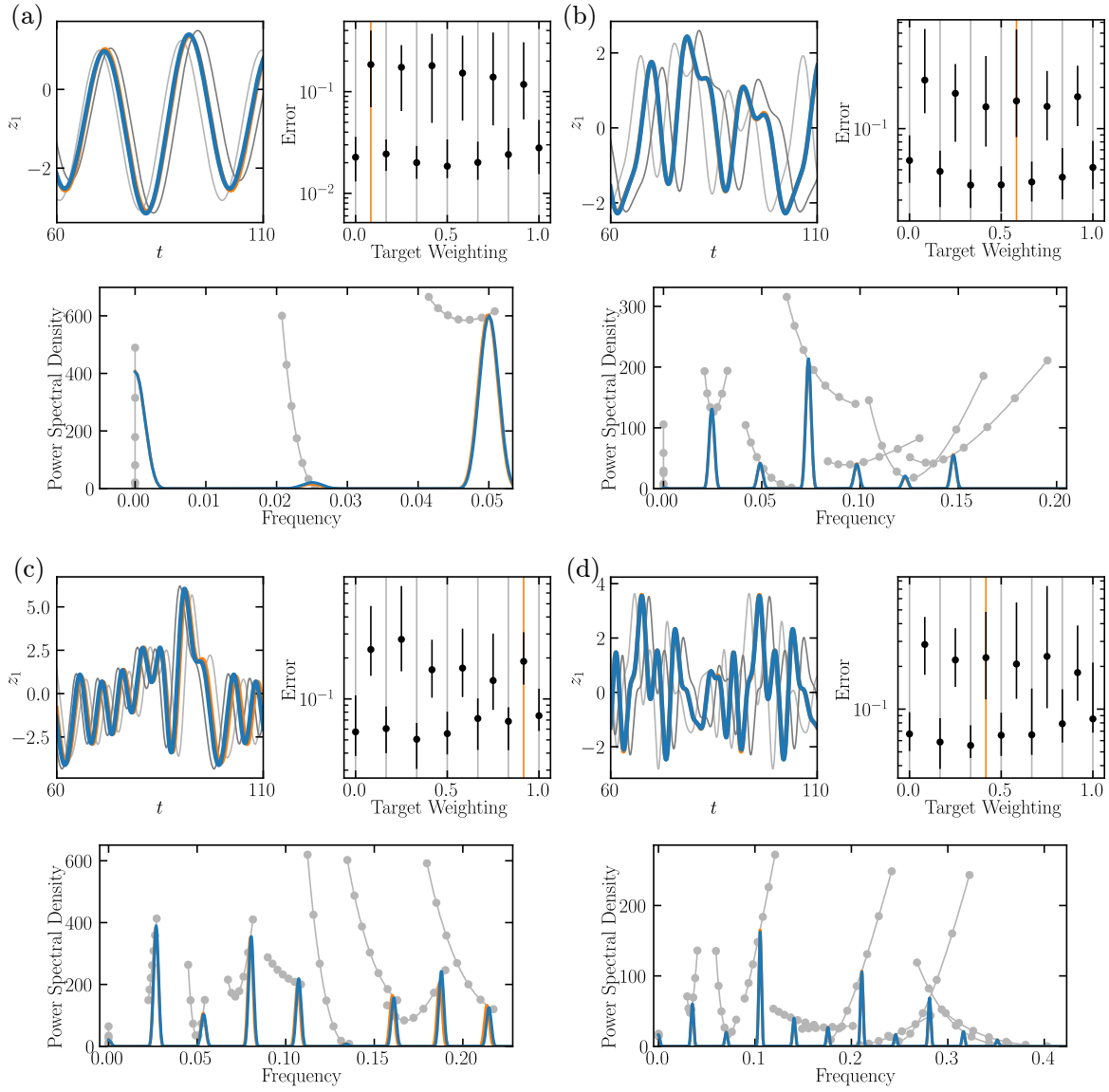

Figure 3.6: Quality of dynamical learning of the superposition of Fourier series in task (ii).
(a) $N = 1000, O = 2$ (a, top left) Signal (blue) and target (orange) together with the two closest pretrained dynamics (gray) during testing after dynamical learning of an unseen target. (a, bottom) Power spectral density of signal (blue) and target (orange) during testing. Gray lines show the power of the individual Fourier components of the target family within the range of pretrained targets. Gray dots indicate the targets used during pretraining. (a, top right) Testing error between signal output and target as a function of the target weighting factor. Vertical gray lines indicate the weighting factors of the pretrained targets and vertical orange lines indicate the weighting factors of the targets used in the other subpanels. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 40 network instances and random Fourier series.
(b-d) Same as (a) but with (b) $N = 2000, O = 6$, (c) $N = 2500, O = 8$, (d) $N = 3000, O = 10$.

Task (iii)

Fig. 3.7a shows the testing error for the learning of superpositions of sines. Again, the error is low within and slightly beyond the range of the parameters of the pretrained targets. Similarly, the averaged local maxima of the signal outputs agree well with the averaged local maxima of their targets, Fig. 3.7b. The measurement of maxima starts at time 100 after the beginning of testing.

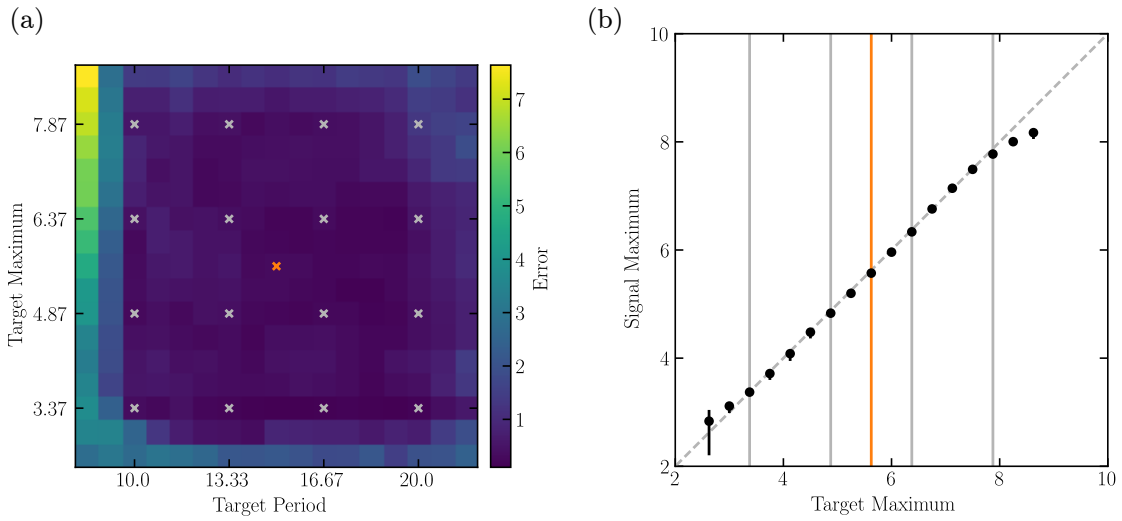

Figure 3.7: Quality of dynamical learning of the superpositions of sines in task (iii).
(a) Median testing error between signal output and target as a function of the maximum and the period of the target function. Gray crosses indicate parameters of the pretrained targets and the orange cross indicates the parameters used in Fig. 3.3a.
(b) Averaged local maxima of the signal output as a function of the averaged local maxima of the target, for a target period of $T = 15$. Vertical gray lines indicate the maxima of the pretrained targets and the vertical orange line indicates the maximum of the target used in Fig. 3.3a. Dots show median value and errorbars represent the interquartile range between first and third quartile. Results in (a) and (b) are obtained using 10 network instances for each parameter pair.

Task (iv)

Fig. 3.8a shows the testing error for the learning of fixed points. It is low for target positions within and slightly beyond the range of the positions of the pretrained targets. Fig. 3.8b shows signal outputs for different targets dynamically learned by a single network instance.

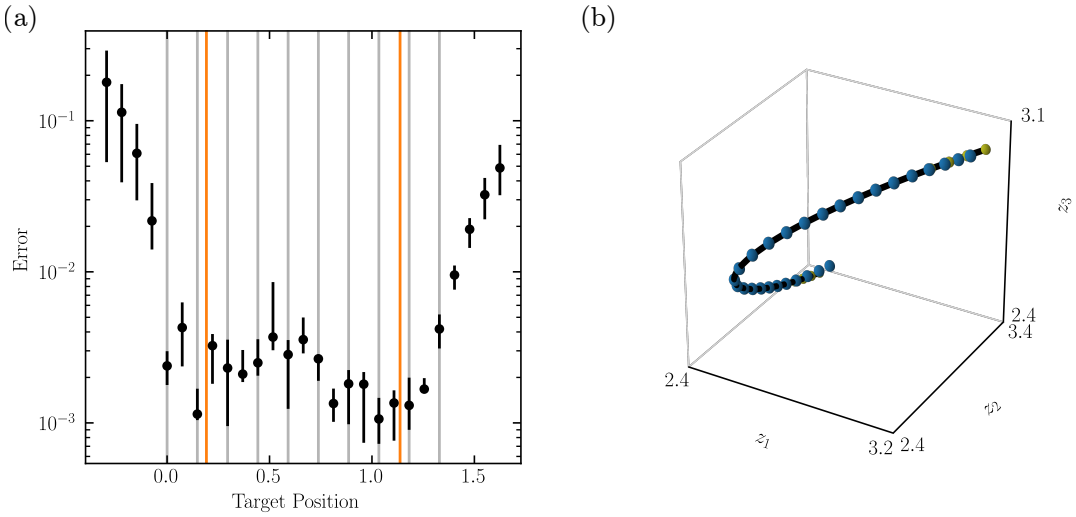

Figure 3.8: Quality of dynamical learning of the fixed points in task (iv).
(a) Testing error between signal output and target as a function of the target position. Vertical gray lines indicate the positions of the pretrained targets and vertical orange lines indicate the positions of the targets used in Fig. 3.3b. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances.
(b) Single network instance learning the same set of dynamical learning targets as in (a). Blue spheres indicate the last signal outputs during testing after the different instances of dynamical learning. Yellow spheres indicate the position of the corresponding targets. They are mostly covered by blue spheres, except in the regions of larger error. The black tube shows the curve $\tilde{z}(t;s)$ on which the targets lie.

Task (v)

Fig. 3.9a shows the testing error for the learning of driven overdamped pendulums. It is small for pendulums with masses within and slightly beyond the range spanned and interspersed by pretrained pendulums. Fig. 3.9b illustrates the dynamical learning and testing phases.

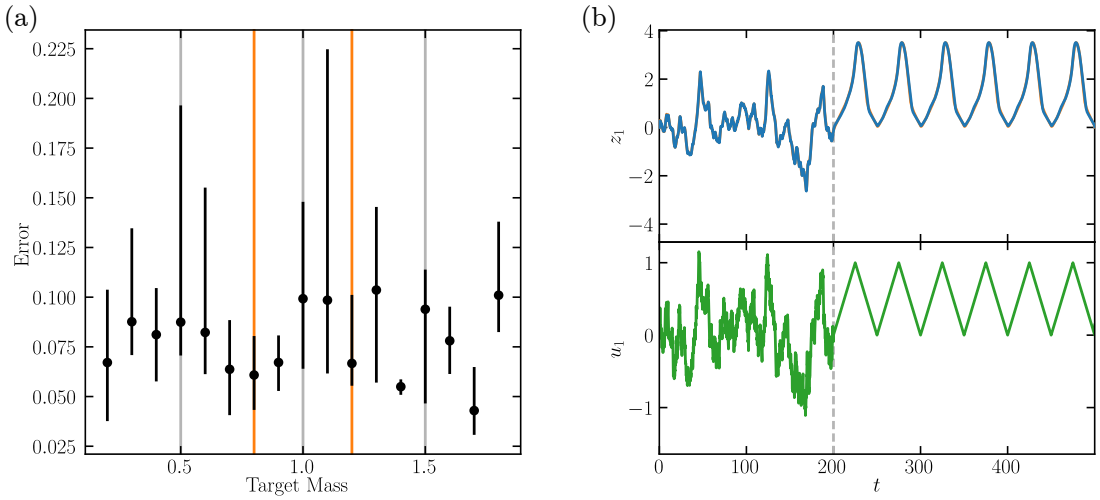

Figure 3.9: Quality of dynamical learning of the overdamped pendulums in task (v).
(a) Error between signal output and target, as a function of the target pendulum's mass. Vertical gray lines indicate the masses of the pretrained targets and vertical orange lines indicate the masses of the targets used in Fig. 3.3d,e. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances.
(b) Dynamical learning and testing. The network and the target receive the same low-pass filtered white noise as input drive during dynamical learning and triangular wave input during testing (lower subpanel). The network response (upper subpanel, blue trace) agrees well with the response of the target (upper subpanel, orange trace, nearly completely covered by the blue trace).

Task (vi)

Since the Lorenz system is chaotic for most of the parameter range that we consider, the signal output trajectory quickly deviates from the target system's trajectory during testing. This holds also if the network approximates the target dynamical system well. Hence, instead of using the root-mean-square error, we compute the testing error as the discrepancy of the limit set M_{net} generated by the network and the limit set M_{tar} generated by the target dynamics. For the comparison, we use the Averaged Hausdorff Distance [148],

$$d_{\text{AHD}}(M_{\text{net}}, M_{\text{tar}}) = \max\left[\frac{1}{\mid M_{\text{net}} \mid}\sum_{m_{\text{net}}\in M_{\text{net}}} d(m_{net}, M_{\text{tar}}), \frac{1}{\mid M_{\text{tar}} \mid}\sum_{m_{\text{tar}}\in M_{\text{tar}}} d(m_{tar}, M_{\text{net}})\right], \quad (3.7)$$
$$d(m, M) = \min_{m'\in M} \parallel m - m' \parallel,$$

which is robust against outliers. Fig. 3.10a shows that the testing error is low within the range of parameters β spanned and interspersed by pretrained targets. In addition, we find that the relation between subsequent maxima of the z-coordinate of the signal output correctly forms the shape of a tent for most tested parameters (Fig. 3.10b). The behavior of our model also reproduces a bifurcation occurring for large β: The target Lorenz system changes from chaotic behavior to fixed point behavior for the largest value of β we consider. Our networks dynamically learn to generate the fixed point dynamics from this target, although they were only pretrained in the chaotic regime. We note that some network instances, for example the one shown in Fig. 3.10b, generate fixed point behavior during testing, if the target has the second largest value of β and is thus still chaotic. However, also in these cases the signal output converges to one of the two fixed points appearing for the largest β. This suggests that due to a shift in the averaged context parameter, the dynamical regime beyond the bifurcation is generated during testing.

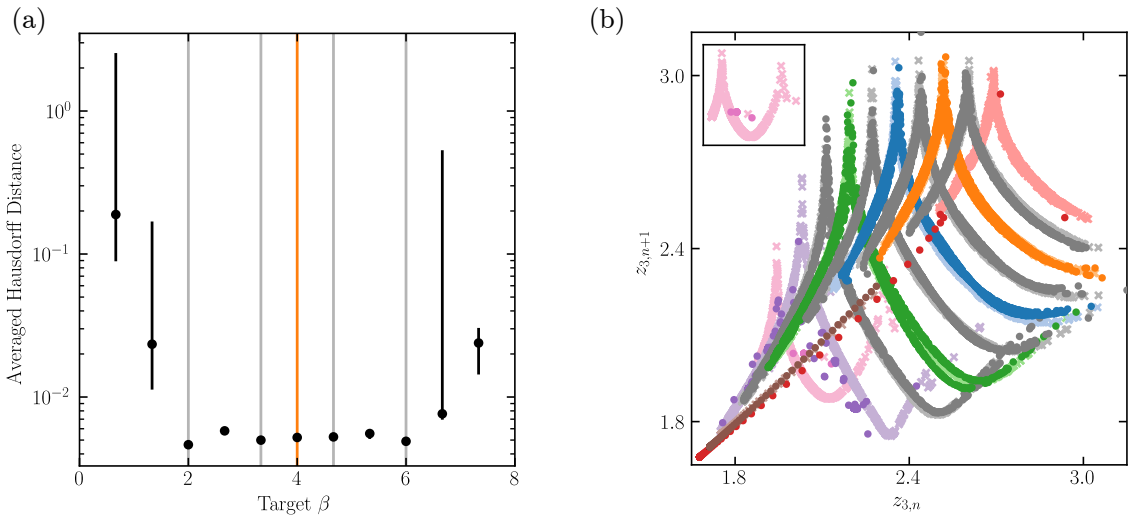

Figure 3.10: Quality of dynamical learning of the Lorenz systems in task (vi).
(a) Testing error comparing the limit sets of signal output and target, as a function of the target's parameter β. Vertical gray lines indicate the parameters of the pretrained targets and the vertical orange line indicates the parameter of the target used in Fig. 3.3e,f. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances.
(b) Tent maps of subsequent maxima in the z-coordinate for the signal output (dots, colored differently for different targets) and for the target dynamics (crosses, light coloring alike corresponding dots). The parameters β of the targets are the same as in (a). Dynamical learning of all targets with a single network instance. Blue data correspond to the signal and target used in Fig. 3.3e,f; gray data indicate pretrained targets. Tent maps of the target dynamics move from bottom left to top right for increasing β except for the largest β (brown, bottom left), for which the target dynamics converge to a fixed point. Inset show close-up of results for the smallest considered value of β. The signal output goes to a fixed point for the two largest, but also for the smallest considered value of β, leading to a focusing of the maxima relation to a small region.

3.A.4 Analysis of dynamical learning of chaotic dynamics

To show that the mechanisms underlying dynamical learning and testing that we worked out using task (i) also hold for a qualitatively different, chaotic system, Fig. 3.11 analyzes them for task (vi). As expected, we observe that during dynamical learning, the error feedback drives the dynamics towards an orbit generalizing the pretrained ones and keeps it there. During testing, the network generalizes the pretrained characteristics to autoencode c such that the dynamics stay near the $\bar{c}$-plane in r-space when the feedback $w_c c$ is clamped to $w_c \bar{c}$. The trajectory is for task (vi) usually chaotic and generates the desired output signal, because the vector field projected to the $c(t) = \bar{c}$-hyperplane inter- or extrapolates nearby vector fields of other c-hyperplanes, which embed pretrained orbits generating Lorenz dynamics with neighboring parameters.

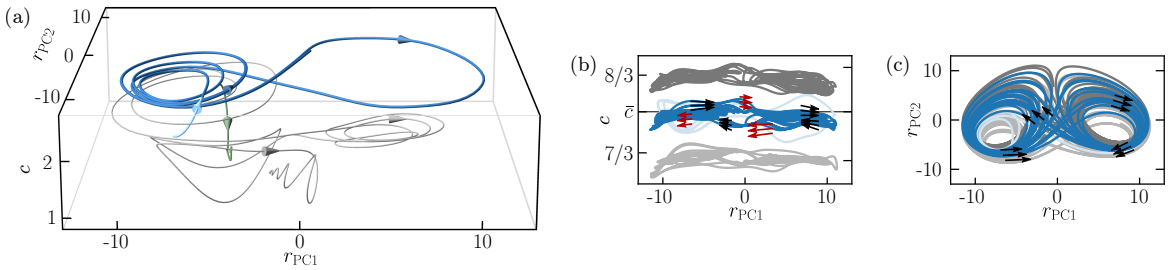

Figure 3.11: Recurrent network dynamics during dynamical learning (a) and testing (b,c) of task (vi), in c, r_{PC1}, r_{PC2}-coordinates (see Fig. 3.4 for task (i)).
(a) During dynamical learning, the error input drives the network to an orbit whose signal output approximates the desired Lorenz system and keeps it there (light blue and blue trajectories). Without input, the dynamics converge to a stable fixed point, after a transient that yields a Lorenz system-like signal (gray). Freezing $\tilde{z}(t) = \tilde{z}(t_0)$ drives the dynamics quickly to a fixed point (green).
(b) During testing, the assumed orbit (blue) resembles the error driven one in the c-$r_{PC,1}$-plane (light blue, closest pretrained orbits with $c(t)$ fixed to their $\tilde{c}$: gray). The constant feedback $\bar{c}$ prevents the dynamics to leave the region where $c(t) \approx \bar{c}$, compare $\dot{r}(r)$ (black vectors, r on/nearby trajectory) with $\dot{r}(r)$ for variable feedback $c(t)$ (red vectors).
(c) All four orbits are similar in the $r_{PC,1}$-$r_{PC,2}$-plane, since the dynamically learned orbit has a similar projected vector field (black vectors) as the nearby pretrained ones.

3.A.5 Learning speed of dynamical learning

In the following we quantitatively assess the speed of dynamical learning. We compare it with that of standard FORCE weight-learning, which uses reservoirs with only a signal output $z(t)$ and output weight-learning. As example tasks we consider learning of the sinusoidal oscillation, task (i), of the superposition of sines, task (iii), and of the Lorenz system, task (vi). The reservoirs for standard FORCE learning have our standard parameters, except that the biases are drawn from a uniform distribution between -5 and 5. Further, the output weight-learning parameters are adapted; we apply weight updates on every integration time step and set α to 0.001. Both changes improve performance and are for some combinations of configuration and task even necessary for convergence. We consider three different configurations of standard FORCE learning (Fig. 3.12a): First, the typical configuration of a reservoir without input and initialization of o_z to 0. In the second configuration o_z is initialized instead to the signal output weights obtained at the end of pretraining for dynamical learning. This accounts for the possibility that these output weights are beneficial initial conditions for weight-learning and that our structural learning facilitates subsequent FORCE learning despite the lack of context input, which was present during pretraining. In the third configuration o_z is initialized to 0 and the reservoir receives an error input $\varepsilon(t) = z(t) - \tilde{z}(t)$ during learning, because this might also facilitate FORCE learning. To evaluate performance after different learning durations, we compute testing errors as described in Section 3.A.3. As usual, we stop weight modifications and, if present, error input during testing. For a fair comparison, for dynamical learning with $t_{\text{learn}} = 0$ we fix the context to 0.

We find that dynamical learning is similarly fast or faster than FORCE (Fig. 3.12b). For tasks (i) and (iii), both dynamical learning and FORCE learning converge within approximately two periods of the target dynamics ($T = 12.5$ and $T = 15$). FORCE learning converges to smaller errors. For task (vi), dynamical learning converges in about five cycles (maxima of the z-coordinate) of the target system. FORCE learning is about five times slower and yields similar errors. The similar convergence speed of the first two configurations in all considered tasks indicates that FORCE weight-learning does not profit from our form of pretraining.

Taken together, we observe that dynamical learning converges within a few characteristic timescales of the target dynamics and is thus on par with FORCE learning for simple and faster converging for complex tasks. This held for both the standard and the hand-tuned parameter sets. The observation is plausible since for complicated tasks FORCE learning needs to gather information that dynamical learning already possesses due to the previous pretraining. It is especially interesting because dynamical learning may be considered biologically plausible and because FORCE is a recommended reservoir computing scheme [134].

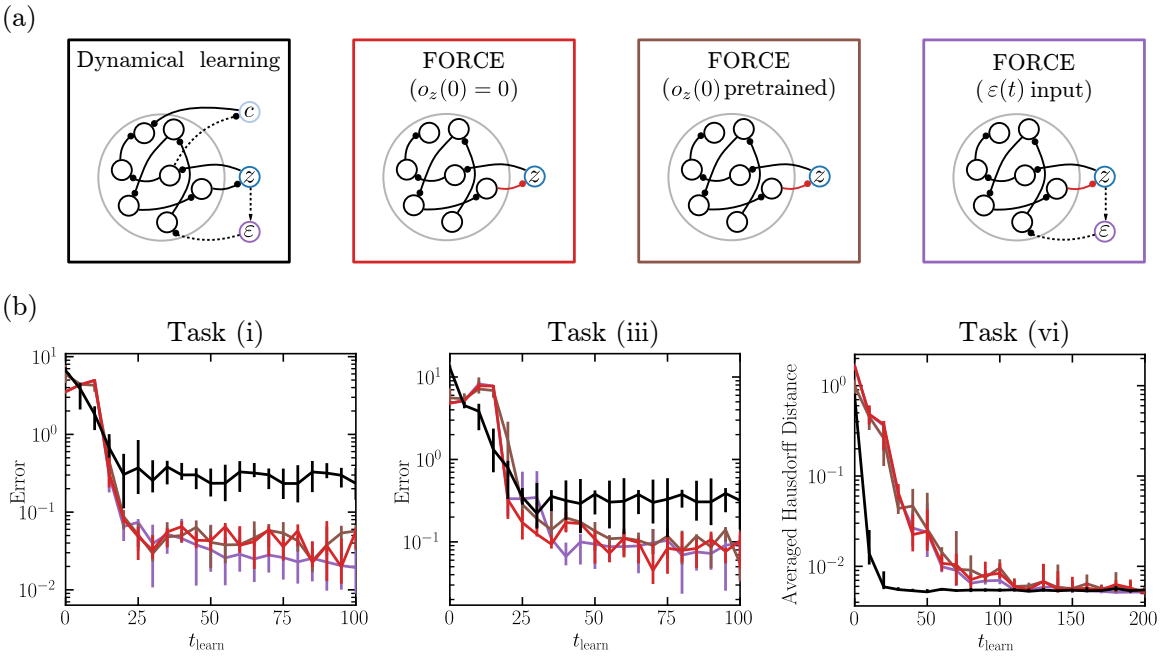

Figure 3.12: Learning speed of dynamical learning and FORCE weight-learning.
(a) Schematics of the different learning schemes. Style of drawing has same meaning as in Fig. 3.1.
(b) Testing error as a function of learning time for dynamical learning (black), FORCE learning with o_z initialized to zero (red), FORCE learning with o_z initialized to the signal output weights after pretraining (brown) and FORCE learning with error input and o_z initialized to 0 (purple, colors are alike frame colors in (a)). Connected points represent median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances.

3.A.6 Robustness of learning performance

To check the robustness of our dynamical learning scheme against changes in task family parameters, we determine its performance for different families of sinusoidal oscillations, task (i). Specifically, we vary the number of pretrained targets, the amplitude of the oscillations, the difference between the maximal and minimal period of the pretrained targets (period range) as well as the minimal period of the weight-learned targets. For each combination of these task family parameters, we pretrain the networks as before. Afterwards, we dynamically learn a set of targets with periods ranging from the smallest to the largest pretrained period, where the period increases by one between neighboring targets. We compute a normalized error for each target and take the average to quantify the performance of the network for the considered task family. The normalized error is the root-mean-square error during a period in the middle of the testing phase, with length three times the target period, divided by the corresponding root-mean-square error assuming that the signal output is zero.

To compute and interpret the errors in high-dimensional parameter space, we cut out slices where we keep all but at most two of the task family parameters at their standard values specified in Section 3.A.2. We find that dynamical learning works robustly for large parameter regions. In particular, the number of targets and the period range can often be changed over an order of magnitude, see Fig. 3.13. Increasing the network size to 1000 neurons and the number of pretrained targets to five instead of three further increases robustness against changing other parameters, see Fig. 3.14. Taken together, we may conclude that our scheme works well for a wide range of task families.

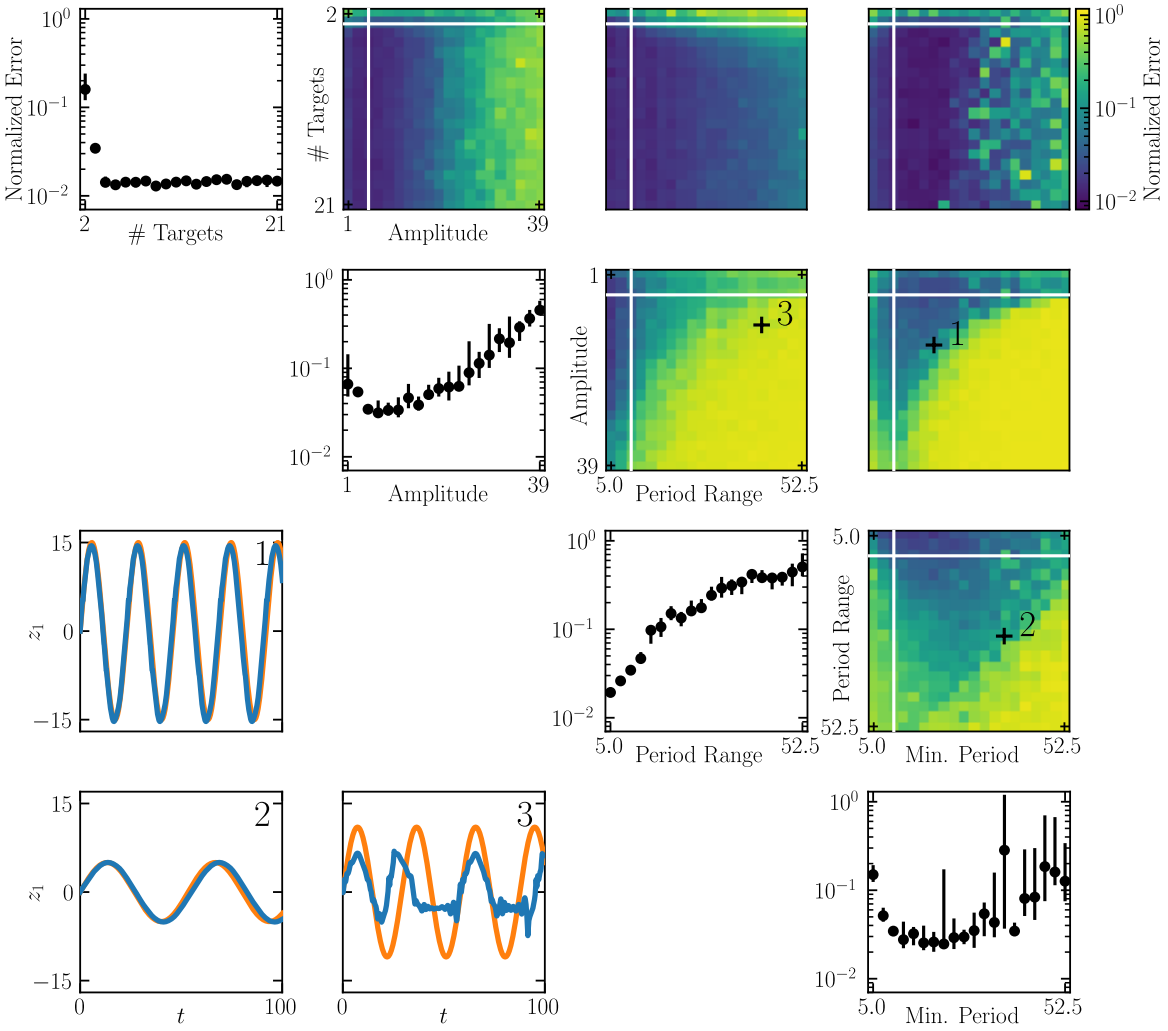

Figure 3.13: Performance over a broad range of task family parameters.
Panels on and above the diagonal show the average normalized errors taken over sets of testing targets. All but the indicated parameters are set to their standard values. White lines in panels above the diagonal indicate the parameter values of the one-dimensional slices shown on the diagonal. Dots and color represent median value and errorbars in panels on the diagonal represent the interquartile range between first and third quartile, using 10 network instances. Panels below the diagonal show representative dynamically learned example signal outputs (blue) and corresponding targets (orange) for the three different parameter combinations indicated by numbered crosses in the panels above the diagonal.

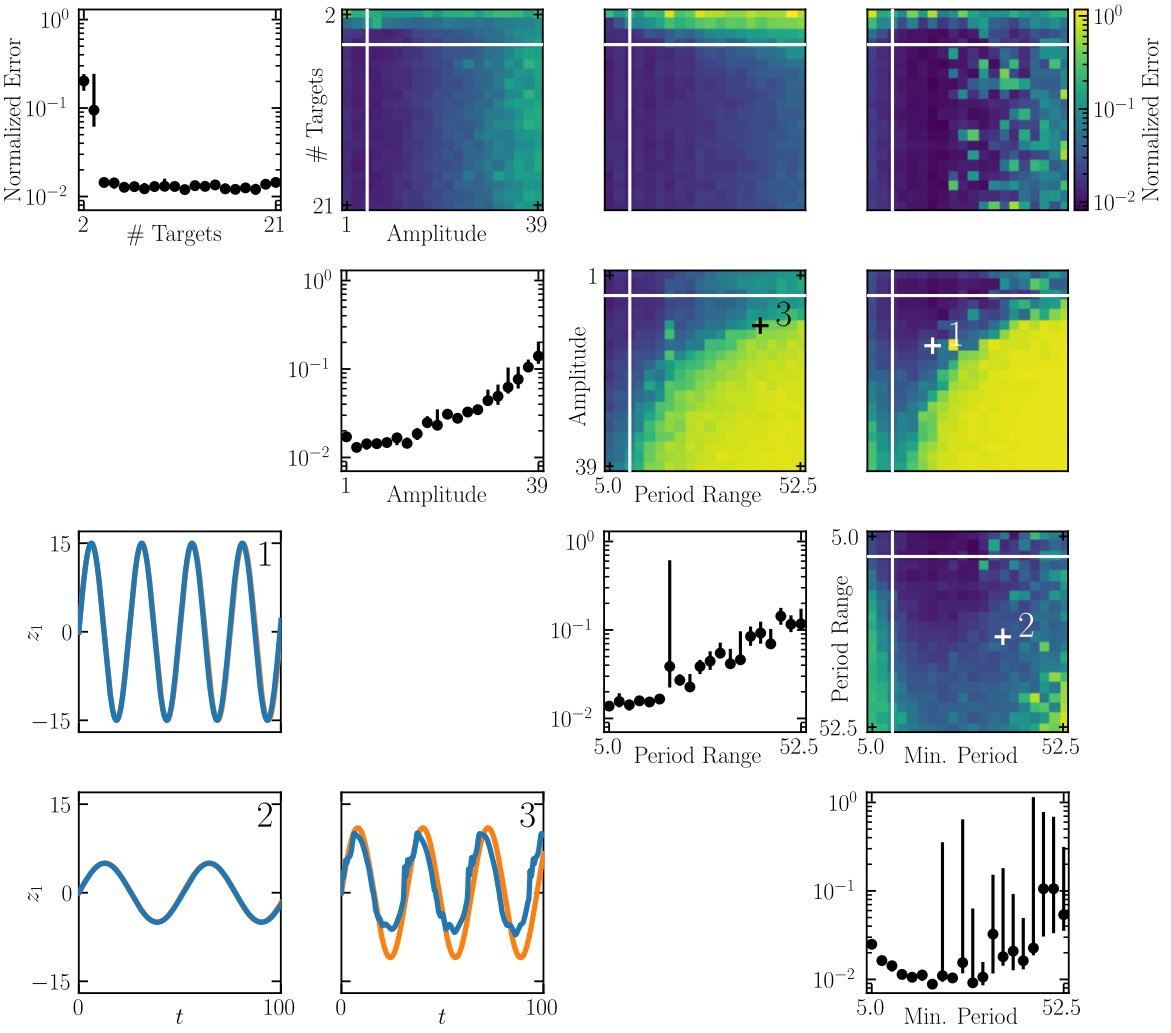

Figure 3.14: Same as Fig. 3.13 for networks with 1000 neurons and five pretrained targets unless the number of pretrained targets is varied.

3.A.7 Induction of unseen signal outputs by a context-like external input

We test whether changing a context-like input $u_c(t)$ allows to generate sinusoidal oscillations with previously unseen frequencies. Like $c(t)$, $u_c(t)$ connects to the neurons in the network with a weight matrix w_c. However, $u_c(t)$ is never generated by a network output, but a purely external input. There is no further context variable $c(t)$ and no error input $\varepsilon(t)$ in the network. Apart from this, the network is setup like in task (i). The output weights w_z are learned using the FORCE rule, similar to pretraining in task (i): during each training period, we teach the network to generate a sinusoidal oscillation $\tilde{z}(t;T)$ with a period $T = 10, 15, 20$, in response to a constant $u_c(t) = 2, 2.5, 3$, analogous to teacher forcing with $\tilde{c}$. We find that the system can interpolate between the pretrained output signals, if driven by previously unseen $u_c(t)$, cf. Fig. 3.15. See ref. [149] for a similar finding when morphing between conceptor weight matrices. (The recent 'conceptor' approach fixes reservoir dynamics by weight changes [149, 150].)

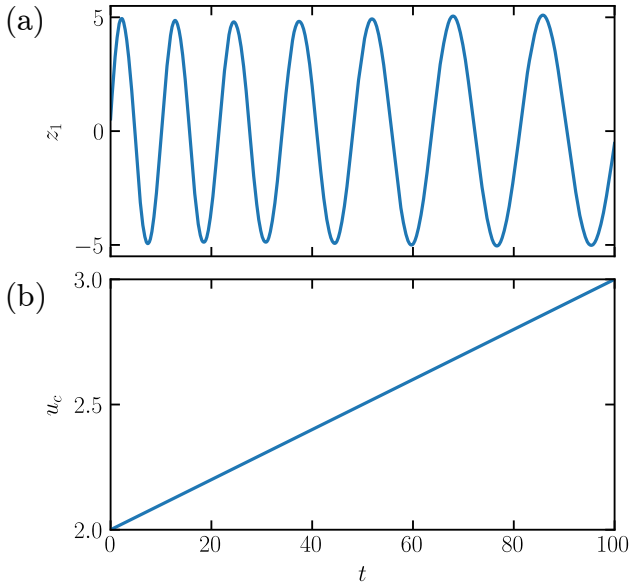

Figure 3.15: Induction of unseen signal outputs by a context-like external input.
The network has been trained similar to pretraining in task (i) to generate sinusoidal oscillations with three different frequencies in response to three constant external context inputs $u_c(t)$. After training, the weights are fixed and the network receives a continuously rising $u_c(t)$ (b). This results in a sinusoidal signal output with continuously rising period, which interpolates between the trained signals (a).

3.A.8 Pretraining with weight perturbation

Introduction. Throughout this chapter reservoir computing with FORCE learning is used for pretraining. In the following we show dynamical learning of simple tasks in networks that are pretrained with a biologically more plausible rule. Specifically, we use reservoir computing with weight perturbation [54, 55, 83, 151] to learn network structures that enable the dynamical learning of fixed points in two-dimensional space. We note that the direct application of a recent node perturbation scheme [60] to the output or all neurons was hindered by difficulties with learning multiple targets (cf. also [152]). Weight perturbation is, in short, a local reinforcement learning rule that consists of three steps: (i) randomly perturbing the connection weights, (ii) comparing the obtained reward with the reward expected without perturbation, and (iii) changing the connection weights into the direction (opposite direction) of the perturbation if the actual reward is higher (lower) than the expected one. In Chapter 4, we perform a thorough comparison between weight and node perturbation.

Structure learning with weight perturbation. We use batch learning, i.e the pretraining phase consists of N_{trials} trials, each of which is comprised of the presentation of all N_{tar} pretraining members of the task family $\tilde{z}(t;s)$ for a time t_{stay}. The signal output weights o_z learn as follows: At the beginning of trial n, the weights $o_{z,ki}(n-1)$ from the end of the previous trial receive small perturbations $\Delta o^{\text{pert}}_{z,ki}(n)$ [55]. The perturbations are drawn from a normal distribution with zero mean and standard deviation σ. We define the reward $R_z(n)$ as the negative sum of the mean squared errors between the signals and their targets during an evaluation period that starts t_{off} after the beginning of the trial. Further, we approximate the reward of the unperturbed network on the training batch by an exponentially weighted average $\bar{R}_z(n-1) = \alpha \bar{R}_z(n-2) + (1-\alpha) R_z(n-1)$ of previous rewards with timescale α [60]. This gives the estimate

$$G_{ki}(n) = \frac{\Delta o^{\text{pert}}_{z,ki}(n)}{\sigma^2}\left(R_z(n) - \bar{R}_z(n-1)\right) \tag{3.8}$$

for the weight gradient [55]. When we obtain the weight updates $\Delta o_{z,ki}(n)$ directly from this estimate, in our model we observe poor performance. It improves markedly when we combine the estimate with the Adam algorithm [153]. Adam introduces a momentum term $v_{ki}(n)$ and an individual learning rate $1/\sqrt{g_{ki}(n)+\mu}$ for each connection such that our weight update equations read

$$o_{z,ki}(n) = o_{z,ki}(n-1) + \Delta o_{z,ki}(n), \tag{3.9}$$

$$\Delta o_{z,ki}(n) = \eta \frac{v_{ki}(n)}{\sqrt{g_{ki}(n)+\mu}}, \tag{3.10}$$

$$v_{ki}(n) = \beta v_{ki}(n-1) + (1-\beta) G_{ki}(n), \tag{3.11}$$

$$g_{ki}(n) = \gamma g_{ki}(n-1) + (1-\gamma) G^2_{ki}(n). \tag{3.12}$$

Here, η is the global learning rate, μ a constant preventing overly large weight updates and β and γ are the timescales of the exponential averaging of the momenta and the learning rates, respectively. Learning of the context output weights is implemented likewise.

Results. At the end of the pretraining trials, our networks have learned to produce (in response to the signal error input) signal and context outputs that are close to the pretrained targets within the evaluation period, see Fig. 3.16. The established underlying network structures also enable the network to dynamically learn previously unseen targets. Like for the networks pretrained with FORCE, a short

presentation of the (here constant) target signal via the error input teaches the network to imitate it and to choose an appropriate context. During a subsequent testing period, the network autonomously continues the desired signal stabilized by the fixated context. Fig. 3.17a shows the testing error after dynamical learning of different signals. It is low for target positions within and slightly beyond the range of the pretrained targets. Fig. 3.17b shows signal outputs, which were dynamically learned by a single network instance.

Discussion. We have shown that for a simple task pretraining can also be performed with a learning rule that satisfies main criteria for biological plausibility, as it is local and causal. Further, it relies on delayed, sparse rewards and updates the weights at a low rate at the end of a trial. It is biologically plausible that synapses tentatively change their weights and then consolidate or reverse the change, depending on reward [154]. To improve learning, we have employed momentum and individual, history dependent learning rates for each connection. Supported by experimental findings it has already been argued that the brain could realize learning with momentum [155]. Furthermore there is ample evidence for a complex history dependence of learning rates in individual synapses [156]. While our weight modifications do not rely on a continuous supervisory signal anymore, such a signal is still present in the error input, like during dynamical learning. Future work may investigate how it can be replaced by sparse supervision.

Task details. $N = 1000, N_z = 2, N_c = 1, N_u = 0, p = 0.1, \tilde{w} = 1, t_{\text{stay}} = 100, t_{\text{fb}} = 50, t_{\text{learn}} = 50, t_{\text{test}} = 1000, t_{\text{off}} = 25, N_{\text{trials}} = 10^5, N_{\text{tar}} = 5, \sigma = 10^{-4}, \alpha = 1/3, \beta = 0.99, \gamma = 0.99, \mu = 10^{-8}$, $\eta = 50 \times 10^{-5}$ for the first 5000 trials, $\eta = 10 \times 10^{-5}$ for trials 5000 to 50000, $\eta = 1 \times 10^{-5}$ afterwards. The model and application details described in Sections 3.A.2 and 3.2 also apply to the current setting, except for those concerning the weight-learning rule. At the beginning of each pretraining trial for each member of the batch we draw the initial activation variables x_i from a uniform distribution between -0.1 and 0.1. The network learns to generate a constant output positioned on a curve in two-dimensional space parameterized by s. The family of target trajectories (fixed points) is $\tilde{z}(t; s) = \left(\frac{s^3}{2} + s_{\text{off}}, \frac{s}{2} + s_{\text{off}}\right)$, where the offset $s_{\text{off}} = 2.5$ ensures that the network feedback is strong enough to entrain the reservoir network. We use four different teacher trajectories for pretraining with parameters s chosen between 0 and 1 such that the corresponding $\tilde{z}(t; s)$ lie equidistantly on the target curve $\{\tilde{z}(t; s) | s \in [0, 1]\}$. The corresponding context targets are distributed equidistantly between 2 and 3.

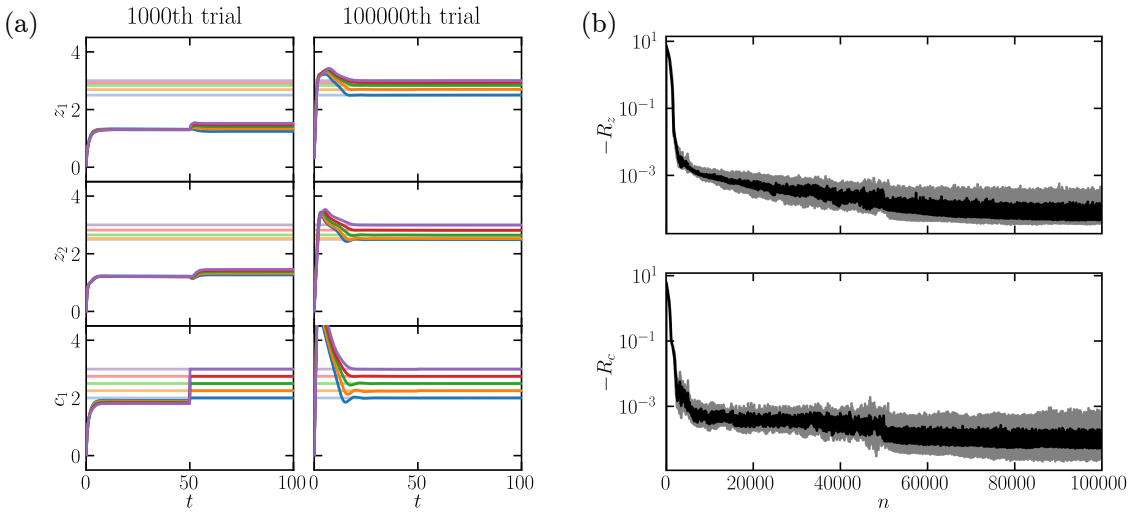

Figure 3.16: Pretraining using weight perturbation.
(a) Signal and context outputs for the different batch members early (left) and late (right) during pretraining for a single network instance. In late trials the networks' error input induces a quick convergence of the outputs (strong colors) to their targets (light colors). At $t = 50$, the context variable is fixed to its desired value.
(b) Negative reward for the signal (top) and context (bottom) output. Black line shows median value of 10 network instances and gray area indicates interquartile range between first and third quartile.

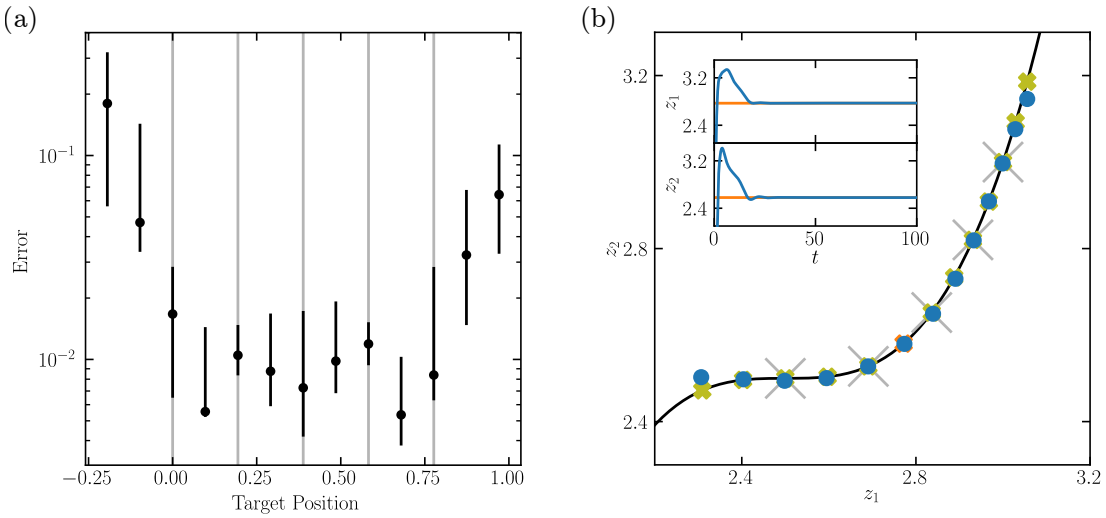

Figure 3.17: Quality of dynamical learning after pretraining with weight perturbation.
(a) Testing error between signal output and target as a function of the target position. Vertical gray lines indicate the positions of the pretrained targets. Dots show median value and errorbars represent the interquartile range between first and third quartile, using 10 network instances.
(b) Single network instance learning the same set of targets as in (a). The inset shows signal outputs (blue) versus time during dynamical learning and truncated testing periods for one of the targets (orange). In the main panel blue dots indicate the last signal outputs during testing. Yellow crosses indicate the position of the corresponding targets. They are mostly covered by blue dots, except in the regions of larger error. The orange cross indicates the position of the target shown in the inset. Gray crosses indicate the positions of the pretrained targets. The black line shows the curve $\tilde{z}(t;s)$ on which the targets lie.

3.A.9 Supplementary discussion

We conclude with an extended discussion of our findings, previous literature and possible future applications. Overall, we have shown how neural networks can quickly learn trajectories and dynamical systems without changing their weights and without requiring a teacher during testing. During the pretraining (learning-to-learn), the networks are taught several dynamics from the same family as the later dynamically learned ones, as well as a corresponding constant context. The process is supervised by an error signal to the synapses and, part of the time, by an error input to the network. During dynamical learning, a short presentation of the latter alone suffices to teach the desired dynamics. The network then also generates a context, which fluctuates around some temporal mean. When subsequently testing the generation of the dynamics, the error input is removed and the context is fixed to its average, telling that the learned dynamics should be continued.

Our analysis indicates that the scheme works due to an interplay of generalization and stabilization: During pretraining, the networks adapt to perform a negative feedback/autoencoder task. During dynamical learning, they generalize this, by generating a new desired signal when receiving its error as input. Simultaneously they choose a consistent context. During testing, this context is externally kept constant, which stabilizes the learned signal. This is possible because a mutual association between contexts and targets has been weight-learned during pretraining.

Approaches to supervised dynamical learning in the literature consider the one-step prediction of time series [34, 35] and input-output maps [33, 37–39, 41, 43], where the correct previous output is fed in. Other networks could adapt to provide negative feedback for control [36, 45, 46, 157], a pretrained oscillation [141], periodic sequences of discrete states [140] or the parameters of a dynamical system [139]. The studies use simple recurrent neural networks [34–36, 41, 45, 46, 139, 157], gated [38, 39, 43] or spiking ones [33], trained by backpropagation [33, 38, 39, 43] or extended Kalman filtering [34–36, 40, 44–46, 139, 157]. The simple networks are similar to ours but use non-leaky neurons, different learning and often assume discrete time. To our knowledge, all the systems with continuous signal space were fed a form of the temporally variable teaching signal also during testing.

Earlier work showed that sufficiently large recurrent neural networks with static weights can approximate any smooth input-output dynamics relation with bound-restricted inputs for finite time [29, 158, 159]. This implies that a network with static weights can in principle approximate the output of another, weight-learning one. The static network's dynamics thereby include the effects of the other network's learning algorithm and thus learns dynamically [30, 31]. Our networks with static weights are not pretrained to approximate during dynamical learning the outputs of weight-learning networks. In particular they do not approximate the outputs of a FORCE weight-learning reservoir computer, as illustrated by the different convergence properties in Fig. 3.12.

In our networks, fixing the intrinsically chosen context $c(t)$ indicates that the dynamics are to be continued. This is analogous to fixing the weights during testing in weight-learning paradigms. It is necessary to avoid convergence to other dynamics (if the system has discrete attractors) or diffusion (for marginally stable dynamics). The instruction to fix $c(t)$ is independent of the task and much simpler than task specific teacher and target signals. The constant $c(t)$ can be stored and kept up by biologically plausible circuits [160]. For long times, weight-learning may consolidate it. $c(t)$ may be understood as a (continuous) memory variable [133, 161]. In contrast to previous ones it is neither a pure feedback output [161, 162] nor an external input (cf. Section 3.A.7 and [136]) and it does not facilitate weight-learning [136, 161, 162]. One can also drive networks with external input such that

unseen, interpolating input leads to interpolating dynamics (cf. Section 3.A.7 and [142]). In contrast to such generalization, our networks learn their new dynamics by imitating a teacher. In particular, they adopt the phase of an oscillatory target.

Our pretraining implements a form of structure learning [25, 129], i.e. learning of the structure (concepts) underlying a family of tasks, which in general facilitates subsequent learning of new representatives. In our networks it enables learning of representatives without synaptic modification. Experiments indicate that animals and humans employ structure learning for example for motor tasks, which requires presentation of a variety of representative tasks and involves a reduction of the dimensionality of the search space, as in our model [25, 129]. Evolution or network plasticity should implement structure learning in biology. Since their functioning is largely unknown, we employ a simple reservoir computing scheme and comparably small neural networks. Only the readout weights, a small fraction of the network weights, are trained. Our dynamically learned tasks have similar difficulty as those used to introduce FORCE weight learning [125]. They are low-dimensional; this may often be the relevant case for biological neural networks, e.g., when learning movements [103].

In experimental physics and engineering, our scheme may find application in neuromorphic computing. Here, intrinsically plastic weights are costly and often difficult to realize, while outsourcing the learning to external controllers introduces computational bottlenecks [163]. As an example, in analog, photonic neuromorphic computing, network weights are externally set to generate desired output dynamics [51–53]. Our scheme may allow such systems to intrinsically learn and thereby fully reap their speed benefits. For spiking hardware, our networks may be efficiently translated into spiking ones [164]. Dynamical learning may reduce the size and power consumption of such hardware, for example in autonomous robots [165].

Our approach suggests a new method for the prediction of chaotic systems [47, 48], which searches for similarity within a predefined family of dynamics and leaves the networks structurally invariant and flexible.

A possible example for dynamical learning in biology is the quick learning of new movements [25, 28], perhaps with subsequent consolidation by plasticity. Another example may be short-term memory of single items and temporal sequences [26, 166]. Our theory predicts that even complicated dynamics may be memorized in biological neural networks without synaptic modification.

CHAPTER 4

Perturbation-based learning of temporally extended tasks

Parts of this chapter are included in the following manuscript:

[2] P. Züge, **C. Klos** and R.-M. Memmesheimer
Weight perturbation learning outperforms node perturbation on broad classes of temporally extended tasks
bioRxiv (2021):2021.10.04.463055

The following sections contain the parts of this work that include significant contributions from me. I modified and extended them substantially compared to the manuscript. The theoretical analysis presented in the following was conceptualized by all authors and originally performed by Paul Züge. I modified, condensed and partly simplified it to better fit the scope of the chapter. The simulated learning experiments presented in the following were conceptualized by all authors and performed by me. I extended them substantially compared to the manuscript.

4.1 Introduction

In the previous chapter, we have presented a scheme for biologically plausible fast learning. It relies on pretrained connection weights, which store acquired knowledge of similar tasks. For the pretraining, we primarily used the biologically implausible FORCE learning, but also employed weight perturbation (WP) [54, 55], a biologically plausible, activity-independent learning rule, in an example task. WP and in particular the related node perturbation (NP) [56, 57] are widely used models for reinforcement learning in the brain [54, 56–60, 167]. From a biological perspective, they are favorably because (i) they make use of the ubiquitous noise in the brain, (ii) they are local learning rules, (iii) they are, especially in the case of WP, potentially relatively simple to implement, and (iv) they are applicable to a wide variety of tasks. From a theoretical perspective, they are additionally popular because they allow for analytical exploration and are optimal in the sense that the perturbation-averaged weight change is to linear order equal to the exact weight gradient.

Briefly, in WP learning, random perturbations are added to the weights, or other network parameters. Then, the weights are changed into the (in the opposite) direction of the perturbation if the perturbation

leads to increased (decreased) reward compared to the unperturbed network. In NP learning, on the other hand, random perturbations are added to the summed weighted inputs of each neuron. Then the weights are updated according to the change in network performance and a perturbation- and activity-dependent eligibility trace assigned to each weight. Hence, for WP the dimension of the perturbation space is equal to the number of weights. For NP it is equal to the typically much smaller number of neurons. The latter only holds for trials without temporal extent, however. To find the direction of the true gradient, one thus needs to search for it in a much larger space in the case of WP compared to NP. Therefore, NP is considered to be superior for reinforcement learning in neural networks [19, 56–62]. Werfel et al. [56] confirmed this argument analytically for single-layer networks consisting of M linear perceptrons (see Section 2.2.2) with N random inputs that performed a student-teacher task. Specifically, the task was to adjust the connection weights such that the network reproduces the output of a teacher network with given weights for any input. In other words, the task was to learn the teacher weights. They found that the optimal error convergence rate of WP is worse than that of gradient descent (GD) by a factor equal to the number of weights (NM) in the network. On the other hand, for NP the convergence rate is worse than that of GD by a factor equal to the number of neurons (M). So WP is worse than NP by a factor equal to the number of input nodes (N).

However, this reasoning assumes that tasks are not extended in time and that inputs are high-dimensional. In biologically relevant settings, the opposite is the case (see Section 2.1.5 and refs. [9, 10, 57, 60, 168, 169]). Further, recent experimental studies show strong spontaneous weight changes, which could underlie WP [22–24]. Finally, our results from the previous chapter indicate that WP can be employed for complicated, temporally extended tasks. In this chapter, we thus investigate systematically how WP and NP perform on temporally extended tasks with variable dimensionality. Specifically, using Wick's theorem, we derive analytical expressions for the error dynamics in such tasks in linear, single-layer perceptrons, the same networks as used by Werfel et al. [56]. We find that the relative performance of WP and NP shifts in favor of WP. For some task settings, WP even outperforms NP. To test if these results extend to more complicated tasks and networks, we then apply WP and NP to a delayed non-match-to-sample task (DNMS), which serves as a simple nonlinear, working memory-reliant decision making task in both experiments [5, 6] and neural network modeling [60, 170]. Further, we apply it to MNIST, which is a standard benchmark task in machine learning [17, 171]. We find that WP also often outperforms NP in these settings. In fact, NP appears to be more susceptible to deviations of the network architecture from linear, single-layer networks as used for our theoretical investigation.

This chapter is structured as follows. In Section 4.2, we formally introduce the WP and NP learning rules. In Section 4.3, we analytically derive the error dynamics in simplified settings. In Section 4.4, we apply WP and NP to a DNMS task and MNIST. Finally, in Section 4.5, we discuss our results.

4.2 Learning rules

To introduce the learning rules, we consider a layer of M rate neurons with no intrinsic time dynamics. They receive N inputs. This layer may be part of a multi-layer perceptron, but could also be part of a network with a different architecture. We consider the learning of tasks that are temporally extended. At the end of each trial of the task, the neural networks receive feedback about their performance in the form of a scalar error (negative reward) feedback E [55, 57, 60, 61, 172]. For the upcoming theoretical analysis, we assume that time is split into discrete steps, indexed by $t = 1, \ldots, T$, where T

is the duration of a trial. Thus, the output firing rate of a neuron i, $i = 1, \ldots, M$, at the t-th time bin in response to inputs r_{jt}, $j = 1, \ldots, N$, is given by

$$z_{it} = g(y_{it}) = g\left(\sum_{j=1}^{N} w_{ij} r_{jt}\right). \tag{4.1}$$

Here, w_{ij} is the connection weight, y_{it} is the total input current and g is the generally nonlinear activation function. We note that the linear summation of the individual synaptic input currents $w_{ij} r_{jt}$ is a requirement for the NP scheme [56, 57, 151, 173].

Weight perturbation

In the standard form of WP learning (Fig. 4.1a), each trial of the task is performed once with unperturbed weights and once with perturbed weights. In the latter case, one adds before the start of the trial temporally static weight perturbations ξ_{ij}^{WP}, which are independent and identically distributed (iid) Gaussian random variables with zero mean and variance σ_{WP}^2, to the weights w_{ij} [55, 172]. The output of neuron i in the perturbed trial then reads

$$z_{it}^{\mathrm{pert,WP}} = g\left(\sum_{j=1}^{N} (w_{ij} + \xi_{ij}^{\mathrm{WP}}) r_{jt}^{\mathrm{pert,WP}}\right). \tag{4.2}$$

Here, the inputs $r_{jt}^{\mathrm{pert,WP}}$ may be perturbed because of upstream weights that are perturbed. The difference $E^{\mathrm{pert}} - E$ between the errors of the perturbed and unperturbed trial can then be used to estimate the gradient. If it is negative (positive), the projection of the weight perturbation onto the reward gradient $-\frac{\partial E}{\partial w_{ij}}$ is positive (negative). Thus, one updates the weights in the (opposite) direction of the perturbation if the error difference is negative (positive). Specifically, after the trial

$$\Delta w_{ij}^{\mathrm{WP}} = -\frac{\eta}{\sigma_{\mathrm{WP}}^2} (E^{\mathrm{pert}} - E) \xi_{ij}^{\mathrm{WP}}, \tag{4.3}$$

where η is the learning rate, is added to weight w_{ij} for all i and j. Assuming E is differentiable with respect to the weights, $E^{\mathrm{pert}} - E \approx \sum_{m=1}^{M} \sum_{k=1}^{N} \frac{\partial E}{\partial w_{mk}} \xi_{mk}^{\mathrm{WP}}$ to linear order. Inserting this expression into Eq. (4.3) shows that the WP updates are on average parallel to the reward gradient:

$$\langle \Delta w_{ij}^{\mathrm{WP}} \rangle \approx -\eta \frac{\partial E}{\partial w_{ij}}, \tag{4.4}$$

where $\langle \cdot \rangle$ denotes the average over the perturbations. This also holds if one would use a different baseline in Eq. (4.3) than E. Using the error of the unperturbed trial as the baseline has the advantage that it minimizes the variance of $\Delta w_{ij}^{\mathrm{WP}}$ (Section 4.A.1). In the brain, E is not available, however, because there is always only one set of weights. For the DNMS task, we thus use an average over the previous, perturbed errors as the baseline for biological plausibility.

WP can be applied to any system that maps parameters w onto a scalar error function E. This makes it very widely applicable, but leaves room for improvement by taking the specifics of the network structure into account. However, WP does take into account that the parameters are fixed during a

trial, as the weight perturbations are constant during a trial.

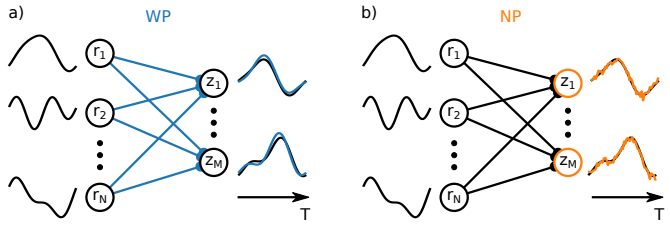

Figure 4.1: Schematic setup of WP and NP for a linear, single-layer network. Specifically, the M outputs z_i are a weighted sum of the N inputs r_j.
(a) WP perturbs the weights at the beginning of a trial; the resulting perturbations of the weighted sums of the inputs and thus the outputs reflect the dimensionality and smoothness of the inputs (blue).
(b) NP perturbs the weighted sum of the inputs with dynamical noise (orange).

Node perturbation

Similar to WP, in the standard form of NP learning (Fig. 4.1b), each trial of the task is performed once with unperturbed nodes and once with perturbed nodes. In the latter case, one adds node perturbations ξ_{it}^{NP}, which are independent and identically distributed (iid) Gaussian random variables with zero mean and variance σ_{NP}^2, at each time step to the total input current [56, 57]. In contrast to WP, the perturbations have to be time dependent, because for temporally static perturbations only the temporal mean of the total input current would be varied. The output of neuron i in the perturbed trial then reads

$$z_{it}^{\mathrm{pert,NP}} = g\left(\sum_{j=1}^{N} w_{ij} r_{jt}^{\mathrm{pert,NP}} + \xi_{it}^{\mathrm{NP}}\right). \tag{4.5}$$

As for WP, the inputs $r_{jt}^{\mathrm{pert,NP}}$ may be perturbed because of upstream nodes that are perturbed and the difference $E^{\mathrm{pert}} - E$ between the errors of the perturbed and unperturbed trial can be used to estimate the gradient. To do so, one computes for each weight an eligibility trace $\sum_{t=1}^{T} \xi_{it}^{\mathrm{NP}} r_{jt}$ and updates the weights according to

$$\Delta w_{ij}^{\mathrm{NP}} = -\frac{\eta}{\sigma_{\mathrm{NP}}^2}(E^{\mathrm{pert}} - E) \sum_{t=1}^{T} \xi_{it}^{\mathrm{NP}} r_{jt}. \tag{4.6}$$

For differentiable E, one can approximate the error difference between the perturbed and the unperturbed trial to linear order with $E^{\mathrm{pert}} - E \approx \sum_{i=1}^{M} \sum_{t=1}^{T} \frac{\partial E}{\partial y_{it}} \xi_{it}^{\mathrm{NP}}$. Inserting this expression into Eq. (4.6) and averaging over the distribution of the perturbations shows that also the NP updates are on average parallel to the reward gradient:

$$\langle \Delta w_{ij}^{\mathrm{NP}} \rangle \approx -\eta \frac{\partial E}{\partial w_{ij}}. \tag{4.7}$$

As for WP, this also holds for any error baseline, but using E minimizes the update noise (Section 4.A.1).

In contrast to WP, NP utilizes the fact that the inputs sum linearly in the networks we consider. It does

so by effectively incorporating an error backpropagation step, i.e. $\frac{\partial E}{\partial w_{ij}} = \sum_t \frac{\partial E}{\partial y_{it}} \frac{\partial y_{it}}{\partial w_{ij}} = \sum_t \frac{\partial E}{\partial y_{it}} r_{jt}$. Hence, one can perturb only the total input currents instead of individual weights, which led to the expectation that NP's learning performance is superior compared to WP's [19, 57, 60, 61].

4.3 Theoretical analysis

4.3.1 Error dynamics for a single input pattern

For our theoretical analysis, we consider linear, single-layer feed-forward networks consisting of M linear perceptrons with N inputs. The task is to learn the mapping of a single fixed input pattern r of duration T with elements r_{jt} to a target output pattern z^* with elements z^*_{it} generated by a teacher network. Thus, the input and target is the same for each trial. This can be motivated by three arguments: First, some biological motor tasks require such a mapping (see Section 4.5). Second, it is the opposite extreme to the case considered by Werfel et al. [56], who assumed that trials are not extended in time and that input changes in each trial (see Section 4.1). Third, our findings can be relatively easily extended to settings where the inputs vary between different trials (Section 4.3.2).

The output of the network is given by Eq. (4.1) with g equal to the identity function,

$$z = wr, \tag{4.8}$$

where w is the weight matrix. The target output z^* can be generated with target weights w^*, i.e. $z^* = w^* r$ and the error is given by the quadratic deviation of each output from its target,

$$E = \frac{1}{2T}\|z - z^*\|_2^2 = \frac{1}{2T}\mathrm{tr}[(z - z^*)(z - z^*)^T] = \frac{1}{2}\mathrm{tr}[WSW^T]. \tag{4.9}$$

Here, $W = w - w^*$ is the weight mismatch matrix and $S = \frac{1}{T} rr^T$ is the input correlation matrix [9]. Note that with this quadratic error function, the average weight updates (see Eqs. (4.4) and (4.7)) are exactly equal to the gradient for both WP and NP.

To investigate the influence of the dimensionality of the inputs, we assume that they are composed of N_{eff} orthogonal latent inputs, i.e. that S has N_{eff} nonzero eigenvalues. In the following we refer to N_{eff} as the effective input dimension. Note that N_{eff} is bounded from above by T, because there are at most T linearly independent vectors of length T. For clarity, we assume that the first N_{eff} inputs are the latent inputs while all other inputs are zero (Fig. 4.2). In other words, we hypothetically "rotate" the inputs. Since the error is invariant under rotations, this has no effect on the learning dynamics. Therefore, the weight space can be divided into an MN_{eff}-dimensional subspace of task-relevant and an $M(N - N_{\mathrm{eff}})$-dimensional subspace of task-irrelevant weights. This distinction will prove valuable for the interpretation of our theoretical results. To simplify the theoretical analysis, we further assume that the input strength of all nonzero inputs is $\alpha^2 = \frac{1}{T}\sum_{t=1}^{N} r_{jt}^2$, i.e. the N_{eff} nonzero eigenvalues of S are the same.

Derivation of error dynamics

In the following, we derive the evolution of the expected error $\langle E(n) \rangle$, where n indexes the trial, for WP, NP and for comparison also GD. To this end, we determine a recurrence relation for the error

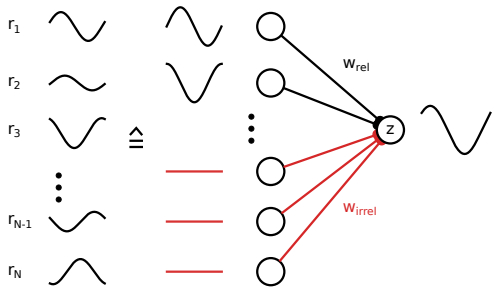

Figure 4.2: Hypothetical rotation of inputs.
The inputs (left, black) of linear networks can be rotated such that for our tasks the first N_{eff} inputs are nonzero and agree with the latent inputs (middle black). The remaining inputs are then zero (middle red) and their weights irrelevant for the output (right, red).

based on an expansion of $E(n)$ in terms of Δw:

$$\begin{aligned}
\langle E(n)\rangle &= \left\langle \tfrac{1}{2}\mathrm{tr}[W(n)SW^T(n)]\right\rangle \\
&= \left\langle \tfrac{1}{2}\mathrm{tr}[(W(n-1)+\Delta w(n-1))\,S\,(W(n-1)+\Delta w(n-1))^T]\right\rangle \\
&= \langle E(n-1)\rangle + \left\langle \mathrm{tr}[W(n-1)S\Delta w^T(n-1)]\right\rangle + \tfrac{1}{2}\left\langle \mathrm{tr}[\Delta w(n-1)S\Delta w^T(n-1)]\right\rangle \\
&= \langle E(n-1)\rangle + \underbrace{\left\langle \mathrm{tr}[WS\Delta w^T]\right\rangle}_{\equiv\left\langle \Delta E^{\text{lin}}_{\Delta w}\right\rangle} + \underbrace{\left\langle \tfrac{1}{2}\mathrm{tr}[\Delta wS\Delta w^T]\right\rangle}_{\equiv\left\langle \Delta E^{\text{quad}}_{\Delta w}\right\rangle}. \qquad (4.10)
\end{aligned}$$

Here, $\Delta w(n)$ is the matrix of weight updates after trial n. In the last line of the above expression and in the following, we omit the dependence on the trial number for clarity if it is unambiguous to do so. To evaluate $\Delta E^{\text{lin}}_{\Delta w}$ and $\Delta E^{\text{quad}}_{\Delta w}$, in addition to $S = \frac{1}{T}rr^T$ we will use that $S^2 = \alpha^2 S$ and $\mathrm{tr}[S] = \alpha^2 N_{\text{eff}} = \alpha^2_{\text{tot}}$, where α^2_{tot} is the total input strength. This follows from our assumptions about the structure of S.

For WP and NP, we will encounter terms of the form $\left\langle \ldots \mathrm{tr}[\xi X^T]\ldots\mathrm{tr}[\xi Y\xi^T]\ldots\right\rangle$, where ξ is the matrix of weight or node perturbations and X and Y are matrices with compatible dimensions. Such terms can be evaluated with the help of Wick's theorem (also known as Isserlis' theorem) [174]. It states how the expected value of a product of k zero-mean Gaussian random variables can be computed: If k is odd, the expected value is 0. If k is even, it is equal to the sum over all possible ways to partition the random variables into distinct pairs, where each summand (Wick contraction) is the product of the covariances of the pairs. Thus, the above given terms are 0 if there is an odd number of ξs. Otherwise one needs to combine the ξs into pairs. If the two ξs of a pair are in different traces, these traces are merged, e.g. $\left\langle \mathrm{tr}[X^T\xi^T]\mathrm{tr}[\xi X]\right\rangle = \sigma^2\mathrm{tr}[X^TX]$. If they are in the same trace, one gets a prefactor equal to the number of summands of the trace, e.g. $\left\langle \mathrm{tr}[\xi Y\xi^T]\right\rangle = \sigma^2 M\mathrm{tr}[Y]$.

Next, we evaluate Eq. (4.10) by inserting the respective weight updates into $\Delta E^{\text{lin}}_{\Delta w}$ and $\Delta E^{\text{quad}}_{\Delta w}$. In doing so, we omit the specifiers GD, WP and NP for clarity.

Gradient descent. For GD, the weight update reads

$$\Delta w = -\eta \frac{\partial E}{\partial w} = -\eta W S. \tag{4.11}$$

As the update is deterministic, the perturbation-averaging in Eq. (4.10) can be omitted and one gets

$$\begin{aligned} \Delta E^{\text{lin}}_{\Delta w} &= \text{tr}[WS\Delta w^T] \\ &= -\eta \text{tr}[WS^2W^T] \\ &= -2\eta\alpha^2 E(n-1), \end{aligned} \tag{4.12}$$

$$\begin{aligned} \Delta E^{\text{quad}}_{\Delta w} &= \tfrac{1}{2}\text{tr}[\Delta w S \Delta w^T] \\ &= \tfrac{1}{2}\eta^2 \text{tr}[WS^3W] \\ &= \eta^2\alpha^4 E(n-1). \end{aligned} \tag{4.13}$$

Inserting these expressions into Eq. (4.10) yields

$$E(n) = E(n-1)a, \tag{4.14}$$

$$E(n) = E(0)a^n, \tag{4.15}$$

where a is the convergence factor given by

$$a = (1-\eta\alpha^2)^2. \tag{4.16}$$

Weight perturbation. For WP, the error of the perturbed trial is

$$\begin{aligned} E^{\text{pert}} &= \tfrac{1}{2}\text{tr}[(W+\xi)S(W+\xi)^T] \\ &= E + \text{tr}[WS\xi^T] + \tfrac{1}{2}\text{tr}[\xi S \xi^T]. \end{aligned} \tag{4.17}$$

Thus, the weight update reads

$$\begin{aligned} \Delta w &= -\frac{\eta}{\sigma^2}(E^{\text{pert}} - E)\xi \\ &= -\frac{\eta}{\sigma^2}(\text{tr}[WS\xi^T] + \tfrac{1}{2}\text{tr}[\xi S\xi^T])\xi. \end{aligned} \tag{4.18}$$

Inserting into $\Delta E^{\text{lin}}_{\Delta w}$ yields

$$\begin{aligned} \left\langle \Delta E^{\text{lin}}_{\Delta w} \right\rangle &= \left\langle \text{tr}[WS\Delta w^T] \right\rangle \\ &= -\frac{\eta}{\sigma^2}\left(\left\langle \text{tr}[WS\xi^T]^2 \right\rangle + \tfrac{1}{2}\left\langle \text{tr}[WS\xi^T]\text{tr}[\xi S\xi^T] \right\rangle \right) \\ &= -\frac{\eta}{\sigma^2}\left\langle \text{tr}[WS\xi^T]\text{tr}[\xi S W^T] \right\rangle \\ &= -\eta\alpha^2 \left\langle \text{tr}[WSW^T] \right\rangle \\ &= -2\eta\alpha^2 \left\langle E(n-1) \right\rangle. \end{aligned} \tag{4.19}$$

For $\Delta E^{\text{quad}}_{\Delta w}$, we obtain

$$\begin{aligned}
\left\langle \Delta E^{\text{quad}}_{\Delta w} \right\rangle &= \tfrac{1}{2} \left\langle \text{tr}[\Delta w S \Delta w^T] \right\rangle \\
&= \frac{\eta^2}{2\sigma^4} \left\langle \left(\text{tr}[WS\xi^T] + \tfrac{1}{2}\text{tr}[\xi S \xi^T] \right)^2 \text{tr}[\xi S \xi^T] \right\rangle \\
&= \frac{\eta^2}{2\sigma^4} \left(\left\langle \text{tr}[WS\xi^T]^2 \text{tr}[\xi S \xi^T] \right\rangle + \tfrac{1}{4} \left\langle \text{tr}[\xi S \xi^T]^3 \right\rangle \right) \\
&= \frac{\eta^2}{2\sigma^4} \Big(\sigma^4 \left\langle M \text{tr}[WS^2W^T]\text{tr}[S] + 2\text{tr}[WS^3W^T] \right\rangle \\
&\qquad + \tfrac{1}{4}\sigma^6 \left(M^3 \text{tr}[S]^3 + 6M^2 \text{tr}[S^2]\text{tr}[S] + 8M\text{tr}[S^3] \right) \Big) \\
&= \eta^2\alpha^4 \left((MN_{\text{eff}} + 2) \left\langle E(n-1) \right\rangle + \tfrac{1}{8}\alpha^2\sigma^2 \left(M^3 N_{\text{eff}}^3 + 6M^2 N_{\text{eff}}^2 + 8MN_{\text{eff}} \right) \right). \qquad (4.20)
\end{aligned}$$

To get from the third to the fourth line, we evaluate the expectation value over the perturbation in the present trial. The expectation value over the perturbations of the previous trials remains. Inserting Eqs. (4.19) and (4.20) into Eq. (4.10) then yields

$$\langle E(n) \rangle = \langle E(n-1) \rangle \, a + b, \qquad (4.21)$$

$$\langle E(n) \rangle = \left(\langle E(0) \rangle - E_f \right) a^n + E_f, \qquad (4.22)$$

$$E_f = \frac{b}{1-a}, \qquad (4.23)$$

where E_f is the final (residual) error. a is the convergence factor and b the per-update error increase given by

$$a = 1 - 2\eta\alpha^2 + \eta^2\alpha^4(MN_{\text{eff}} + 2), \qquad (4.24)$$

$$b = \tfrac{1}{8}\eta^2\sigma^2\alpha^6 \left(M^3 N_{\text{eff}}^3 + 6M^2 N_{\text{eff}}^2 + 8MN_{\text{eff}} \right). \qquad (4.25)$$

Node perturbation. As the difference between output and target matrix for the perturbed trial of NP is $z^{\text{pert}} - z^* = Wr + \xi$, one obtains for the error of the perturbed trial

$$\begin{aligned}
E^{\text{pert}} &= \tfrac{1}{2T}\text{tr}[(Wr + \xi)(Wr + \xi)^T] \\
&= E + \tfrac{1}{T}\text{tr}[Wr\xi^T] + \tfrac{1}{2T}\text{tr}[\xi\xi^T]. \qquad (4.26)
\end{aligned}$$

Thus, the weight update reads

$$\begin{aligned}
\Delta w &= -\frac{\eta}{\sigma^2}(E^{\text{pert}} - E)\xi r^T \\
&= -\frac{\eta}{T\sigma^2}(\text{tr}[Wr\xi^T] + \tfrac{1}{2}\text{tr}[\xi\xi^T])\xi r^T. \qquad (4.27)
\end{aligned}$$

Inserting into $\Delta E^{\text{lin}}_{\Delta w}$ yields

$$\begin{aligned}
\left\langle \Delta E^{\text{lin}}_{\Delta w} \right\rangle &= \left\langle \text{tr}[WS\Delta w^T] \right\rangle \\
&= -\frac{\eta}{T\sigma^2} \left(\left\langle \text{tr}[WSr\xi^T]\text{tr}[Wr\xi^T] \right\rangle + \tfrac{1}{2} \left\langle \text{tr}[WSr\xi^T]\text{tr}[\xi\xi^T] \right\rangle \right) \\
&= -\frac{\eta}{T\sigma^2} \left\langle \text{tr}[WSr\xi^T]\text{tr}[\xi r^T W^T] \right\rangle \\
&= -\eta\alpha^2 \left\langle \text{tr}[WSW^T] \right\rangle \\
&= -2\eta\alpha^2 \left\langle E(n-1) \right\rangle . \qquad (4.28)
\end{aligned}$$

For $\Delta E^{\text{quad}}_{\Delta w}$, we obtain

$$\begin{aligned}
\left\langle \Delta E^{\text{quad}}_{\Delta w} \right\rangle &= \tfrac{1}{2} \left\langle \text{tr}[\Delta w S \Delta w^T] \right\rangle \\
&= \frac{\eta^2}{2\sigma^4 T^2} \left\langle \left(\text{tr}[Wr\xi^T] + \tfrac{1}{2}\text{tr}[\xi\xi^T] \right)^2 \text{tr}[\xi r^T S r \xi^T] \right\rangle \\
&= \frac{\eta^2}{2\sigma^4 T^2} \left(\left\langle \text{tr}[Wr\xi^T]^2 \text{tr}[\xi r^T S r \xi^T] \right\rangle + \tfrac{1}{4} \left\langle \text{tr}[\xi\xi^T]^2 \text{tr}[\xi r^T S r \xi^T] \right\rangle \right) \\
&= \frac{\eta^2}{2\sigma^4} \left(\sigma^4 \left\langle M\text{tr}[WSW^T]\text{tr}[S^2] + 2\text{tr}[WS^3W^T] \right\rangle + \tfrac{1}{4}\sigma^6 \left(M^3 T + 6M^2 + 8M\frac{1}{T} \right) \text{tr}[S^2] \right) \\
&= \eta^2\alpha^4 \left((MN_{\text{eff}} + 2) \left\langle E(n-1) \right\rangle + \tfrac{1}{8}\sigma^2 \left(M^3 N_{\text{eff}} T + 6M^2 N_{\text{eff}} + 8M\frac{N_{\text{eff}}}{T} \right) \right) . \qquad (4.29)
\end{aligned}$$

Inserting Eqs. (4.28) and (4.29) into Eq. (4.10) yields

$$\left\langle E(n) \right\rangle = \left\langle E(n-1) \right\rangle a + b, \qquad (4.30)$$

$$\left\langle E(n) \right\rangle = \left(\left\langle E(0) \right\rangle - E_f \right) a^n + E_f, \qquad (4.31)$$

$$E_f = \frac{b}{1-a}, \qquad (4.32)$$

where E_f is the final (residual) error. a is the convergence factor and b the per-update error increase given by

$$a = 1 - 2\eta\alpha^2 + \eta^2\alpha^4(MN_{\text{eff}} + 2), \qquad (4.33)$$

$$b = \tfrac{1}{8}\eta^2\sigma^2\alpha^4 \left(M^3 N_{\text{eff}} T + 6M^2 N_{\text{eff}}^2 + 8M\frac{N_{\text{eff}}}{T} \right) . \qquad (4.34)$$

Comparison of error dynamics

As we have shown, the error dynamics for GD, WP and NP are given by

$$\left\langle E(n) \right\rangle = \left(\left\langle E(0) \right\rangle - E_f \right) a^n + E_f . \qquad (4.35)$$

For $0 < a < 1$, the average error $\langle E(n) \rangle$ thus converges exponentially at a rate $-\ln(a) > 0$ towards a final error of $E_f = \frac{b}{1-a}$. The convergence rate can be approximated by $-\ln(a) \approx 1 - a$ if a is close to

1, which is typically the case for WP and NP in our settings. To compare the learning rules and for the remainder of this section, we set η to the optimal learning rate η^*, which is defined to minimize a. This definition is chosen because it is conceptually straightforward and turns out to be relevant for the MNIST task. We further set the perturbation strengths σ^2_{WP} and σ^2_{NP} such that they yield the same total output variance $\sigma^2_{\mathrm{eff}} = \frac{1}{MT}\left\langle \|z^{\mathrm{pert}} - z\|_2^2 \right\rangle$, which we refer to as effective perturbation strength. This results in $\sigma^2_{\mathrm{NP}} = \sigma^2_{\mathrm{eff}}$ and $\sigma^2_{\mathrm{WP}} = \frac{1}{\alpha^2 N_{\mathrm{eff}}}\sigma^2_{\mathrm{eff}}$.

For GD, the optimal learning rate is $\eta^*_{\mathrm{GD}} = \frac{1}{\alpha^2}$, leading to $a^*_{\mathrm{GD}} = 0$. Hence, the error decays to $E_f = 0$ after the first update. For WP and NP, one finds

$$\eta^*_{\mathrm{WP}} = \eta^*_{\mathrm{NP}} = \frac{1}{(MN_{\mathrm{eff}} + 2)\alpha^2}, \tag{4.36}$$

leading to

$$a^*_{\mathrm{WP}} = a^*_{\mathrm{NP}} = 1 - \frac{1}{MN_{\mathrm{eff}} + 2}. \tag{4.37}$$

Thus, the learning rate is by a factor of $(MN_{\mathrm{eff}} + 2)$ worse than that of GD. This factor, which also determines the convergence rate of WP and NP, is generally smaller than the number of weights and larger than the number of nodes. Hence, unlike conjectured by the intuitive argument explained in Section 4.1, these quantities are insufficient to predict the performance of WP and NP. The scaling of the learning and the convergence rate can be explained by considering the fluctuations of the weight updates. They originate from the credit assignment problem of finding the gradient direction. WP cannot identify this direction and, thus, equally amplifies the perturbations of all MN weights (Eq. (4.18)). Therefore, all weights, including the MN_{eff} relevant weights, which affect the error, fluctuate. On the other hand, NP partially solves the credit assignment problem, as the eligibility traces are zero for connections starting at zero inputs (Eq. (4.27)). It projects the (M) T-dimensional node perturbations onto the N_{eff}-dimensional inputs, thereby restricting the weight updates to the MN_{eff}-dimensional subspace of relevant weights. Since only these relevant weights influence the error, the learning and convergence rate is the same for both WP and NP.

Furthermore, the per-update error increase b and the final error E_f are nonzero for WP and NP. This is because the error function is nonlinear, which leads to reward noise due to finite size perturbations. The effects become particularly apparent when the error is close to zero. In this case, any finite perturbation leads to a positive error difference $E^{\mathrm{pert}} - E$ and in turn to a weight update into the opposite direction of the perturbation. Therefore, the weights do not reach their optimal values and a nonzero final error E_f remains. At the optimal learning rate, the leading order term of the final error is

$$E_f^{\mathrm{WP}} = \frac{b^*_{\mathrm{WP}}}{1 - a^*} \approx \tfrac{1}{8}\sigma^2_{\mathrm{eff}} M^2 N_{\mathrm{eff}}, \tag{4.38}$$

$$E_f^{\mathrm{NP}} = \frac{b^*_{\mathrm{NP}}}{1 - a^*} \approx \tfrac{1}{8}\sigma^2_{\mathrm{eff}} M^2 T, \tag{4.39}$$

i.e. the final error of WP is smaller by a factor $N_{\mathrm{eff}}/T \leq 1$ compared to the final error of NP. Hence, a longer trial duration harms NP, while a larger effective input dimensionality harms WP. In short, the different scaling of the final errors is because for WP only MN_{eff} of the MN perturbations ξ^{WP}_{ij} affect the perturbed error, while for NP all MT perturbations ξ^{NP}_{it} affect it (see our manuscript [2] for more

details). The additional factor of M is a result of the general scaling of the error with M (Eq. (4.9)).

Taken together, we find that WP works just as well as or better than NP for the learning of a single temporally extended input-output mapping. The convergence rate is the same for WP and NP, but the final error is smaller or equal for WP compared to NP. The results are summarized in Table 4.1. We also confirm our results numerically for different values of N_{eff} and T in an example setting. We find that the simulation results agree well with our theoretical predictions (Fig. 4.3).

	GD	WP	NP
η^*	$\frac{1}{\alpha^2}$	$\frac{1}{(MN_{\text{eff}}+2)\,\alpha^2}$	$\frac{1}{(MN_{\text{eff}}+2)\,\alpha^2}$
a^*	0	$1-\frac{1}{MN_{\text{eff}}+2}$	$1-\frac{1}{MN_{\text{eff}}+2}$
E_f^*	0	$\approx \frac{1}{8}\sigma_{\text{eff}}^2 M^2 N_{\text{eff}}$	$\approx \frac{1}{8}\sigma_{\text{eff}}^2 M^2 T$
$\langle E(n)\rangle = \left(\langle E(0)\rangle - E_f\right) a^n + E_f$			$E_f = \frac{b}{1-a}$

Table 4.1: Theoretical results for the error dynamics of a network consisting of linear perceptrons that learn a single temporally extended input-output mapping. GD, WP and NP lead to an exponential decay of the error with convergence factor a, per-update increase b and final error E_f (bottom section). Top section shows results for optimal learning rates η^*, yielding the fastest convergence, and weight and node perturbations with the same effective perturbation strength σ_{eff}. The final error E_f^* is given in leading order. Other parameters: M: number of output neurons, N_{eff}: effective input dimensionality, T: trial length, α^2: input strength.

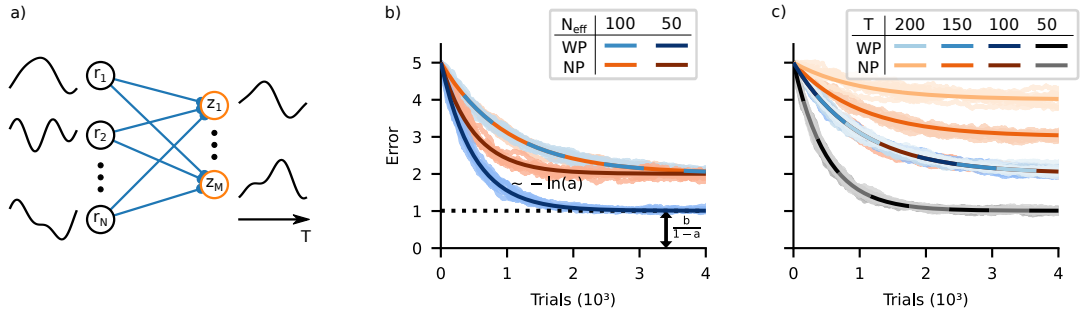

Figure 4.3: Learning of a single temporally extended input-output mapping in linear networks.
(a) Schematic of the linear network consisting of M output neurons and N inputs.
(b) WP (blue) works just as well or, in terms of the final error, better than NP (orange). The error decay time decreases for WP and NP likewise with decreasing N_{eff}. In contrast, the residual error only decreases for WP. Error curves from simulations (10 runs, shaded) agree well with analytical curves for the decay of the expected error (solid). For WP and $N_{\text{eff}} = 50$ the decay rate ($-\ln(a)$) and the final error (dashed line) are highlighted. Parameters: $M = 10$, $N = T = 100$, $N_{\text{eff}} \in \{100, 50\}$, $\sigma_{\text{eff}} = 4 \times 10^{-2}$, $\alpha^2 = N/N_{\text{eff}}$.
(c) Increased trial duration T does not change the progress of WP learning. In contrast, increasing T hinders NP learning by increasing the residual error. If T decreases N_{eff} (gray curves), convergence is faster and to a lower residual error in both WP (because of the decrease in N_{eff}) and NP (because of the decrease in N_{eff} and T). Again, simulations (10 runs, shaded) agree well with analytical curves. Parameters: $M = 10$, $N = 100$, $T \in \{200, 150, 100, 50\}$, N_{eff} is set to 100 but cannot be greater than T, so that $T = 50$ forces $N_{\text{eff}} = 50$, $\sigma_{\text{eff}} = 4 \times 10^{-2}$, $\alpha^2 = N/N_{\text{eff}}$.

4.3.2 Error dynamics for multiple input patterns

In the previous section, we considered the task of learning the mapping of a single input pattern to a single output pattern. In general learning tasks, however, inputs and targets may vary from trial to trial. Put differently, the network has to learn multiple subtasks. To examine how this affects WP and NP, we extend our theoretical analysis. For this purpose, we consider the same type of network as before, i.e. a single-layer feed-forward network consisting of M linear perceptrons with N inputs and output $z = wr$, where w is the weight matrix. The task is to learn multiple subtasks of the type described in the previous section. Specifically, for each subtask p, $p = 1, \dots, P$, the goal is to learn the mapping of an input pattern r_p of duration T to a target output pattern $z_p^* = w^* r_p$, where w^* is the target weight matrix. For the full task, the learning goal is to reduce the task-averaged quadratic error

$$E = \left\langle E_p \right\rangle_p = \left\langle \tfrac{1}{2T} \| z_p - z_p^* \|_2^2 \right\rangle_p = \left\langle \tfrac{1}{2} \mathrm{tr}[W S_p W^T] \right\rangle_p = \tfrac{1}{2} \mathrm{tr}[W S W^T]. \tag{4.40}$$

Here, $\langle \cdot \rangle_p$ denotes the expectation value over the subtasks, $W = w - w^*$ is again the weight mismatch matrix, $S_p = \frac{1}{T} r_p r_p^T$ is the input correlation matrix of pattern p and $S = \left\langle S_p \right\rangle_p$ is the task-averaged correlation matrix.

We again assume that the inputs of each subtask are composed of $N_{\mathrm{eff}}^{\mathrm{trial}}$ orthogonal and equally strong latent inputs, i.e. all S_p have $N_{\mathrm{eff}}^{\mathrm{trial}}$ nonzero eigenvalues given by α^2. Furthermore, we assume that the subtasks are pairwise orthogonal, $\frac{1}{T} r_p r_q^T = S_p \delta_{pq}$. Thus, the effective input dimension of the full task is $N_{\mathrm{eff}}^{\mathrm{task}} = P N_{\mathrm{eff}}^{\mathrm{trial}}$ and S has $N_{\mathrm{eff}}^{\mathrm{task}}$ nonzero eigenvalues given by α^2/P. As done above, for clarity we will hypothetically rotate the inputs, such that for each input pattern there is a distinct group of $N_{\mathrm{eff}}^{\mathrm{trial}}$ input units that are nonzero and equal to the latent inputs. For the same arguments as given above, this does not affect WP or NP learning. Therefore, there are $M N_{\mathrm{eff}}^{\mathrm{task}}$ task-relevant and $M N_{\mathrm{eff}}^{\mathrm{trial}}$ trial-relevant weights for each trial or subtask.

The precise order in which the different input patterns are presented only has a marked effect on the error dynamics if the timescale on which the error changes, $-\ln(a)^{-1} \approx (1-a)^{-1}$ (see Eq. (4.35)), is not much larger than P. If it is much larger, all subtasks are presented numerous times before the error changes substantially and the ordering of the input patterns is largely irrelevant. It turns out the latter is the case for WP and NP, but not necessarily for GD.

Finally, we note that the learning of multiple input patterns can also be considered as splitting a single task into multiple subtasks. For slowly changing errors, the content of the input pattern can be chosen randomly in each trial without significantly affecting the error dynamics. In other words, the input pattern presented in each trial may be constructed from a randomly chosen subset of $N_{\mathrm{eff}}^{\mathrm{trial}}$ out of the $N_{\mathrm{eff}}^{\mathrm{task}}$ latent inputs. In this case, P is defined as $P = N_{\mathrm{eff}}^{\mathrm{task}} / N_{\mathrm{eff}}^{\mathrm{trial}}$.

Derivation of error dynamics

In the following, we derive the evolution of the expected error $\langle E(n) \rangle$, where n indexes the trial, for WP, NP and GD. Assuming that the subtask presented in trial $n-1$ is indexed by q, we split the task

error into two parts:

$$\langle E(n)\rangle = \frac{1}{P}\left(\left\langle E_q(n)\right\rangle + \sum_{\substack{p=1\\p\neq q}}^{P}\left\langle E_p(n)\right\rangle\right). \tag{4.41}$$

As before, $\left\langle E_p(n)\right\rangle$ can be expanded in terms of the weight update Δw of the previous trial. Omitting the trial numbers if possible, we obtain

$$\left\langle E_p(n)\right\rangle = \left\langle E_p(n-1)\right\rangle + \underbrace{\left\langle \mathrm{tr}[WS_p\Delta w^T]\right\rangle}_{\equiv\left\langle \Delta E^{\mathrm{lin}}_{p,\Delta w}\right\rangle} + \underbrace{\left\langle \tfrac{1}{2}\mathrm{tr}[\Delta w S_p \Delta w^T]\right\rangle}_{\equiv\left\langle \Delta E^{\mathrm{quad}}_{p,\Delta w}\right\rangle}. \tag{4.42}$$

If $p = q$, $\Delta E^{\mathrm{lin}}_{p,\Delta w}$ and $\Delta E^{\mathrm{quad}}_{p,\Delta w}$ are the same as in the single input pattern case with $N_{\mathrm{eff}} = N^{\mathrm{trial}}_{\mathrm{eff}}$. To evaluate $\Delta E^{\mathrm{lin}}_{p,\Delta w}$ and $\Delta E^{\mathrm{quad}}_{p,\Delta w}$ when $p \neq q$, we use the same kind of relations as for the derivation in the single input pattern case.

Gradient descent. Inserting the weight update

$$\Delta w = -\eta\frac{\partial E_q}{\partial w} = -\eta W S_q \tag{4.43}$$

into Eq. (4.42) yields

$$E_p(n) = \begin{cases} E_p(n-1)(1-\eta\alpha^2)^2 & \text{if } p = q,\\ E_p(n-1) & \text{if } p \neq q, \end{cases} \tag{4.44}$$

where we used in the last line that $S_pS_q = 0$ for $p \neq q$. Inserting this expression into Eq. (4.41) gives

$$E(n) = E(n-1) - E_q(n-1)\frac{2\eta\alpha^2 - \eta^2\alpha^4}{P}. \tag{4.45}$$

If η is close to $1/\alpha^2$, the timescale of the error dynamics is not much larger than P and the precise ordering of the input patterns matters. Assuming cyclic subtask selection and that the initial error on all subtasks is relatively similar, $E_p(0) \approx E(0)$, one finds

$$E(n) = E(0)\left(1 - \frac{n}{P}(2\eta\alpha^2 - \eta^2\alpha^4)\right) \tag{4.46}$$

for $n \leq P$. Thus, the error dynamics are linear. If η has a value for which the error changes slowly, $E_p(0) \approx E(0)$ implies $E_p(n-1) \approx E(n-1)$ in Eq. (4.45), which leads to

$$E(n) = E(0)a^n, \tag{4.47}$$

$$a = 1 - \frac{2\eta\alpha^2}{P} + \frac{\eta^2\alpha^4}{P}. \tag{4.48}$$

Weight perturbation. For WP, the weight update reads

$$\begin{aligned}\Delta w &= -\frac{\eta}{\sigma^2}(E_q^{\text{pert}} - E_q)\xi \\ &= -\frac{\eta}{\sigma^2}(\text{tr}[WS_q\xi^T] + \tfrac{1}{2}\text{tr}[\xi S_q \xi^T])\xi. \end{aligned} \tag{4.49}$$

Inserting the weight update into Eq. (4.42) and using our results from the single input pattern case yields

$$\begin{aligned}\left\langle E_q(n)\right\rangle = \left\langle E_q(n-1)\right\rangle &(1 - 2\eta\alpha^2 + \eta^2\alpha^4(MN_{\text{eff}}^{\text{trial}} + 2)) \\ &+ \tfrac{1}{8}\eta^2\sigma^2\alpha^6\left(M^3{N_{\text{eff}}^{\text{trial}}}^3 + 6M^2{N_{\text{eff}}^{\text{trial}}}^2 + 8MN_{\text{eff}}^{\text{trial}}\right).\end{aligned} \tag{4.50}$$

For $p \neq q$, we obtain for $\Delta E^{\text{lin}}_{p,\Delta w}$ and $\Delta E^{\text{quad}}_{p,\Delta w}$ the following equations:

$$\left\langle \Delta E^{\text{lin}}_{p,\Delta w}\right\rangle = \left\langle \text{tr}[WS_p\Delta w^T]\right\rangle = 0, \tag{4.51}$$

$$\begin{aligned}\left\langle \Delta E^{\text{quad}}_{p,\Delta w}\right\rangle &= \tfrac{1}{2}\left\langle \text{tr}[\Delta w S_p \Delta w^T]\right\rangle \\ &= \eta^2\alpha^4\left(MN_{\text{eff}}^{\text{trial}}\left\langle E_q(n-1)\right\rangle + \tfrac{1}{8}\alpha^2\sigma^2\left(M^3{N_{\text{eff}}^{\text{trial}}}^3 + 2M^2{N_{\text{eff}}^{\text{trial}}}^2\right)\right).\end{aligned} \tag{4.52}$$

Inserting these expressions into Eq. (4.42) yields

$$\begin{aligned}\left\langle E_p(n)\right\rangle = \left\langle E_p(n-1)\right\rangle &+ \left\langle E_q(n-1)\right\rangle \eta^2\alpha^4 MN_{\text{eff}}^{\text{trial}} \\ &+ \tfrac{1}{8}\eta^2\sigma^2\alpha^6\left(M^3{N_{\text{eff}}^{\text{trial}}}^3 + 2M^2{N_{\text{eff}}^{\text{trial}}}^2\right).\end{aligned} \tag{4.53}$$

Finally, inserting Eqs. (4.50) and (4.53) into Eq. (4.41) and assuming that the error on all subtasks is relatively similar, $\left\langle E_p(n-1)\right\rangle \approx \langle E(n-1)\rangle$, gives

$$\langle E(n)\rangle = \left(\langle E(0)\rangle - E_f\right)a^n + E_f, \tag{4.54}$$

$$E_f = \frac{b}{1-a}, \tag{4.55}$$

$$a = 1 - \frac{2\eta\alpha^2}{P} + \frac{\eta^2\alpha^4(PMN_{\text{eff}}^{\text{trial}} + 2)}{P}, \tag{4.56}$$

$$b = \tfrac{1}{8}\eta^2\sigma^2\alpha^6\left(M^3{N_{\text{eff}}^{\text{trial}}}^3 + 2\frac{P+2}{P}M^2{N_{\text{eff}}^{\text{trial}}}^2 + 8\frac{MN_{\text{eff}}^{\text{trial}}}{P}\right). \tag{4.57}$$

Node perturbation. For NP, the weight update reads

$$\begin{aligned}\Delta w &= -\frac{\eta}{\sigma^2}(E_q^{\text{pert}} - E_q)\xi r_q^T \\ &= -\frac{\eta}{T\sigma^2}(\text{tr}[Wr_q\xi^T] + \tfrac{1}{2}\text{tr}[\xi\xi^T])\xi r_q^T. \end{aligned} \tag{4.58}$$

Inserting the weight update into Eq. (4.42) and using our results from the single input pattern case yields

$$\begin{aligned}\left\langle E_q(n)\right\rangle = \left\langle E_q(n-1)\right\rangle \left(1 - 2\eta\alpha^2 + \eta^2\alpha^4(MN_{\text{eff}}^{\text{trial}} + 2)\right) \\ + \tfrac{1}{8}\eta^2\sigma^2\alpha^4\left(M^3 N_{\text{eff}}^{\text{trial}} T + 6M^2 {N_{\text{eff}}^{\text{trial}}}^2 + 8M\frac{N_{\text{eff}}^{\text{trial}}}{T}\right).\end{aligned} \tag{4.59}$$

For $p \neq q$, we have $\Delta E_{p,\Delta w}^{\text{lin}} = 0$ and $\Delta E_{p,\Delta w}^{\text{quad}} = 0$ and therefore

$$\left\langle E_p(n)\right\rangle = \left\langle E_p(n-1)\right\rangle. \tag{4.60}$$

Finally, inserting Eqs. (4.59) and (4.60) into Eq. (4.41) and assuming that the error on all subtasks is relatively similar, $\left\langle E_p(n-1)\right\rangle \approx \langle E(n-1)\rangle$, gives

$$\langle E(n)\rangle = \left(\langle E(0)\rangle - E_f\right) a^n + E_f, \tag{4.61}$$

$$E_f = \frac{b}{1-a}, \tag{4.62}$$

$$a = 1 - \frac{2\eta\alpha^2}{P} + \frac{\eta^2\alpha^4(MN_{\text{eff}}^{\text{trial}} + 2)}{P}, \tag{4.63}$$

$$b = \tfrac{1}{8P}\eta^2\sigma^2\alpha^4\left(M^3 N_{\text{eff}}^{\text{trial}} T + 6M^2 {N_{\text{eff}}^{\text{trial}}}^2 + 8M\frac{N_{\text{eff}}^{\text{trial}}}{T}\right). \tag{4.64}$$

Comparison

For GD and cyclic subtask selection, the error decays (piecewise) linearly (Eq. (4.46)). The optimal learning rate is $\eta_{\text{GD}}^* = \frac{1}{\alpha^2}$, for which the error decays to zero in exactly P trials. This is because in each trial, the error gradient and therefore also the weight update (Eq. (4.43)) only has nonzero components for the weights relevant to the current subtask. Hence, one can choose the same learning rate as for the single input pattern case. The error of the current subtask thus decays to zero while the errors of all other subtasks are unchanged. The trial-dependent change of the gradient direction is known as gradient noise in machine learning and also affects WP and NP. Although it underestimates the performance of GD (at least for cyclic subtask selection), it is instructive to also consider the exponential approximation of GD's error dynamics (Eq. (4.47)) with optimal learning rate $\eta_{\text{GD}}^* = \frac{1}{\alpha^2}$. This results in $a_{\text{GD}}^* = 1 - \frac{1}{P}$ and a convergence rate of approximately $-\ln(a) \approx 1/P$ for large P.

As for the single input pattern case, WP and NP exhibit exponential error dynamics, $\langle E(n)\rangle = \left(\langle E(0)\rangle - E_f\right) a^n + E_f$. To compare the learning rules, we again set η to the optimal learning rate η^*, which minimizes a. Further, we set σ_{WP} and σ_{NP} such that they lead to the same effective perturbation strength $\sigma_{\text{eff}}^2 = \frac{1}{MT}\left\langle \|z_p^{\text{pert}} - z_p\|_2^2\right\rangle$ in each trial. This results in $\sigma_{\text{NP}}^2 = \sigma_{\text{eff}}^2$ and $\sigma_{\text{WP}}^2 = \frac{1}{\alpha^2 N_{\text{eff}}^{\text{trial}}}\sigma_{\text{eff}}^2$.

For WP, one finds

$$\eta^*_{\mathrm{WP}} = \frac{1}{(MN^{\mathrm{task}}_{\mathrm{eff}} + 2)\alpha^2}, \tag{4.65}$$

$$a^*_{\mathrm{WP}} = 1 - \frac{1}{P(MN^{\mathrm{task}}_{\mathrm{eff}} + 2)}. \tag{4.66}$$

Thus, the learning rate and, approximately, the convergence rate is by a factor of $(MN^{\mathrm{task}}_{\mathrm{eff}} + 2)$ worse than that of GD's exponential approximation. Again, this can be linked to the fluctuations of the weight updates. WP updates all weights such that the weights that are irrelevant for the current subtask, but potentially relevant for other subtasks, change randomly (Eq. (4.49)). Hence, while the error of the current subtask can improve (Eq. (4.50)), it deteriorates for the other subtasks (Eq. (4.53)). As there are $MN^{\mathrm{task}}_{\mathrm{eff}}$ task-relevant weights, the learning and convergence rate worsen by approximately this factor compared to GD's exponential approximation. The final error is $E^{\mathrm{WP}}_f = \frac{b^*_{\mathrm{WP}}}{1-a^*} \approx \frac{1}{8}\sigma^2_{\mathrm{eff}} M^2 N^{\mathrm{trial}}_{\mathrm{eff}}$, similar to the single input pattern case.

For NP, one finds

$$\eta^*_{\mathrm{NP}} = \frac{1}{(MN^{\mathrm{trial}}_{\mathrm{eff}} + 2)\alpha^2}, \tag{4.67}$$

$$a^*_{\mathrm{NP}} = 1 - \frac{1}{P(MN^{\mathrm{trial}}_{\mathrm{eff}} + 2)}. \tag{4.68}$$

NP only updates the weights that are relevant for the current subtask (Eq. (4.58)). Thus, the error of the current subtask can improve (Eq. (4.59)), while the errors of the other subtasks are unchanged (Eq. (4.60)). As there are $MN^{\mathrm{trial}}_{\mathrm{eff}}$ trial-relevant weights, the learning and convergence rate worsen by approximately this factor compared to GD's exponential approximation. The final error is $E^{\mathrm{NP}}_f = \frac{b^*_{\mathrm{NP}}}{1-a^*} \approx \frac{1}{8}\sigma^2_{\mathrm{eff}} M^2 T$, similar to the single input pattern case.

Our results also have important implications for learning of multiple actions such as sequences of movements [175]. They can be learned by splitting them into subsets, which are called (mini-)batches in machine learning. In our terminology, each batch corresponds to a subtask, the number of batches to P, the dimensionality of the input data to $N^{\mathrm{task}}_{\mathrm{eff}}$ and the batch size N_{batch} to $N^{\mathrm{trial}}_{\mathrm{eff}}$, assuming for simplicity that individual data points are pairwise orthogonal and have no time dimension. For $MN^{\mathrm{trial}}_{\mathrm{eff}} \gg 2$, Eqs. (4.66) and (4.68) thus imply that the convergence rate of NP is independent of the batch size while that of WP is proportional to the batch size and reaches NP's convergence rate for full batch learning. The same holds for the optimal learning rates as α^2 scales inversely with the batch size. As for the single input pattern case, longer trial durations T harm NP and larger effective trial input dimensionalities $N^{\mathrm{trial}}_{\mathrm{eff}}$ harm WP by linearly increasing E^*_f. The results are summarized in Table 4.2. Again, we confirm our theoretical results with numerical simulations for different values of P in an example setting (Fig. 4.4).

	GD	GD (exp. approx.)	WP	NP
η^*	$\frac{1}{\alpha^2}$	$\frac{1}{\alpha^2}$	$\frac{1}{(MN_{\text{eff}}^{\text{task}}+2)\,\alpha^2}$	$\frac{1}{(MN_{\text{eff}}^{\text{trial}}+2)\,\alpha^2}$
a^*	-	$1-\frac{1}{P}$	$1-\frac{1}{P(MN_{\text{eff}}^{\text{task}}+2)}$	$1-\frac{1}{P(MN_{\text{eff}}^{\text{trial}}+2)}$
E_f^*	-	0	$\approx \frac{1}{8}\sigma_{\text{eff}}^2 M^2 N_{\text{eff}}^{\text{trial}}$	$\approx \frac{1}{8}\sigma_{\text{eff}}^2 M^2 T$
$E(n) = E(0)\left(1-\frac{n}{P}\right)$		$\langle E(n)\rangle = \left(\langle E(0)\rangle - E_f\right) a^n + E_f$	$E_f = \frac{b}{1-a}$	

Table 4.2: Theoretical results for the error dynamics of a network consisting of linear perceptrons learning multiple temporally extended input-output mappings. Assuming cyclic subtask selection, GD leads to a linear decay of the error (bottom section). WP and NP lead to an exponential decay of the error with convergence factor a, per-update increase b and final error E_f (bottom section). Top section shows results for optimal learning rates η^*, yielding the fastest convergence, and weight and node perturbations with the same effective perturbation strength σ_{eff}. The final error E_f^* is given in leading order. Other parameters: M: number of output neurons, $N_{\text{eff}}^{\text{trial}}$: effective input dimensionality in each trial, $N_{\text{eff}}^{\text{task}}$: effective input dimensionality of the full task, T: trial length, α^2: input strength.

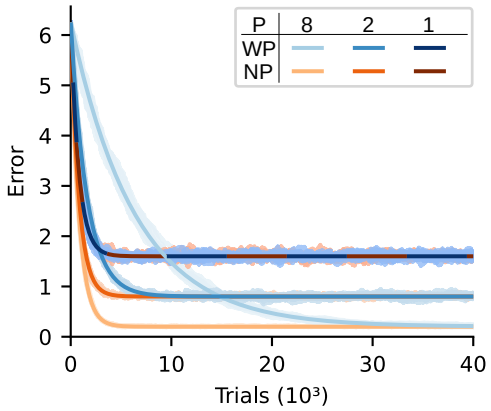

Figure 4.4: Learning of multiple temporally extended input-output mappings in linear networks.
Error dynamics scale differently for WP and NP when splitting the input into different patterns for different trials. The network is the same as the one used in Fig. 4.3 ($N = 100$, $M = 10$, $\sigma_{\text{eff}} = 4 \times 10^{-2}$). The task is to reproduce the output of a teacher network in response to input with a dimensionality of $N_{\text{eff}}^{\text{task}} = 80$, which we split into P non-overlapping input patterns each having dimensionality $N_{\text{eff}}^{\text{trial}} = N_{\text{eff}}^{\text{task}}/P$ (hence $PN_{\text{eff}}^{\text{trial}} = 80$). For simplicity, we use input patterns where at each timestep a different input unit has the value $\sqrt{N/N_{\text{eff}}^{\text{task}}}$, while all other input units are zero. This implies $T = N_{\text{eff}}^{\text{trial}}$. The figure shows error curves for WP (blue) and NP (orange) from simulations (10 runs, shaded) together with analytical curves for the decay of the expected error (solid) for different values of P (simulation results for NP with $P = 8$ are mostly covered by the analytical curve). Theoretical curves and simulations agree well.

4.4 Simulated learning experiments

In the following, we investigate numerically how WP and NP perform in more complicated settings with nonlinear and partly recurrent networks and compare the results to our analytical predictions. Specifically, we consider a biologically relevant delayed non-match-to-sample task (DNMS) and MNIST, which is a standard task in machine learning.

4.4.1 Delayed non-match-to-sample task

To ensure analytical tractability and for simplicity, so far we made a few biologically implausible assumptions. Specifically, only connection weights to linear units were trained and each trial consisted of a perturbed and an unperturbed run. In the following we show that our findings generalize to settings without these assumptions. To this end, we consider the learning of a DNMS task (temporal XOR) by nonlinear recurrent networks. DNMS tasks and closely related variants are widely used both in experiment [6] and theory [60, 170], where they serve as simple nonlinear, working memory-reliant decision making tasks.

Model

We use the same setting as [60], which shows that a new variant of NP is able to solve the DNMS task. In particular, the setting is not adjusted to WP. The network is initialized as a network of rate neurons of the type introduced in Section 2.2.2. It consists of $N = 200$ nonlinear rate neurons receiving input from two external units $u_1(t)$ and $u_2(t)$. One of the network neurons, whose rate we denote with $z(t)$, serves as its output (Fig. 4.5a). In each trial, the network receives two input pulses, where each pulse is a 200 ms long period with either $u_1(t)$ or $u_2(t)$ set to 1, and subsequently has to output 1 for 200 ms if different inputs were presented and -1 if the same inputs were presented (Fig. 4.5b). There is a 200 ms long delay period after each input pulse.

More specifically, the dynamics of neuron i, $i = 4, \dots, N$, are governed by

$$\tau \dot{x}_i = -x_i(t) + \sum_{j=1}^{N} w_{ij}^{\text{rec}} r_j(t) + \sum_{q=1}^{2} w_{iq}^{\text{in}} u_q(t), \tag{4.69}$$

with time constant $\tau = 30\,\text{ms}$. The constant activations $x_1(t) = x_2(t) = 1$ and $x_3(t) = -1$ provide biases [60]. The rate of each neuron i, $i = 1, \dots, N$, is given by $r_i(t) = \tanh(x_i(t))$. $z(t) = r_4(t)$ is the network output. We use the forward Euler-method with stepsize $dt = 1\,\text{ms}$ to simulate the dynamics and draw the initial activations from a uniform distribution, $x_i(0) \sim \mathcal{U}(-0.1, 0.1)$ for $i = 4, \dots, N$. Recurrent weights are drawn from a Gaussian distribution, $w_{ij}^{\text{rec}} \sim \mathcal{N}(0, g^2/N)$, with $g = 1.5$. Input weights are drawn from a uniform distribution, $w_{iq}^{\text{in}} \sim \mathcal{U}(-1, 1)$.

We train all recurrent weights w_{ij}^{rec} using the usual update rules (Eqs. (4.3) and (4.6)), but replace the error of the unperturbed trial by an exponential average of the errors of the previous trials of the same trial type [58–60]. Hence, each trial now only consists of a perturbed and not additionally an unperturbed run. The error function is the mean squared difference between the output $z(t)$ and the target within the last 200 ms of each trial. For each of the different trial types k, $k = 1, \dots, 4$, we use an exponential average of the previous errors $E^{\text{pert}}(n_k)$ for this trial type (n_k indexes the trials of type

k) as the error baseline:

$$E_k(n_k) = E_k(n_k - 1) + \frac{1}{\tau_E}(E^{\text{pert}}(n_k) - E_k(n_k - 1))\,, \tag{4.70}$$

where $\tau_E = 4$. To get the best performing learning parameters, we performed a grid search, which yielded $\eta^{\text{WP}} = 1 \times 10^{-5}$, $\sigma^{\text{WP}} = 4.64 \times 10^{-3}$, $\eta^{\text{NP}} = 1 \times 10^{-5}$, $\sigma^{\text{NP}} = 4.64 \times 10^{-1}$.

For both WP and NP, the exact perturbations ξ have to be accessible for the weight update, which seems biologically plausible for WP (see Section 4.5), but less so for NP (see Section 4.5 and [60]). Therefore we also compare WP and NP to the biologically plausible version of NP proposed by [60], which avoids this assumption: in the weight update rule, it approximates the exact node perturbations ξ^{NP} with a nonlinearly modulated difference between the momentary input to a neuron and its short term temporal average. For the details of this version of NP, see ref. [60]. Here we briefly mention the main differences to the vanilla NP version (Eq. (4.6)): For each network neuron a node perturbation is applied at a simulation time step only with a probability of 0.3 % and is drawn from a uniform distribution, $\xi \sim \mathcal{U}(-16, 16)$. The error is given by the absolute difference between output and target. Weight updates are computed via $\Delta w_{ij}^{\text{rec}}(n_k) = -\eta E_k(n_k - 1)\left(E^{\text{pert}}(n_k) - E_k(n_k - 1)\right)\sum_{t=1}^{T}\left[(x_{it} - \bar{x}_{it})\, r_{j,t-1}\right]^3$, and clipped when they exceed $\pm 3{\times}10^{-4}$ (see code accompanying ref. [60]). t indexes the simulation time step of each trial, T is the total number of simulation time steps per trial and $\bar{x}_{it} = \bar{x}_{i,t-1} + \frac{1}{\tau_x}(x_{it} - \bar{x}_{i,t-1})$ is an exponential average of past activations. Parameter values are $\eta = 0.1$, $\tau_x = \frac{20}{19}$.

We implement the model using Python and NumPy [146].

Results

Fig. 4.5c shows the performance of the three update rules in terms of their accuracy over the last 100 trials, where a trial is considered successful if the mean absolute difference between z and the target output is smaller than 1. We find that all update rules learn the task comparably well and reach perfect accuracy within at most 2000 trials when considering the median of network instances. Thus, our previous findings that WP can perform as well as or better than NP in simplified settings extend to the considered biologically plausible setup. That means WP can perform well for nonlinear neuron models, recurrent connectivity and when the error of the unperturbed network is not available. Furthermore, the results indicate that approximating the perturbation as in ref. [60] only mildly impacts the performance of NP for the considered task.

4.4.2 MNIST

Finally, we consider MNIST classification [17]. The MNIST dataset consists of images of handwritten single-digit numbers. The associated classification problem is to correctly link the images to the corresponding numbers. It is one of the most widely used benchmarks in machine learning. Here, we use WP and NP to train neural networks on batches of images. Each time step thereby corresponds to the presentation of one image and the networks receive error feedback only at the end of a batch. This allows us to test how well WP and NP work on a more complicated, temporally extended task and in networks with a multi-layer structure. In addition, it allows us to study how our analytical results for the learning of multiple input patterns (Table 4.2) extend to real-world tasks.

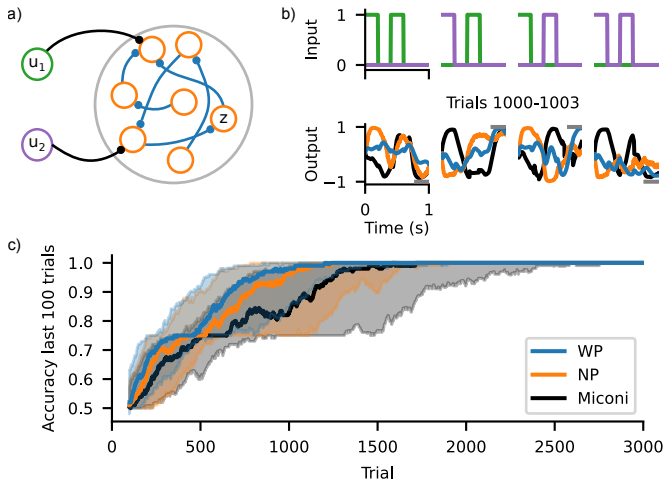

Figure 4.5: WP performs as well as NP on a DNMS task.
(a) Schematic of the recurrent network with inputs u_1 and u_2 and output z. All network weights are learned, i.e., for WP, all network weights (blue) are perturbed and for NP all network nodes (orange) are perturbed.
(b) Inputs and outputs during example trials. Top row: Inputs u_1 (green) and u_2 (purple) for the four different trial types. Bottom row: Outputs for WP (blue), NP (orange) and the version of NP proposed by ref. [60] (black) for trials 1000–1003 for the inputs shown above. Gray bars show target outputs.
(c) Accuracy during training. WP (blue) performs similarly well as NP (orange) and the version of NP used by ref. [60] (black). There is a noticeable transient slowdown at an accuracy of 75 %, which corresponds to the successful learning of three out of the four different trial types. Solid lines show the median and shaded areas represent the interquartile range between first and third quartile using 100 network instances.

Model

We use single-, two-, or three-layer, fully-connected feed-forward networks of nonlinear perceptrons (see also Section 2.2.2). The input layer consists of 784 units encoding the grayscale pixel values of the data. The hidden layers consist of 100 neurons with tanh activation function and biases. The output layer consists of 10 neurons, one for each single-digit number, with softmax activation function and biases. Specifically, the rates and input currents are given by

$$r^{(0)} = u, \tag{4.71}$$

$$y^{(l)} = w^{(l)} r^{(l-1)} + b^{(l)}, \tag{4.72}$$

$$r^{(l)} = \begin{cases} \tanh(y^{(l)}) & \text{if } l \neq L, \\ \frac{\exp(y^{(l)})}{\sum_{i=1}^{N^{(l)}} \exp(y_i^{(l)})} & \text{if } l = L. \end{cases} \tag{4.73}$$

Here, u is the input vector, $w^{(l)}$ is the weight matrix, $b^{(l)}$ the vector of biases, $y^{(l)}$ the input current vector and $r^{(l)}$ the rate or activation vector of layer l, $l = 1, \ldots, L$, where $L = 1, 2$ or 3. $N^{(l)}$ is the number of neurons in layer l. The output of the network is the activation of the last layer, $z = r^{(L)}$. Its ith element can be considered to encode the probability that the image label is $i - 1$. We initialize the weights and biases using standard LeCun initialization, i.e. weights and biases of layer l are drawn from a uniform distribution $\mathcal{U}(-\sqrt{1/N^{(l)}}, \sqrt{1/N^{(l)}})$.

We employ vanilla WP (Eq. (4.3)), NP (Eq. (4.6)) or stochastic GD (SGD) to train all parameters of the networks. T is equal to the batch size $N_{\text{batch}} \in \{1, 10, 100, 1000\}$. Hence, the perturbation is different for each image in a batch in the case of NP, while it is the same for WP. Updating the biases with WP is straightforward, while for NP we have to assume a presynaptic activity of 1 in Eq. (4.6). Unless noted otherwise, the error function for a single input image is the cross-entropy loss

$$E = -\sum_{i=1}^{10} p(\tilde{z} = i - 1|u) \log(z_i), \tag{4.74}$$

where $p(\tilde{z} = i - 1|u) = \delta_{i-1,\tilde{z}(u)}$ is a one-hot encoding of the target label $\tilde{z}(u)$. The error for an input batch is the cross-entropy loss averaged over the batch. We also tried to combine the gradient estimates obtained from WP and NP with Momentum, RMSProp or Adam [17], but did not find an improvement of performance compared to the vanilla versions with carefully tuned parameters. The same holds for SGD. This may be because of the rather simple network architecture.

We use the standard training and test data set, but split the standard training data set into a training data set of 50 000 images and a validation data set of 10 000 images. No preprocessing is done on the data. To obtain the best-performing parameters (the learning rate η for all three algorithms and the perturbation strengths σ^2_{WP} and σ^2_{NP} for WP and NP), we perform a grid search for each of the considered batch sizes and network structures: For each parameter set we train the network for 50 000 trials (i.e.: weight updates) on the training data set. We then select the best-performing parameter sets based on the final accuracy on the validation data set and apply them to the test data set. We find that high final accuracy appears to concur with fast convergence speed (Fig. 4.6), such that a comparison to our analytical results (where learning rate optimizes the convergence speed) seems justified. In addition, the perturbation strength has little impact on performance, indicating that the final error is not restricted by reward noise due to finite size perturbations (Table 4.2).

We implement the model using Python and PyTorch [176].

Results

Fig. 4.7 shows the performance of WP, NP and SGD on the test data set for the different considered network architectures and batch sizes (see Section 4.A.2 for numerical values of network performance after learning and direct comparisons of the learning performance for fixed L or N_{batch}). For WP the performance improves drastically with increasing batch size. The final test accuracy is only about 65 %–70 % (depending on L) for $N_{\text{batch}} = 1$ but reaches 91 %–93 % for $N_{\text{batch}} = 1000$. Simultaneously the optimal learning rate increases strongly, by a factor of roughly 100 (Fig. 4.6 and Tables 4.3 to 4.5). For comparison: the biologically implausible stochastic gradient descent (SGD) rule reaches accuracies of 91 %–98 % for the considered batch sizes. In contrast, the learning curves of NP appear to be entirely independent of the batch size; for, e.g., two-layer networks, the final test accuracy is always about 86 % and the optimal learning rate is roughly constant as well (Fig. 4.6 and Table 4.4). Hence, WP often outperforms NP, especially for large batch sizes.

The improvement of WP with batch size and NP's independence of it are in agreement with our theoretical analysis (Table 4.2). However, from this analysis we also expected that WP's learning rate can reach at most that of NP for large batch size. NP's slower convergence suggests that it is more susceptible to deviations of the network architecture from linear, single-layer networks. Indeed, increasing the number of network layers has a comparably small effect on the performance of WP,

while NP's performance decreases considerably (final test accuracy decreases from 89 % to 78 % when increasing L from 1 to 3, Fig. 4.10).

For single-layer networks there still remains a noticeable performance difference between WP and NP (91 % vs. 89 % final test accuracy for $N_{\text{batch}} = 1000$). This remaining discrepancy from our theory might be a result of the unrealizable part of the output targets. The targets are most likely not fully realizable because it is impossible for the networks to exactly output the one-hot encoded target and because the MNIST dataset contains images that are very hard to classify, even for humans, due to poor handwriting. Unrealizable targets are harmful for NP, as it perturbs the nodes with white noise, leading to a perturbed network output z^{pert} that might be impossible to reproduce by the unperturbed network given the input. This unrealizable part of the perturbed output can, however, make the unrealizable part of the targets seem realizable. The resulting change of E^{pert} is non-instructive and represents reward noise to the updates. Consequently, the final error of NP is expected to increase. WP does not suffer from unrealizable targets as its perturbations always lead to network outputs that are also realizable by the unperturbed network. Our manuscript includes a theoretical analysis of this effect for linear, single-layer perceptrons [2].

To check if unrealizable output targets are a possible explanation for the remaining performance difference between WP and NP, we modify the learning task to make the targets realizable. In a first step, we change the original target labels to target labels determined by the maximal output of a teacher network that was trained on MNIST using SGD (Fig. 4.8). This should negate the unrealizable target outputs stemming from poor handwriting. As WP still outperforms NP, we additionally change the cross-entropy loss with one-hot encoded targets to a mean-squared error loss with targets given by the raw output of the same teacher network. We find that this recovers the prediction of our analysis: NP performs better than WP even for large batch sizes (Fig. 4.8).

Thus, our results show that WP can work better than NP on realistic tasks and for multi-layer networks. This is particularly remarkable when naively comparing the number of perturbed nodes and weights: For, e.g. the two-layer network, there are only 110 output and hidden nodes, but 79 510 weights (including biases). Nevertheless, WP can clearly outperform NP. Also a comparison of the actual perturbation dimensions cannot explain the better performance of WP for, e.g., $N_{\text{batch}} = 100$ (WP pert. dim.: 79 510, NP pert. dim.: $110 \times T = 11\,000$).

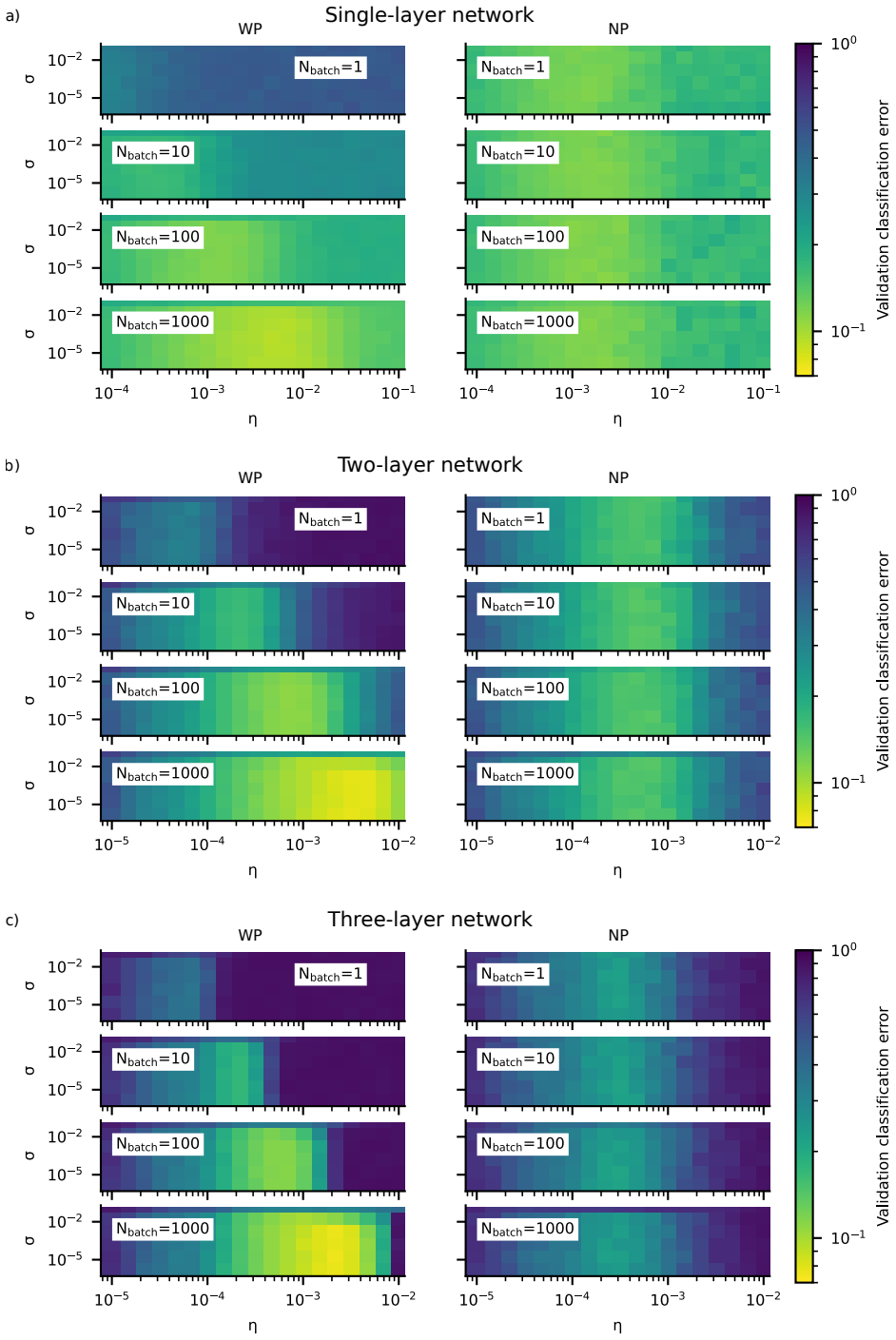

Figure 4.6: Grid search results for WP and NP.
(a) Validation classification error (one minus validation accuracy) for different learning parameters for WP (left) and NP (right) for single-layer networks ($L = 1$). Both WP and NP perform similarly well for a wide range of perturbation strengths σ indicating that the final error is not restricted by reward noise due to finite size perturbations (Table 4.2). Increasing the batch size makes it possible to increase the learning rate in the case of WP but not NP. Errors are averaged over 5 network instances.
(b, c) Same as (a) but for two-layer (b, $L = 2$) and three-layer (c, $L = 3$) networks.

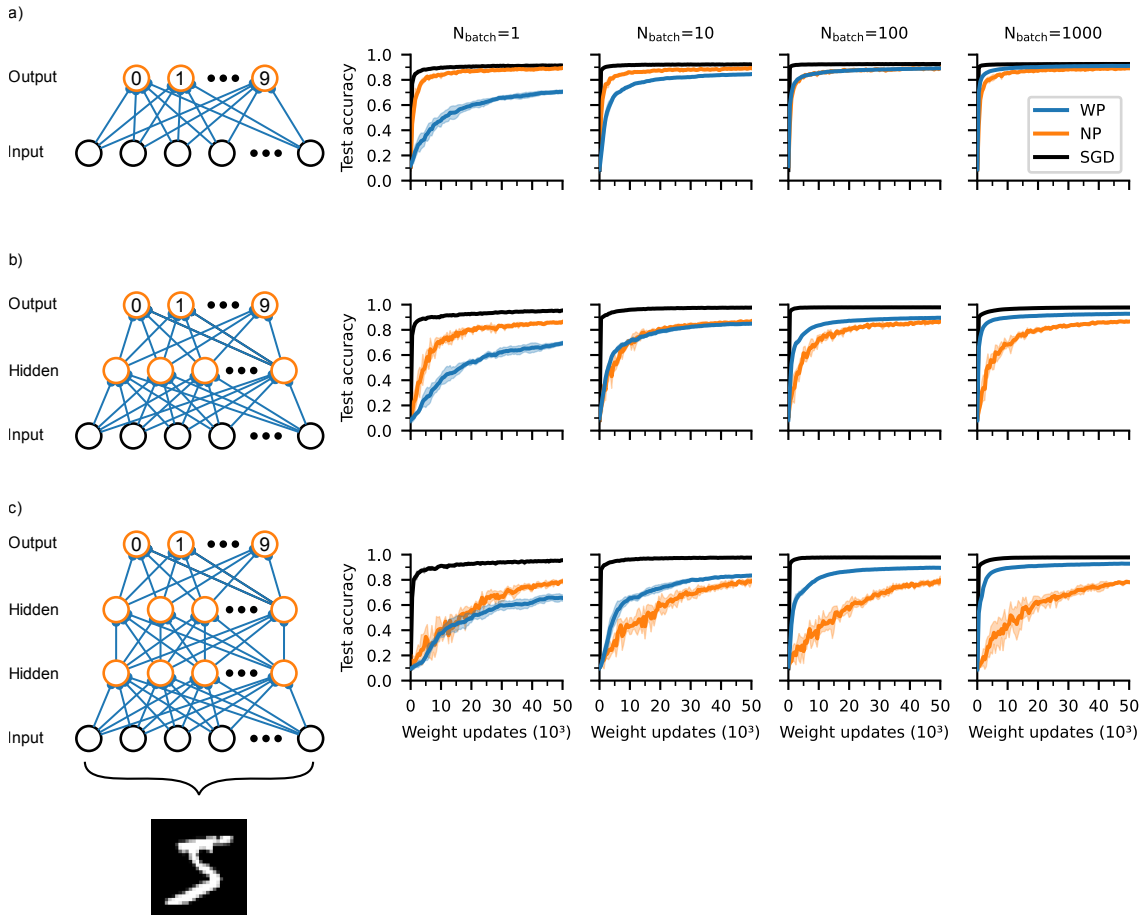

Figure 4.7: WP can outperform NP on MNIST.
(a) Results for single-layer networks ($L = 1$). Left: Network schematic. All network weights are learned, i.e., for WP all network weights (blue) are perturbed and for NP all network nodes (orange) are perturbed. Right: Test accuracy as a function of the number of weight updates for WP (blue), NP (orange) and SGD (black) for different batch sizes. NP does not profit from increasing the batch size and always reaches a final accuracy of ~89 %. WP improves considerably with increasing batch sizes and reaches a final accuracy of ~91 % for $N_{\text{batch}} = 1000$. Solid lines show the mean and shaded areas show the standard deviation using 5 network instances.
(b) Same as (a) but for two-layer networks ($L = 2$). NP's final accuracy decreases compared to single-layer networks, reaching ~86 % independent of batch size. WP's final accuracy stays relatively constant, reaching ~92 % for $N_{\text{batch}} = 1000$.
(c) Same as (a) but for three-layer networks ($L = 3$). NP's final accuracy further decreases to ~78 % independent of batch size. WP again reaches a final accuracy of ~92 % for $N_{\text{batch}} = 1000$.

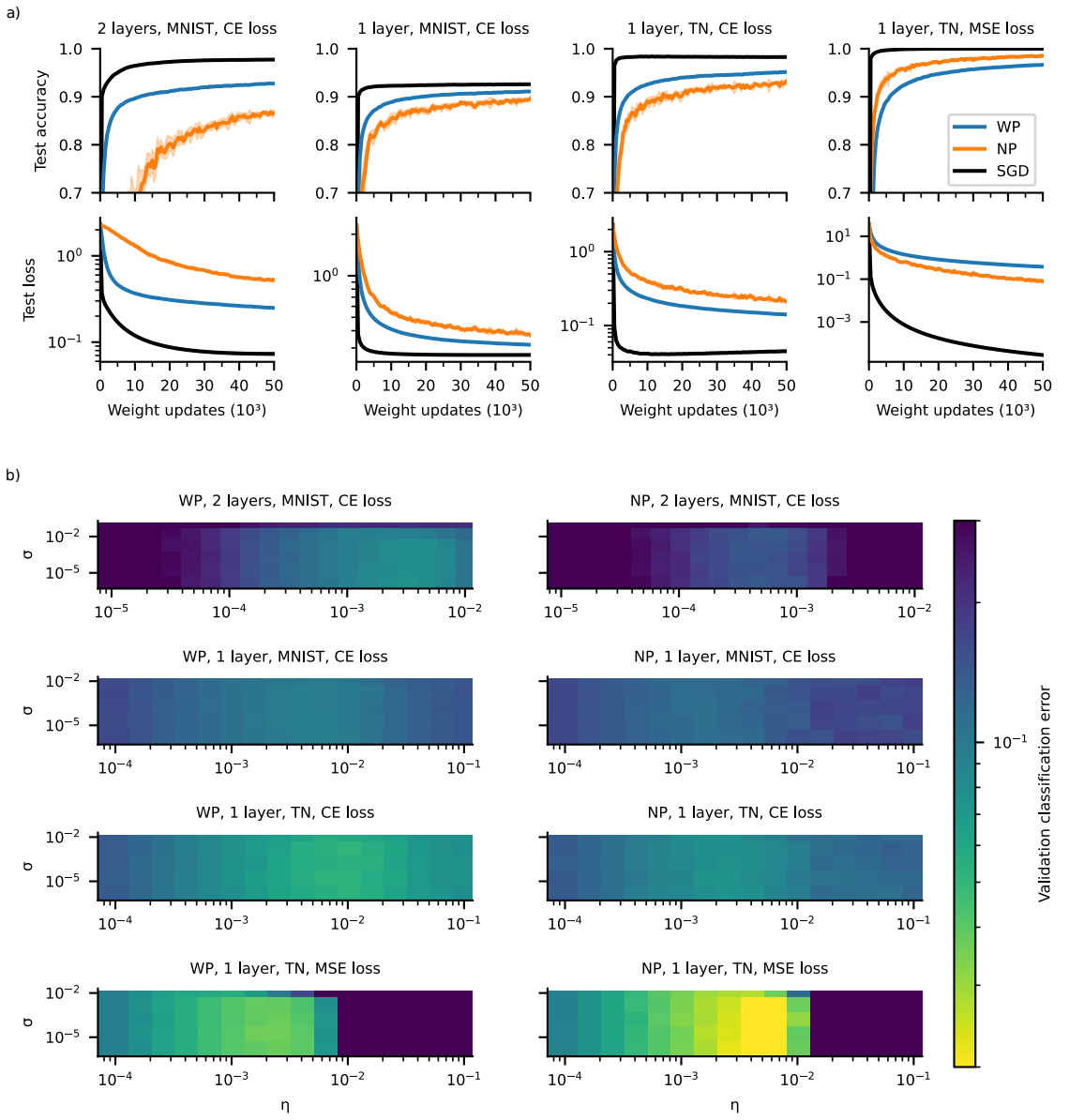

Figure 4.8: Learning performance and grid search results for four variations of the MNIST task. Specifically, for a two-layer network trained on MNIST using cross-entropy loss (2 layers, MNIST, CE loss), a single-layer network trained the same way (1 layer, MNIST, CE loss), a single-layer network with the MNIST images as input but with target labels determined by the maximal output of a teacher network (1 layer, TN, CE loss) and a single-layer network with the MNIST images as input using mean-squared error loss with targets given by the raw output of the same teacher network (1 layer, TN, MSE loss). $N_{\text{batch}} = 1000$.
a) Test accuracy (upper row) and loss (lower row) for WP (blue), NP (orange) and SGD (black) for the best performing learning parameters. Removing the hidden layer worsens the performance of WP and SGD but improves it for NP (compare first to second column). Using a teacher network to create the target labels does not change the relative performance of WP and NP, indicating that potentially unrealizable target labels do not significantly harm NP (compare third to second column). Further removing all nonlinearities from the networks leads to better performance of NP compared to WP (compare fourth to third column). Lines show the mean and shaded areas the standard deviation using 5 network instances.
b) Grid search results for WP and NP as given by the mean classification error for 5 instances on a validation data set not used for training. The error is clipped at 0.3 for better visualization.

4.5 Discussion

Our results show that WP performs better than NP for tasks where long trials capture most of the task's content. This might seem paradoxical as NP incorporates more structural knowledge on the network, namely the linear summation of inputs. However, WP accounts for the fact that the weights in a neural network are (approximately) static. Further, by perturbing the weights it implicitly accounts for low input dimensionality and generates only realizable output changes. Therefore it generates better tentative perturbations. This leads to less noise in the reward signal and better performance (smaller final error and sometimes faster convergence) in the tasks where WP is superior to NP.

Our theoretical analysis shows that the relative performance of WP and NP highly depends on the effective input dimensionality and duration of the task, i.e. N_{eff} and T. In biology, task-related neural dynamics appear to be confined to a low-dimensional space (see Section 2.1.5), indicating low effective input dimensionality N_{eff}. For example, the dynamics for simple movements as investigated in typical experiments are embedded in spaces of dimension of order 10 [102]. Furthermore, neurons under in vivo conditions can faithfully follow input fluctuations on a timescale of 10 ms [177] and significant changes in neuronal trajectories happen on a timescale of 100 ms [102, 103, 178]. For the learning of a movement of duration 1 s, this suggests a number of time bins T of about 10 to 100. Thus, N_{eff} and T are roughly on the same order suggesting both WP and NP as promising candidates for the learning of simple movements. If the movements are longer lasting, our results indicate that WP will be superior.

As another concrete example, consider the learning of the single song in certain birds. A single, stereotypical input sequence in a "conductor area" (HVC) may drive the circuit [179, 180]. The effective input dimension N_{eff} is thus at most as large as the temporal dimension T of the task. Based on recent experiments, ref. [180] proposed that the output of the tutor/experimenter area (LMAN) is modified by reinforcement learning via NP, such that it guides the motor area (RA) to learn the right dynamics. Our analytical results predict that WP is as well or better suited to achieve this task since $N_{\text{eff}} \leq T$. Earlier work suggested that WP [172] or NP [61] may directly mediate the learning of the connections from HVC to RA. Reward-based learning of mappings between conductor sequences and downstream neural networks may also be important for different kinds of precisely timed motor activity [181, 182] and for sequential memory [183, 184].

For WP and NP to be potential mechanisms for learning in the brain, they need to have possible biologically plausible implementations. In the case of NP, this necessitates that synapses can keep track of their input, which seems unproblematic, and of perturbations at the soma of the postsynaptic neuron, where the spatial input integration happens. Previous work proposed that the latter may happen by subtracting a short term temporal average of the past from the present overall input [58, 59]. Assuming the perturbations fluctuate more quickly than the other input, this difference approximates the perturbations and can be used to replace them in the eligibility trace. In the case of WP, the biological implementation may be even simpler. It requires random weight changes that can be tracked, that are approximately constant during a task trial and that can be enhanced, deleted or reversed by a subsequent reward signal. Spontaneous synaptic plasticity, which is observed in experiments on timescales ranging from minutes to days (see Section 2.1.3, e.g. ref. [24]), could produce the weight perturbations. As suggested by previous work [83], synaptic unreliability may also provide such perturbations. If neurons only spike during a short part of the trial, corresponding to a single time bin in our analysis, a failure of synaptic transmission translates to a weight perturbation that is constant during a trial. Further, fluctuations of activity-dependent plasticity may provide weight

perturbations, while the deterministic baseline acts as a useful prior. In addition, experiments have shown the existence of reward-modulated weight changes [154, 185] and of modulations of past weight changes [154, 186].

Previous work proposed WP [54–56, 83, 151, 167, 172, 187] in many variants, which differ in, for example, the temporal extension of the task, the perturbation scheme, where all weights or only one weight are perturbed in a trial, or the form of the weight update. A similar diversity of variants exists for NP [56, 58–60, 62, 173, 188]. In this chapter, we use variants of WP that are similar to the ones studied in refs. [55, 56] and variants of NP similar to the ones studied in refs. [56, 57]. Specifically, all weights are perturbed, the reward of the perturbed network is compared with the reward of an unperturbed network or an estimate of it, and the weight update is proportional to the measured success in order to ensure that it occurs on average parallel to the reward gradient. Refs. [55, 57] considered temporally extended tasks. Ref. [56] considered tasks without temporal extension and derived analytical expressions for the error dynamics. In this chapter, we derive analytical expressions for the error dynamics in temporally extended tasks and take the dimensionality of the input into account. Our results show that the long-standing belief that NP is to be preferred over WP is often wrong. Importantly, our numerical experiments show that this also holds for standard networks and tasks used in biology and machine learning.

NP is studied in various concrete neurobiological settings. Previous work used feedforward networks with NP to model the learning of coordinate transforms in the visual system [189], birdsong [61, 180] and motor output [58, 190]. Ref. [59] shows that reservoir computers with NP trained, fed back readouts can learn periodic inputs, routing and working memory tasks. Ref. [60] uses a fully plastic recurrent network for the learning of a delayed non-match-to-sample, a selective integration and a motor control task. Finally, NP is employed for reference and comparison [113, 191–196]. WP is considered less in studies of neurobiological learning. It is implemented in early feedforward network models for birdsong [197] and binary output task learning [83, 187]. Further, it is occasionally used for comparison [192, 194]. The results of this chapter suggest that for many neurobiological tasks WP is at least as suitable as NP, while the neurobiological implementation may be even simpler. They further suggest that WP might often be the better choice for reference and comparison.

4.A Appendix

4.A.1 Dependence of weight update variance on error baseline

Using the error E of the unperturbed trial as baseline in the weight updates (Eqs. (4.3) and (4.6)) minimizes the variance $\langle\langle \Delta w_{ij}^{\mathrm{WP}} \rangle\rangle$ of the weight update. To show this for WP, we compute the variance using E plus a potentially trial- and connection-dependent term $\tilde{E}_{ij}$ as the error baseline. In linear approximation, one then has

$$\begin{aligned}\langle\langle \Delta w_{ij}^{\mathrm{WP}} \rangle\rangle &\approx \frac{\eta^2}{\sigma_{\mathrm{WP}}^4} \left\langle \left[\left(\sum_{m=1}^{M} \sum_{l=1}^{N} \frac{\partial E}{\partial w_{ml}} \xi_{ml}^{\mathrm{WP}} - \tilde{E}_{ij} \right) \xi_{ij}^{\mathrm{WP}} \right]^2 \right\rangle - \eta^2 \left(\frac{\partial E}{\partial w_{ij}} \right)^2 \\ &= \eta^2 \sum_{m=1}^{M} \sum_{l=1}^{N} \left(\frac{\partial E}{\partial w_{ml}} \right)^2 + \eta^2 \left(\frac{\partial E}{\partial w_{ij}} \right)^2 + \frac{\eta^2}{\sigma_{\mathrm{WP}}^2} \tilde{E}_{ij}^2 . \end{aligned} \tag{4.75}$$

Thus, the variance is minimal if $\tilde{E}_{ij} = 0$ for all i and j.

In the case of NP, we add a potentially trial- and neuron-dependent term $\tilde{E}_i$ to the usual error baseline E. In linear approximation, one then has

$$\begin{aligned}\langle\langle \Delta w_{ij}^{\mathrm{NP}} \rangle\rangle &\approx \frac{\eta^2}{\sigma_{\mathrm{NP}}^4} \left\langle \left[\left(\sum_{m=1}^{M} \sum_{s=1}^{T} \frac{\partial E}{\partial y_{ms}} \xi_{ms}^{\mathrm{NP}} - \tilde{E}_i \right) \sum_{t=1}^{T} \xi_{it}^{\mathrm{NP}} r_{jt} \right]^2 \right\rangle - \eta^2 \left(\frac{\partial E}{\partial w_{ij}} \right)^2 \\ &= \eta^2 \sum_{m=1}^{M} \sum_{s=1}^{T} \left(\frac{\partial E}{\partial y_{ms}} \right)^2 \sum_{t=1}^{T} r_{jt}^2 + \eta^2 \left(\frac{\partial E}{\partial w_{ij}} \right)^2 + \frac{\eta^2}{\sigma_{\mathrm{NP}}^2} \tilde{E}_i^2 \sum_t r_{jt}^2 . \end{aligned} \tag{4.76}$$

Again, the variance is minimal if $\tilde{E}_i = 0$ for all i.

4.A.2 Numerical results for MNIST

Algorithm	N_{batch}	η	Test loss	Test accuracy
WP	1	1.00×10^{-4}	0.988(32)	0.696(14)
	10	2.15×10^{-4}	0.630(7)	0.840(5)
	100	1.47×10^{-3}	0.401(5)	0.884(2)
	1000	4.64×10^{-3}	0.331(6)	0.907(3)
NP	1	1.47×10^{-3}	0.407(2)	0.885(3)
	10	2.15×10^{-3}	0.396(11)	0.886(4)
	100	2.15×10^{-3}	0.399(13)	0.887(5)
	1000	2.15×10^{-3}	0.395(9)	0.887(6)
SGD	1	5.62×10^{-3}	0.320(3)	0.910(2)
	10	3.16×10^{-2}	0.289(8)	0.920(3)
	100	1.00×10^{-1}	0.274(7)	0.923(3)
	1000	1.00×10^{-1}	0.281(8)	0.923(3)

Table 4.3: Performance of single-layer networks ($L = 1$) for SGD, WP and NP on a held-out test set, after training, for the MNIST task. Third column shows the best learning rate obtained from the grid search. Values in the last two columns are the mean loss and the accuracy after 50 000 weight updates, averaged over five instances (standard deviation in brackets).

Algorithm	N_{batch}	η	Test loss	Test accuracy
WP	1	6.80×10^{-5}	1.158(55)	0.690(20)
	10	2.15×10^{-4}	0.613(15)	0.839(9)
	100	6.81×10^{-4}	0.390(8)	0.890(3)
	1000	3.16×10^{-3}	0.270(7)	0.923(2)
NP	1	6.81×10^{-4}	0.515(26)	0.856(7)
	10	4.64×10^{-4}	0.541(19)	0.860(11)
	100	6.81×10^{-4}	0.510(36)	0.860(14)
	1000	4.64×10^{-4}	0.545(25)	0.859(5)
SGD	1	1.00×10^{-2}	0.165(7)	0.952(3)
	10	5.62×10^{-2}	0.083(6)	0.976(1)
	100	5.62×10^{-1}	0.098(7)	0.977(1)
	1000	5.62×10^{-2}	0.079(8)	0.977(2)

Table 4.4: Performance of two-layer networks ($L = 2$) for SGD, WP and NP on a held-out test set, after training, for the MNIST task. Third column shows the best learning rate obtained from the grid search. Values in the last two columns are the mean loss and the accuracy after 50 000 weight updates, averaged over five instances (standard deviation in brackets).

Algorithm	N_{batch}	η	Test loss	Test accuracy
WP	1	6.80×10^{-5}	1.383(47)	0.646(20)
	10	2.15×10^{-4}	0.640(21)	0.829(3)
	100	6.81×10^{-4}	0.389(9)	0.887(3)
	1000	2.15×10^{-3}	0.258(12)	0.925(4)
NP	1	3.16×10^{-4}	0.775(40)	0.784(19)
	10	3.16×10^{-4}	0.795(54)	0.786(18)
	100	3.16×10^{-4}	0.802(40)	0.791(35)
	1000	2.15×10^{-4}	0.844(34)	0.778(9)
SGD	1	1.00×10^{-2}	0.162(16)	0.951(4)
	10	5.62×10^{-2}	0.092(14)	0.975(3)
	100	1.78×10^{-1}	0.107(11)	0.978(2)
	1000	5.62×10^{-2}	0.095(4)	0.977(1)

Table 4.5: Performance of three-layer networks ($L = 3$) for SGD, WP and NP on a held-out test set, after training, for the MNIST task. Third column shows the best learning rate obtained from the grid search. Values in the last two columns are the mean loss and the accuracy after 50 000 weight updates, averaged over five instances (standard deviation in brackets).

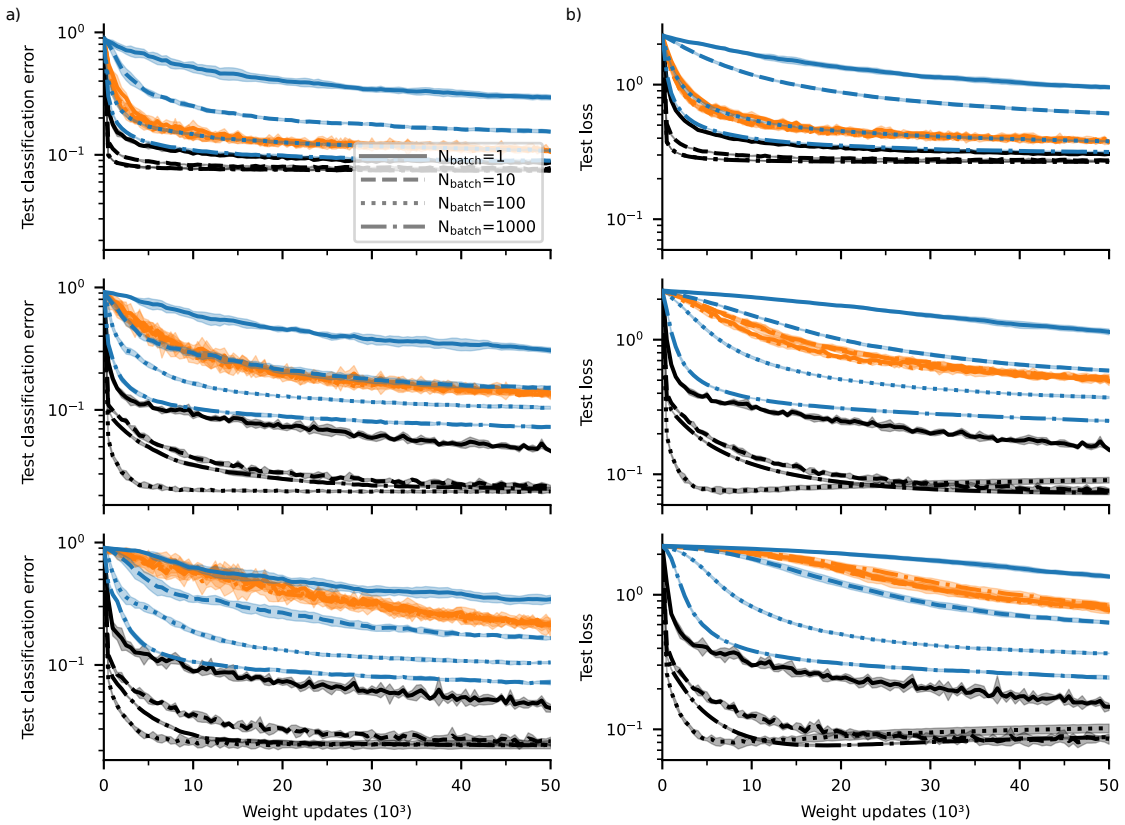

Figure 4.9: WP's performance on MNIST increases with batch size, while NP's does not.
(a) Test classification error (one minus test accuracy) as a function of the number of weight updates for WP (blue), NP (orange) and SGD (black) for different batch sizes N_{batch}. Top: single-layer network, middle: two-layer network, bottom: three-layer network. Solid lines show the mean and shaded areas show the standard deviation using 5 network instances.
(b) Same as (a) but for the test loss.

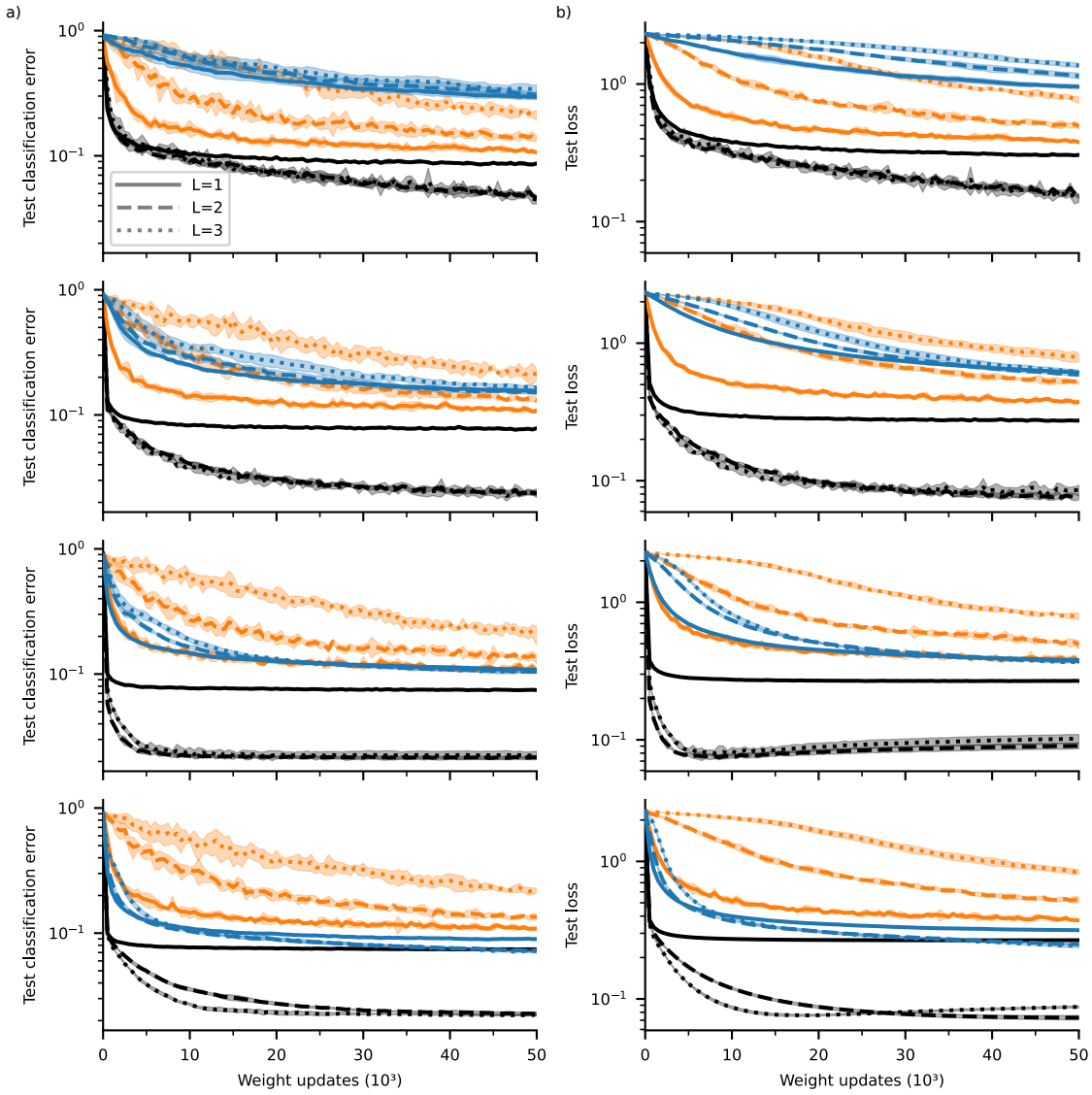

Figure 4.10: NP's performance on MNIST is harmed by increasing the number of network layers, while WP's performance is relatively constant.
(a) Test classification error (one minus test accuracy) as a function of the number of weight updates for WP (blue), NP (orange) and SGD (black) for different numbers of network layers L. Rows show results for different batch sizes ($N_{\text{batch}} = 1, 10, 100, 100$ from top to bottom). Solid lines show the mean and shaded areas show the standard deviation using 5 network instances.
(b) Same as (a) but for the test loss.

CHAPTER 5

Drifting assemblies for persistent memory

Parts of this chapter are included in the following published article:

[3] Y. F. Kalle Kossio, S. Goedeke*, **C. Klos*** and R.-M. Memmesheimer
(* equal contribution)
Drifting assemblies for persistent memory: Neuron transitions and unsupervised compensation
PNAS **118** (2021) e2023832118

The following sections contain the parts of this article that include significant contributions from me. I modified them substantially compared to the article. My contributions were the development and simulation of the LIF networks where noisy autonomous activity drives the drift, based on the LIF network where spontaneous synaptic turnover drives the drift (see article). Further, I performed most of the analysis, which was conceptualized by all authors, of the simulation results for these networks. The random walk model based on statistics of weight changes was also conceptualized by all authors; I contributed large parts of the derivation and semi-analytical computation of the stationary probability densities and performed the Markov simulations.

5.1 Introduction

In the previous chapters, we considered weight changes that directly allow to learn a task (Chapter 4) or have the goal of pretraining a network to allow it to dynamically learn a task (Chapter 3). For this, we partly used WP learning, which leads to a diffusion of weights that are irrelevant for the current task. However, these seemingly irrelevant weights may be relevant for other tasks or memories. Similarly, many of the other plasticity mechanisms, such as the basic variants of STDP, continuously affect synapses. Furthermore, experimental studies have shown that neural representations of memories change spontaneously [63–65]. These observations raise the question of how memories can persist despite ongoing representational as well as weight and connectivity changes. Previous work suggested that memories can be maintained because the changes are rather weak, because they only happen in a subspace of the full weight space that does not affect behavior or because they can be compensated with the help of external supervision [64, 65, 90, 198, 199].

In this chapter, we propose a novel memory model that explains the representational drift based on

weight and connectivity changes and preserves memories. It does not assume that the changes are weak, that they only happen in the irrelevant part of the weight space or that an external supervisory signal is available. Specifically, we consider the standard memory model based on assemblies (see Section 2.1.5). So far, these assemblies have been assumed to consist of the same neurons for faithful memory storage [67]. In our model, neurons constantly transition between assemblies, i.e. the assemblies drift (Fig. 5.1). The transitions are induced by noisy autonomous (without receiving external stimulation or feedback) network activity. In our article, we also show that the transitions can be induced by spontaneous (activity-independent) synaptic changes [3]. As this happens only gradually, in- and output neurons can learn about the changed assembly compositions in an unsupervised fashion, thus keeping memories intact. We refer to the neurons that make up the assemblies as interior neurons and to the in- and output neurons as periphery neurons. To illustrate our scheme, we simulate a recurrent network of LIF neurons consisting of a few assemblies. We carefully track the weight dynamics to show the assembly drift. Based on the statistics of weight changes, we also construct a simplified random walk model of the neuron transitions. This allows us to understand the transitions in greater detail.

This chapter is structured as follows. In Section 5.2, we introduce our network model. In Section 5.3, we present our simulation results, analyze them and construct the effective model for the neuron transitions. Finally, in Section 5.4, we discuss our results.

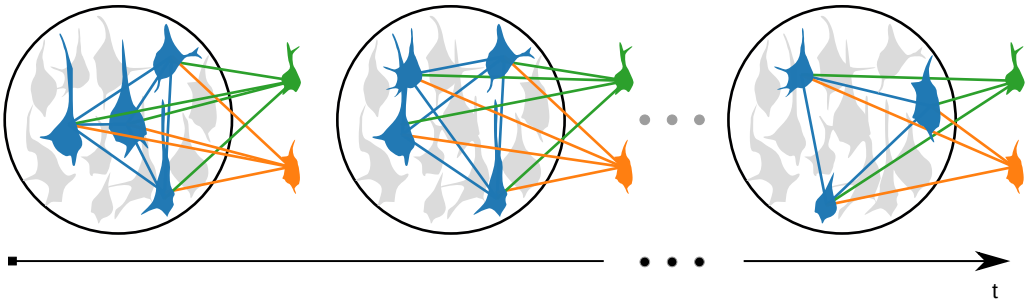

Figure 5.1: Schematics of drifting assemblies.
Snapshots of a network at three different points in time. (Left) The assembly consists of an ensemble of strongly-connected neurons (blue), which form the neural representation of a memory. Input (green) and output (orange) neurons are connected to it. For example, the memory may be that of a soccer ball. Seeing it activates the input neurons, which triggers the assembly. In turn, the assembly activates the output neurons, which initiate the kicking of the ball. (Middle) With time, the ensemble of neurons that constitute the assembly changes. Since this happens only gradually, the periphery neurons can learn about the changed assembly compositions. (Right) After a long enough time, the assembly consists of a completely different ensemble of neurons, which is only indirectly related to the neural ensemble from the beginning via the ensembles forming the assembly at the time in between.

5.2 Model

5.2.1 Networks

We consider networks of LIF neurons that consist of excitatory and inhibitory neurons. The excitation and inhibition approximately balance each other (see Section 2.2.2). The networks consist of on the

order of a hundred neurons and store between two and four assemblies in the plastic connections between excitatory neurons. The remaining connections are homogeneous and static.

The networks consist of N_E excitatory and N_I inhibitory neurons. N_{int} of the excitatory neurons are interior neurons. For most of our simulations, we initially group the interior neurons into n_{asbly} assemblies consisting of $N_{\text{asbly}}(0)$ neurons each. Further, we assign either no or $N_{\text{peri}} = 4$ periphery neurons to each assembly. If an assembly has periphery neurons, two of them are designated input, and the other two are designated output neurons. All neurons are current-based LIF neurons (see Section 2.2.1). The membrane potential $V_i(t)$ of neuron i obeys

$$\tau_m \frac{dV_i(t)}{dt} = V_{\text{rest}} - V_i(t) + RI_i^E(t) + RI_i^I(t) + \sqrt{2\tau_m}\sigma\xi_i(t), \tag{5.1}$$

$$RI_i^E(t) = \sum_{j\in M_E} w_{ij} \sum_{t_j \le t} e^{-\frac{t-t_j}{\tau_E}},\ RI_i^I(t) = \sum_{j\in M_I} w_{ij} \sum_{t_j \le t} e^{-\frac{t-t_j}{\tau_I}}. \tag{5.2}$$

Here, τ_m is the membrane time constant, V_{rest} the resting membrane potential, R the input resistance, $I_i^E(t)$ the total input current generated by the excitatory neurons, $I_i^I(t)$ the total input current generated by the inhibitory neurons and $\xi_i(t)$ is standard Gaussian white noise. σ^2 is equal to the variance of the stationary distribution of the membrane potential in absence of a threshold and input from the other neurons, in which case the membrane potential follows an Ornstein-Uhlenbeck process. The resting potential is $V_{\text{rest}} = 10\,\text{mV}$, the spike threshold is $V_\theta = 20\,\text{mV}$ and the reset potential is $V_0 = 0\,\text{mV}$, at which the membrane potential is held constant during the $\tau_{\text{ref}} = 5\,\text{ms}$ long absolute refractory period. Further, τ_E is the time constant of excitatory and τ_I the time constant of inhibitory synapses. t_j are the spike times of neuron j, M_E is the set of all N_E excitatory and M_I that of all N_I inhibitory neurons.

Except for self-connections and connections between periphery neurons, all connections are present. The periphery neurons are not connected, because they might lie in distinct brain areas with little interconnectivity. Previous theoretical work has shown that a combination of STDP and homeostatic plasticity (see Section 2.2.1) can enable the learning [200, 201] and spontaneous emergence [202, 203] of static assemblies. We also use these plasticity mechanisms for the connections between excitatory neurons. Specifically, we use pair-based STDP with all-to-all spike interaction and a symmetric STDP window function (Fig. 5.2). Such symmetric STDP can be found in CA3 [84] (Fig. 2.2), which is assumed to serve as an associative memory network and to store assemblies [66]. We use weight normalization for both in- and output weights. After each spike of an excitatory neuron in the network, we divisively normalize both the columns and rows of the weight matrix to a total weight of $\sum_{j\in M_E} w_{ij} = \sum_{j\in M_E} w_{ji} = w_{\text{sum}}$ for interior and to a total weight of $w_{\text{sum,peri}}$ for periphery neurons. There is substantial experimental evidence for input normalization [88] and indications that also a neuron's output is normalized [89] (see Section 2.1.3). The weights of the connection between interior neurons are bounded by 0 mV and $w_{\max}$, for connections from or to periphery neurons the upper bound is $w_{\text{max,peri}}$. These bounds are enforced by clipping weights before and after weight normalization. See Section 5.A.1 for the numerical values of all model parameters.

5.2.2 Simulations

To simulate our networks, we use Python and the Brian simulator for spiking neural networks [204] (see [205] for example code). We initialize the networks by setting connection weights between neurons within an assembly and between an assembly and its periphery neurons to 1 mV and all

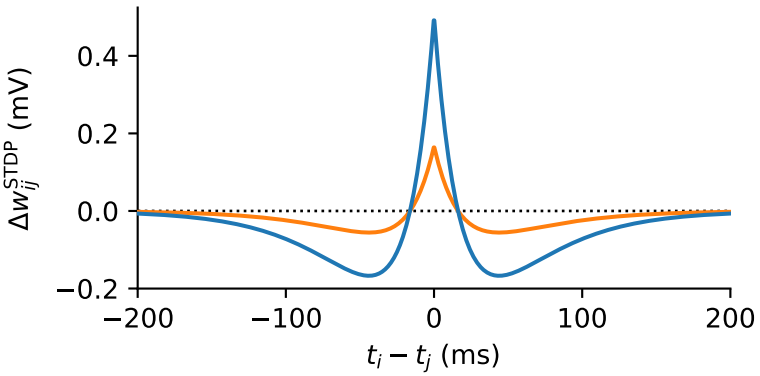

Figure 5.2: STDP rule.
STDP windows used in the network model with three assemblies (see Section 5.A.1) for synapses between interior neurons (blue) and for synapses between interior and periphery neurons (orange). Weight change is given in terms of the change of the peak EPSP that a presynaptic spike evokes in a resting neuron. Black dotted line indicates border between potentiation and depression.

others to 0 mV, then we apply weight normalization and clipping. We perform five alike simulations with long durations (Section 5.A.1) and different random realizations of the noise to check that the representational structure is conserved over time, i.e. that assemblies continuously drift and that their periphery neurons faithfully follow them. To detect the assemblies at later stages during the simulations, we cluster the weight matrices with the Louvain clustering algorithm [206] as implemented in ref. [207].

5.3 Results

5.3.1 Drifting memory representations

Our scheme posits that assemblies consist of different neurons at different points in time and that periphery neurons can keep track of this representational drift (Fig. 5.3a). To demonstrate it, we simulate networks with three assemblies. Each of them is initialized with 30 interior neurons and connected to two in- and two output neurons. The assembly drift can be visualized by considering the weight matrix (between excitatory neurons) at different points in time (Fig. 5.3b). The indices of the neurons are ordered according to their initial assembly membership, i.e. the weights of the connections between, for example, the neurons of assembly 1 take up the upper left corner of the weight matrix. At the beginning of the simulation the three assemblies are clearly visible in the weight matrix (Fig. 5.3b left). Then, the neurons start to gradually switch assemblies. This results in strong connections with the assembly they transition to and weak connections with their original assembly (Fig. 5.3b middle). After a sufficiently long simulation of the network, no clear structure in the weight matrix is visible anymore (Fig. 5.3b right). However, reindexing the neurons using a clustering algorithm shows that the assemblies are preserved (Fig. 5.3c). It also shows that the periphery neurons stay connected to the assembly they were connected to in the beginning.

The network neurons exhibit asynchronous irregular background activity interspersed with occasional, synchronous spiking of all neurons that form an assembly (Fig. 5.3d). In the following,

we refer to the latter as assembly reactivations. Further, the membrane potentials of the neurons fluctuate irregularly. A partial excitation of the assembly neurons leads to the activation of the whole assembly and a strong stimulation of the input neurons leads to the activation of the output neurons (Section 5.A.3). Thus, the assemblies possess the associative memory property and are suitable for memory recall (see Section 2.1.5).

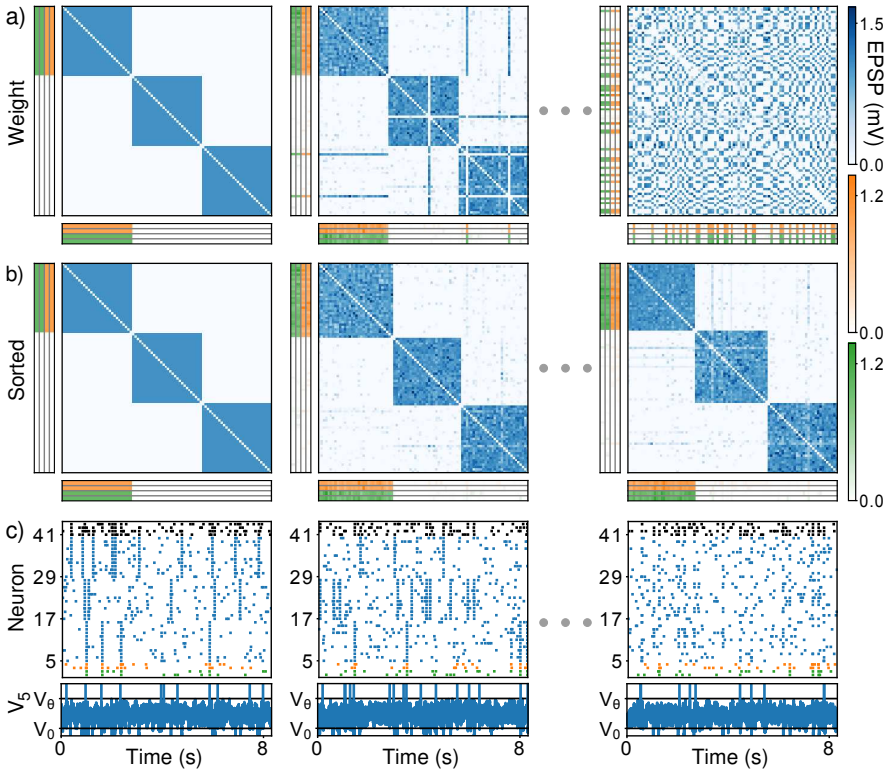

Figure 5.3: Drifting assemblies in a network model of LIF neurons.
(a) Weight matrices of the connections between interior neurons (blue), of the connections from input and output neurons to interior neurons (green and orange vertical) and of the connections from interior to input and output neurons (green and orange horizontal). For clarity, only the weights from and to the periphery neurons attached to assembly 1 are shown. The network is initialized with three assemblies (first column). After 27 min of simulated time, a few interior neurons have switched assemblies (second column) After 30 h, the weight matrix is completely remodeled as the assemblies have drifted away (third column).
(b) Like (a), but the neurons are reindexed with the help of a clustering algorithm. This shows that the assemblies persist and that the periphery neurons follow their drift.
(c, top) Spike trains (colored ticks indicate the spike times) of the input (green) and output (orange) neurons of assembly 1 and the first twelve interior neurons of each assembly (blue) sorted according to their initial assembly membership. Further, the spike trains of four inhibitory neurons are shown (black). (c, bottom) Membrane potential and spikes (vertical lines) of the first interior neuron.

5.3.2 Analysis of drifting assemblies

To characterize the assembly drift, we take a closer look at the weight dynamics in the network. For single interior neurons, they are characterized by relatively long times of membership to the same

assembly and quick transitions between them (Fig. 5.4a). Periphery neurons do not transition between assemblies (Fig. 5.4b).

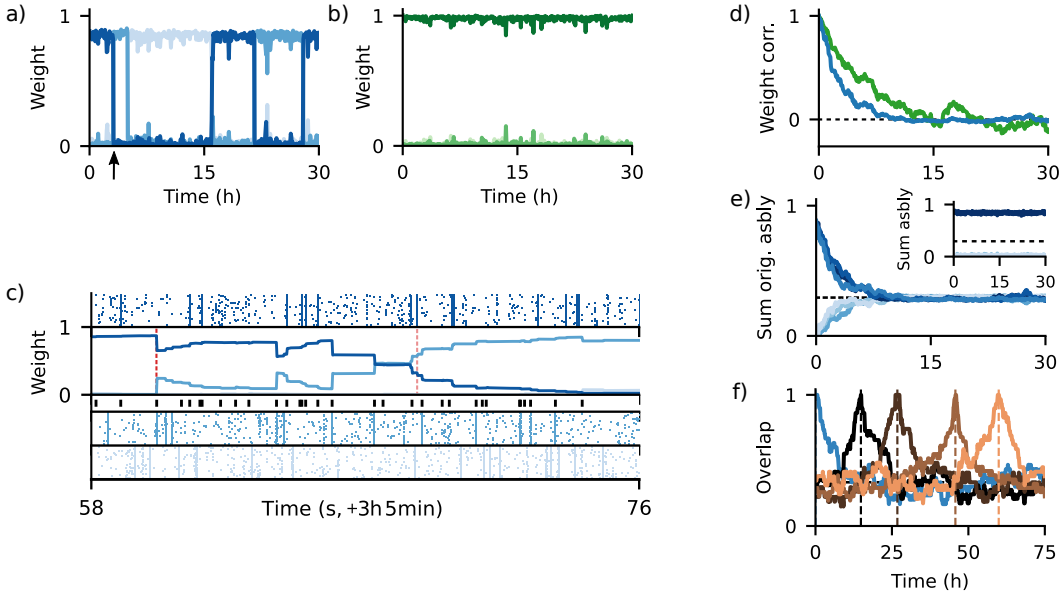

Figure 5.4: Analysis of drifting assemblies.
(a) Adhesion to and quick transitions between assemblies of interior neurons. Curves show normalized sums of weights between an interior neuron and assemblies 1, 2 and 3 (dark to light blue).
(b) Constant attachment of periphery neurons to the same assembly. Curves show normalized sums of weights between a periphery neuron and assemblies 1, 2 and 3 (dark to light green).
(c) Close-up view of the switching event marked with an arrow in (a). Raster plots show spikes of assembly 1 (first subpanel), assembly 2 (fourth subpanel) and assembly 3 (fifth subpanel). Second subpanel shows the normalized sums of weights and third subpanel the spikes of the neuron that switches. It initially belongs to assembly 1, but transitions to assembly 2 primarily due to coincident spiking with reactivations of assembly 2 (red dashed) and failures to spike together with reactivations of assembly 1 (light red).
(d) Complete remodeling of connection weights. Pearson correlation between initial and later weights between interior neurons (blue) and between interior and periphery neurons (green) decay to chance level (black dashed).
(e) Complete remodeling of assemblies. Normalized summed weights within initial assemblies 1, 2 and 3 (dark blues) decay to chance level (black dashed). Normalized summed weights between initial assemblies 1 and 2, 1 and 3, and 2 and 3 (light blues) increase to chance level. Inset: Normalized summed weights within current assembly 1 (dark blues) and between current assemblies 1 and 2 and 1 and 3 (light blues) stay approximately constant, indicating that the assemblies persist.
(f) Continuously ongoing assembly drift. Overlap of the neuron ensemble forming assembly 1 at reference times (dashed verticals) with the ensembles at past and future times repeatedly falls to chance level (black, mostly covered). Reference times are selected based on when the assembly has completely remodeled with respect to the previous reference time (brown curves), starting with the initial assembly (blue).
See Section 5.A.2 for more details.

To consider in detail how an interior neuron transitions between assemblies, we track its weight dynamics during a switching event (Fig. 5.4c). The switching is induced by several mechanisms: First, if a neuron spikes by chance near-synchronously with a reactivation event of an assembly it does not belong to, STDP strengthens the connection weights between the neuron and this assembly. In addition, the weight normalization leads to a weakening of the connection weights between the

neuron and its original assembly. The strengthened connections to the assembly that reactivated increases the tendency of the neuron to spike together with the neurons of this assembly, which further strengthens the connections. Second, if a neuron does not spike during a reactivation event of the assembly it belongs to, STDP strengthens the connections between the reactivated neurons. Weight normalization then weakens the connections between the neuron that did not spike and the reactivated neurons and thereby strengthens its connections to the other assemblies. Third, if the spike timing of a neuron relative to a reactivation of the assembly it belongs to falls into the depression-dominated regime of STDP (Fig. 5.2), the connection weights between the neuron and the assembly are weakened. Fourth, the asynchronous, irregular background spiking induces small weight fluctuations. The first mechanism is the most important for neuron switches as it leads to the largest weight changes. Because STDP is weaker for the connections involving periphery neurons, they do not switch assemblies. The continued switching of neurons leads to a complete remodeling of the connection weights (Fig. 5.4d) and assemblies (Fig. 5.4e). Further, the assembly drift goes on continuously (Fig. 5.4f).

The neuron transitions are rather rare events as usually a neuron spikes together with the other neurons of its assembly when it reactivates. STDP then leads to the strengthening of intra-assembly connection weights. As our STDP rule is depression-dominated, the inter-assembly connection weights are on average weakened due to the uncorrelated activity of neurons belonging to different assemblies. These are similar mechanisms to the ones that kept static assemblies intact in previous models [84, 200–202, 208, 209]. Furthermore, the synaptic competition induced by the weight normalization weakens inter-assembly connections, because they get potentiated less by STDP [126, 210, 211].

The neuron transitions lead to moderately fluctuating assembly sizes (Fig. 5.5a), but no assembly vanishes. An important factor for this is likely the size dependence of the reactivation rate of the assemblies. Reactivations tend to strengthen the intra-assembly weights and increase the chance of recruitment of neurons from other assemblies. Assemblies that have less than the average number of neurons have a higher reactivation rate compared to large assemblies (Fig. 5.5b) since the average connection weight in small assemblies is larger. Thus, fewer neurons of a small assembly need to spike coincidentally to initiate its reactivation. The average connection weight is larger in small assemblies because there are fewer intra-assembly connections, which take up most of the total weight w_{sum} that is available for the in- or output connections of each neuron in the assembly.

Finally, we note that assemblies can also emerge spontaneously for random initial weight matrices (Section 5.A.5). The same was found for static assemblies [202, 203, 211].

5.3.3 Simplified model of neuron switching and assembly drift

To get a better understanding of the mechanisms that underlie the neuron switches, we collect summary statistics of the weight dynamics. For this, we consider a network with two assemblies and without periphery neurons for simplicity (see Section 5.A.6 for the same analysis in a network with three assemblies). In a network simulation, we then track the change Δw_1 of the normalized summed input weight w_1 from assembly 1 to a neuron, Fig. 5.6a. Focusing on the inputs is justified as the output of a single neuron has only a small influence on assembly membership. We measure the change Δw_1 that occur between regular time points separated by the typical inter-spike-interval as a function of the weight w_1. Afterwards, we compute the average $\overline{\Delta w_1}(w_1)$ and standard deviation $\text{Std}(\Delta w_1)(w_1)$ of all recorded weight changes (see Section 5.A.2 for details). The results are shown in Fig. 5.6a-d. If w_1 is close to 1, the neuron is part of assembly 1 and is on average drawn closer to 1 ($\overline{\Delta w_1}(w_1) > 0$, Fig. 5.6b). If w_1 is close to 0, it is part of assembly 2 and on average drawn to 0 ($\overline{\Delta w_1}(w_1) < 0$). To

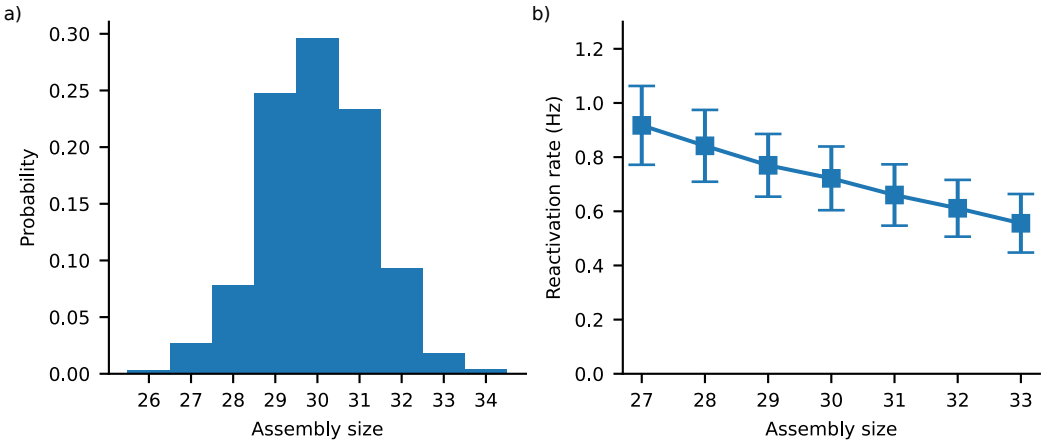

Figure 5.5: Assembly size distribution and reactivation rate.
(a) Distribution of assembly sizes, which we record every 270 s during a 75 h long simulation.
(b) Reactivation rate of the assemblies as a function of assembly size. Smaller assemblies tend to reactivate more often. A reactivation is defined as an event where the number of spikes emitted by assembly neurons within 15 ms exceeds 50 % of the assembly size. We measure the reactivation rates during 60 s long intervals starting every 270 s. Squares show the mean and error bars the standard deviation over the measuring intervals for assembly sizes that are observed at least 50 times.

further visualize the dynamics, we think of $\overline{\Delta w_1}(w_1)$ as being evoked by a force $F(w_1) = \overline{\Delta w_1}(w_1)$, similar to classical mechanics system where friction dominates over inertia. Thus, $U(w_1)$ determines the force by $F(w_1) = -dU(w_1)/dw_1$. The potential has two wells near $w_1 \approx 1$ or $w_1 \approx 0$, which correspond to the two assemblies. Hence, the neuron switches between the assemblies rely on the noise of the weight changes ($\mathrm{Std}(\Delta w_1)(w_1)$, Fig. 5.6d). The assembly drift thus appears to be a result of noise-induced transitions between meta-stable state [68].

To verify this interpretation of the neuron switching and to selectively study the impact of the noise or the average of the weight updates on the neuron transitions, we construct a simplified random walk model. It is based on the observed statistics of the weight changes and makes use of the Markov assumption, i.e. we assume that the change in w_1 depends only on its previous value. This can be justified by the sufficient length of the sampling intervals. We further assume for simplicity that the weight change fluctuations are normally distributed. Thus, the simplified weight dynamics are given by

$$w_1(t+1) = w_1(t) + \overline{\Delta w_1}(w_1(t)) + \mathrm{Std}(\Delta w_1)(w_1(t))\xi(t), \tag{5.3}$$

where the $\xi(t)$s are normally distributed random variables with zero mean and unit variance, and $w_1(t)$ is clipped to the interval $[0, 1]$ after each step. We compare the random walk model with the full model by considering their stationary probability densities. It can be obtained from the random walk model via simulations but also analytically by using a diffusion approximation. For the latter, we interpret Eq. (5.3) as the Euler-Maruyama discretization with timestep equal to one of the drift-diffusion process $\frac{dw_1}{dt}(t) = \overline{\Delta w_1}(w_1(t)) + \mathrm{Std}(\Delta w_1)(w_1(t))\xi(t)$ with drift $\overline{\Delta w_1}(w_1)$ and diffusion coefficient $\mathrm{Std}(\Delta w_1)^2(w_1)/2$ [68]. The stationary solution of the corresponding Fokker-Planck equation is thus

given by

$$p_{\mathrm{FP}}(w_1) = \frac{\mathcal{N}}{\mathrm{Std}(\Delta w_1)^2(w_1)} \exp\left[2 \int_0^{w_1} du \, \frac{\overline{\Delta w_1}(u)}{\mathrm{Std}(\Delta w_1)^2(u)}\right], \tag{5.4}$$

where $\mathcal{N}$ is a normalization constant and where we assume reflecting boundary conditions.

We find acceptable agreement between the stationary probability densities of the full model and the random walk model (Fig. 5.6e), with the remaining deviations likely resulting from the non-Gaussianity of the weight change distributions in the full model. Further, the analytical approximation of the stationary probability density of the random walk model agrees well with the results of the simulations (Fig. 5.6e). The deviations at the boundaries originate from the clipping of w_1 in the Markov simulations. If we do not clip but reflect the overshooting of the increments at the boundaries, the densities also agree there. Thus, our interpretation of the neuron switching as a random walk is justified.

To examine the contribution of weight update fluctuations, we repeat the above analysis, but without including the effect of the mean in Eq. (5.3), $w_1(t+1) = w_1(t) + \mathrm{Std}(\Delta w_1)(w_1(t))\xi(t)$. We find that the noise alone is sufficient to generate the meta-stable states and the switching (Fig. 5.6f). Such noise-induced multistability can also be observed for, e.g., electrical and chemical oscillations, populations dynamics and foraging behavior [69–72]. Finally, to examine the contribution of the mean weight change, we replace the state-dependent noise strength $\mathrm{Std}(\Delta w_1)(w_1(t))$ with the state-independent average noise strength $\overline{\mathrm{Std}(\Delta w_1)(w_1)} = \int_0^1 \mathrm{Std}(\Delta w_1)(w_1) p_{\mathrm{full}}(w_1) dw_1$, where p_{full} is the stationary probability density of the full model. With this noise, the neuron does not leave the potential well it started in within the simulated periods (Fig. 5.6g). Thus, the mean weight change is sufficient to generate states that are almost stable. In conclusion, both drift and noise inhomogeneity contribute to the neuron transitions.

5.4 Discussion

In this chapter, we have considered how continuously ongoing weight changes can give rise to changing neural representations and how memories can persist despite such changing weights and representations. To this end, we considered assemblies of neurons that store associative memories. In our model, the fluctuations of the weight changes lead to fast transitions of neurons between assemblies. Thus, the assemblies drift. Nevertheless, the memories stay intact, because STDP and weight normalization allows the input and output neurons to follow their assembly. By considering the statistics of the weight changes and constructing an effective random walk model for the switches, we found that they can be understood as noise-induced transitions between meta-stable states. The meta-stable states arise from both the mean weight changes and the inhomogeneity of the weight change fluctuations.

Previous computational work that addressed how ongoing weight and connectivity fluctuations can be reconciled with persistent memory often focused on how neural representations can stay stable despite such fluctuations. Specifically, refs. [209, 212, 213] suggest that the fluctuations are fully compensated by unsupervised plasticity mechanisms and reactivations, refs. [199, 214, 215] suggest that they are compensated by retraining the network and refs. [64, 216–218] suggest that there is a preserved core of neurons that keeps memories stable.

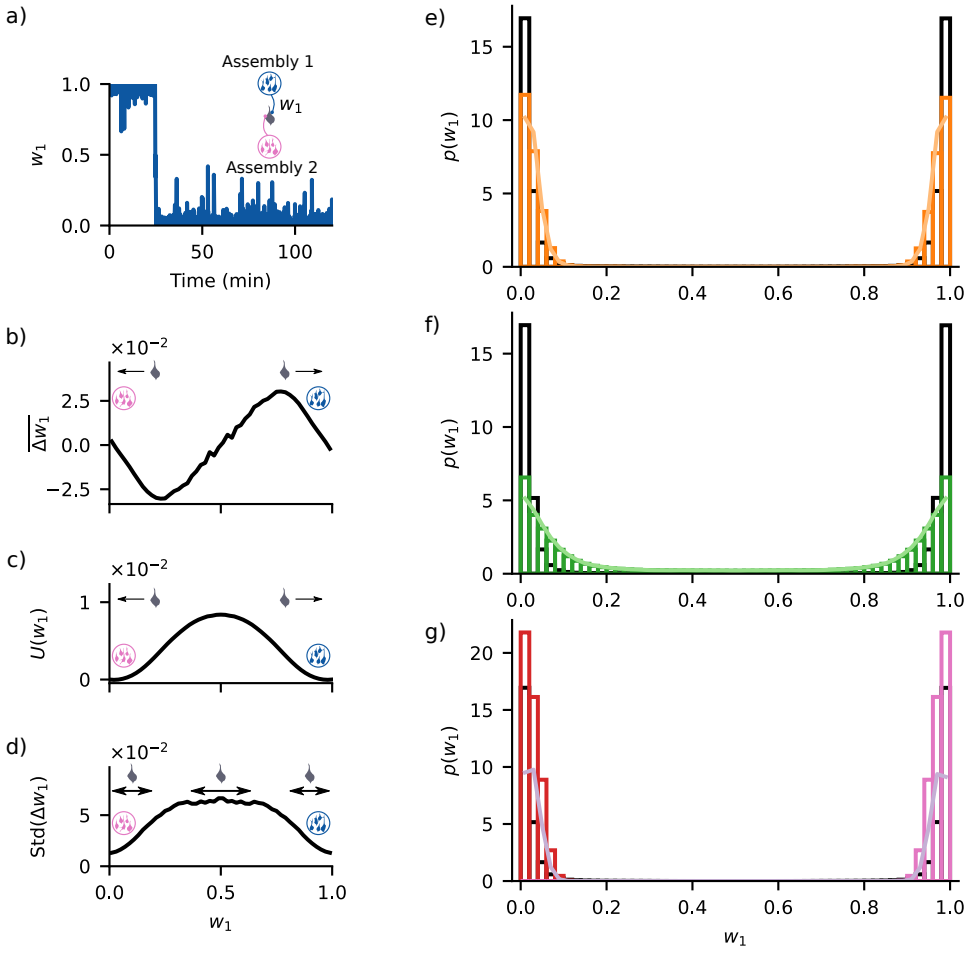

Figure 5.6: Weight dynamics and simplified model of neuron transitions between assemblies.
(a–d) Weight dynamics and summary statistics of it.
(a) Dynamics of the normalized summed input weight w_1 from assembly 1 to a neuron. Large (small) w_1 indicates membership to assembly 1 (assembly 2).
(b) Average weight change $\overline{\Delta w_1}$ as a function of w_1. It draws the neurons closer to either one of the assemblies.
(c) Potential $U(w_1)$ for $\overline{\Delta w_1}$. Its minima correspond to the membership to assembly 1 or 2.
(d) Standard deviation $\mathrm{Std}(\Delta w_1)$ of the weight change as a function of w_1. It induces neuron transitions between the assemblies.
See Section 5.A.2 for more details on panels (b–d).
(e–g) Stationary probability densities of w_1 as observed in a network simulation (black histograms), as observed in simulations of a simplified, Markovian random walk model of the neuron transitions (colored histograms) and as computed from an analytical diffusion approximation of the random walk model (colored curves).
(e) Results for the random walk model with drift and noise (orange). Both simulation and analytical results show acceptable agreement with the results from the network simulations.
(f) Results for the random walk model with noise only (green). The noise alone is sufficient to generate the meta-stable states and the switching.
(g) Results for the random walk model with drift and homogenized noise. The mean weight change is sufficient to generate states that are almost stable. In the Markov simulations, switching does not occur within the used simulation time. Thus, the distribution depends on the initial condition ($w_1(0) = 0$ (red) and $w_1(0) = 1$ (pink)). Despite the low probability, the analytical approximation accounts for the switching (purple).

In addition, a few modeling studies considered how weight and connectivity fluctuations give rise to changing neural representations. In the models of refs. [219, 220] assemblies change but also merge and break down, i.e. they are not suitable to store persistent memories. Ref. [199] shows that connectivity remodeling leads to fluctuations of the preferred direction of neurons in a model of the motor cortex. In the model of ref. [218], adding a fluctuating part to otherwise stable connection weights leads to partial changes of the neural representations of sequences.

Further, a few modeling studies, including some of the above, addressed how memories can persist despite changing neural representations. They assumed supervised retraining of outputs [221], that the neural representations change in a subspace that does not affect behavior [199] or the neural representations changes are only partial and thus allow for stable memory [218].

In contrast to these studies, in our networks the weight fluctuations lead to a complete remodeling of the weight matrix and neural representation, while the memories persist due to unsupervised plasticity mechanisms. Thus, there is no preserved core of network structure. The interior neurons encode different memories at different time points. Further, no external supervision is necessary to preserve the memories. STDP and weight normalization keeps the periphery neurons attached to their assembly. We suggest that drifting assemblies are the basis of associative memory and make the experimentally verifiable prediction that the remodeling of neural representations is complete, i.e. that the overlap of the realizations of assemblies at different time points decreases to chance level.

By themselves, drifting assemblies are functionally neutral. However, the drift may be helpful for the storage of memories as new memories may be easy to store in a highly plastic region of the brain (for example the hippocampus). Afterwards, they may drift away to less plastic regions (for example parts of the cortex). This would be in line with the two-stage model for memory [66]. Furthermore, static assemblies may be difficult to realize in the brain because of the noisy activity and ongoing synaptic changes. The latter could result from both activity-dependent or -independent plasticity. Since drifting assemblies still allow for persistent memories, there would be no evolutionary pressure to develop static assemblies.

In this thesis, we did not consider the learning of new memories, but it could be implemented by imprinting new assemblies with the help of strong input. This may result in the vanishing or merging of existing assemblies, which would lead to the forgetting or generalization of existing memories. Another possibility for the learning of novel memories in our model would be if specific inputs and outputs are assigned to preexisting assemblies. Such preexisting assemblies may form during development [222].

5.A Appendix

5.A.1 Parameters of models used for the simulations

Network model with three assemblies

Neuron numbers: excitatory neurons: $N_E = 102$; interior neurons: $N_{\text{int}} = 90$; periphery neurons: 12; inhibitory neurons: $N_I = 20$.

Network structure: all connections are present except for connections between periphery neurons and self-connections.

Neuron parameters: spike threshold: $V_\theta = 20\,\text{mV}$; reset potential: $V_0 = 0\,\text{mV}$; resting potential: $V_{\text{rest}} = 10\,\text{mV}$; membrane time constant: $\tau_m = 10\,\text{ms}$; absolute refractory period: $\tau_{\text{ref}} = 5\,\text{ms}$; sum of input and sum of output weights of an interior neuron: $w_{\text{sum}} = 256.25\,\text{mV} = \frac{1}{2}\left[\left(N_{\text{asbly}} - 1\right) w_{\max} + N_{\text{peri}} w_{\max,\text{peri}}\right]$, the angular bracketed term is the expected input of an interior neuron from a typical size assembly and its periphery neurons, if all weights were at their individual maximum; sum of input and sum of output weights of a periphery neuron: $w_{\text{sum,peri}} = 225.0\,\text{mV} = \frac{1}{5}\left[N_{\text{asbly}} w_{\max,\text{peri}}\right]$, the angular bracketed term is the expected input of a periphery neuron from a typical size assembly, if all weights are at their individual maximum; noise input strength: $\sigma = 3.5\,\text{mV}$.

Excitatory synapses: time constant: $\tau_E = 2\,\text{ms}$; maximal synaptic strength of synapses between interior neurons: $w_{\max} = 12.5\,\text{mV}$, evoking a peak EPSP of 1.67 mV in a resting postsynaptic neuron; maximal synaptic strength of synapses between interior and periphery neurons: $w_{\max,\text{peri}} = 37.5\,\text{mV}$, evoking a peak EPSP of 5.02 mV in a resting postsynaptic neuron; strength of synapses to inhibitory neurons: $w_{\text{E}\to\text{I}} = 5.02\,\text{mV}$, evoking a peak EPSP of 0.67 mV in a resting postsynaptic neuron.

Inhibitory synapses: time constant: $\tau_I = 5\,\text{ms}$; strength of synapses to excitatory neurons: $w_{\text{I}\to\text{E}} = -5.13\,\text{mV}$, evoking a peak inhibitory postsynaptic potential (IPSP) of $-1.28\,\text{mV}$ in a resting postsynaptic neuron; strength of synapses to inhibitory neurons: $w_{\text{I}\to\text{I}} = -5.39\,\text{mV}$ evoking a peak IPSP of $-1.35\,\text{mV}$ in a resting postsynaptic neuron.

STDP window: $\Delta w_{ij}(\Delta t) = \frac{\eta}{a - b(1+\delta)}\left[a \exp(-a\,|\Delta t|) - b(1+\delta)\exp(-b\,|\Delta t|)\right]$, where $\Delta t = t_i - t_j$ is the time difference between the postsynaptic and the presynaptic spike; window amplitude for connections between interior neurons: $\eta = 3.75\,\text{mV}$; window amplitude for connections between interior and periphery neurons: $\eta = 1.25\,\text{mV}$; LTP decay rate: $a = \frac{1}{\tau_{\text{LTP}}} = \frac{1}{20\,\text{ms}}$, where $\tau_{\text{LTP}} = 20\,\text{ms}$; LTD decay rate: $b = \frac{1}{\tau_{\text{LTD}}} = \frac{1}{40\,\text{ms}}$, where $\tau_{\text{LTD}} = 40\,\text{ms}$; ratio of integrated LTD and LTP: $1 + \delta = 1 + \frac{1}{3}$; in terms of the induced change of peak EPSP, peak LTP is 0.5 mV (0.17 mV), at 0 ms, peak LTD is $-0.17\,\text{mV}$ ($-0.06\,\text{mV}$), at $\pm 44\,\text{ms}$, for connections between interior neurons (connections from or to periphery neurons).

Memory representation: number of assemblies: $n_{\text{asbly}} = 3$; initial number of interior neurons per assembly: $N_{\text{asbly}}(0) = 30$; periphery neurons per assembly: $N_{\text{peri}} = 4$.

Simulation: time step: 0.25 ms; total simulated time: 75 hours.

Network model with two assemblies and without periphery neurons

Same parameters as before with the following exceptions:

Neuron numbers: excitatory neurons: $N_E = 68$; interior neurons: $N_{\text{int}} = 68$; periphery neurons: 0; inhibitory neurons: $N_I = 13$.

Neuron parameters: sum of input and sum of output weights of an interior neuron: $w_{\text{sum}} = 309.375\,\text{mV} = \frac{3}{4}\left[\left(N_{\text{asbly}} - 1\right) w_{\max}\right]$.

Excitatory synapses: strength of synapses to inhibitory neurons: $w_{\text{E}\to\text{I}} = 9.10\,\text{mV}$, evoking a peak EPSP of 1.22 mV in a resting postsynaptic neuron.

Inhibitory synapses: strength of synapses to excitatory neurons: $w_{\text{I}\to\text{E}} = -9.52\,\text{mV}$, evoking a peak inhibitory postsynaptic potential (IPSP) of −2.38 mV in a resting postsynaptic neuron; strength of synapses to inhibitory neurons: $w_{\text{I}\to\text{I}} = -10.31\,\text{mV}$ evoking a peak IPSP of −2.58 mV in a resting postsynaptic neuron.

STDP window: window amplitude: $\eta = 5\,\text{mV}$; in terms of the induced change of peak EPSP, peak LTP is 0.67 mV at 0 ms, peak LTD is −0.22 mV at ±44 ms,.

Memory representation: number of assemblies: $n_{\text{asbly}} = 2$; initial number of interior neurons per assembly: $N_{\text{asbly}}(0) = 34$.

Simulation: total simulated time: 50 hours.

Network model with three assemblies and without periphery neurons

Same parameters as for simulations with three assemblies and periphery neurons with the following exceptions:

Neuron numbers: excitatory neurons: $N_E = 102$; interior neurons: $N_{\text{int}} = 102$; periphery neurons: 0; inhibitory neurons: $N_I = 20$.

Neuron parameters: sum of input and sum of output weights of an interior neuron: $w_{\text{sum}} = 247.5\,\text{mV} = \frac{3}{5}\left[\left(N_{\text{asbly}} - 1\right) w_{\max}\right]$.

Excitatory synapses: strength of synapses to inhibitory neurons: $w_{\text{E}\to\text{I}} = 4.85\,\text{mV}$, evoking a peak EPSP of 0.65 mV in a resting postsynaptic neuron.

Inhibitory synapses: strength of synapses to excitatory neurons: $w_{\text{I}\to\text{E}} = -4.95\,\text{mV}$, evoking a peak inhibitory postsynaptic potential (IPSP) of −1.24 mV in a resting postsynaptic neuron; strength of synapses to inhibitory neurons: $w_{\text{I}\to\text{I}} = -5.21\,\text{mV}$ evoking a peak IPSP of −1.30 mV in a resting postsynaptic neuron.

STDP window: window amplitude: $\eta = 3.75\,\text{mV}$; in terms of the induced change of peak EPSP, peak LTP is 0.50 mV at 0 ms, peak LTD is −0.17 mV at ±44 ms,.

Memory representation: initial number of interior neurons per assembly: $N_{\text{asbly}}(0) = 34$.

Simulation: total simulated time: 50 hours.

5.A.2 Details on the network analysis

Analysis of drifting assemblies

Panel (a) of Fig. 5.4 depicts the normalized sum of the weights between the second interior neuron (index 6 in Fig. 5.3d) and the three assemblies in the networks. The normalization constant is $2w_{\text{sum}}$, which is equal to the total input plus the total output weight. Panel (b) depicts the normalized sum of the weights between the first input neuron and all three assemblies in the network. The normalization constant is $2w_{\text{sum,peri}}$. Panel (d) shows the Pearson correlation between weight matrices at times 0 and

t. Specifically, it is given by

$$\mathrm{Corr}(t) = \frac{\sum_{ij} \tilde{w}_{ij}(0)\tilde{w}_{ij}(t)}{\sqrt{\sum_{ij} \tilde{w}_{ij}(0)^2}\sqrt{\sum_{ij} \tilde{w}_{ij}(t)^2}}, \tag{5.5}$$

where $\tilde{w}_{ij} = w_{ij} - \bar{w}$ are the weights centered by their average $\bar{w}$. Panel (e) shows the normalized sums of the connection weights between the neurons that form the initial assemblies 1, 2 and 3 (dark blue colors). The normalization constant is the maximal sum $N_{\mathrm{asbly}}(0)w_{\mathrm{sum}}$. It also shows the normalized sums of the connection weights between the neurons that form the initial assemblies 1 and 2, 1 and 3, and 2 and 3 (light blue colors). The normalization constant is the maximal sum $2N_{\mathrm{asbly}}(0)w_{\mathrm{sum}}$. The inset in panel (e) shows these quantities (but only those involving assembly 1) for the assemblies at the current time. The normalization constants are adjusted according to the current assembly sizes. The chance level is the normalized sum of all weights between interior neurons. The normalization constant is $n_{\mathrm{asbly}}N_{\mathrm{int}}w_{\mathrm{sum}}$. Panel (f) depicts the overlap of the ensemble of neurons that form assembly 1 at the current time with the ensemble of neurons that form assembly 1 at different reference times (different colors). The overlap is computed as the number of neurons shared by the ensembles, divided by the number of neurons in the reference ensemble. We also use the overlap to define the complete remodeling of an assembly and the network: If the overlap with respect to a previous reference ensemble has decreased to chance level, we say that the assembly is completely remodeled with respect to the reference. If this happens for all assemblies with respect to their original realization, we say that the network is completely remodeled.

Statistics of weight changes

To collect the weight change statistics, we record for all interior neurons the change Δw_1 of their normalized summed input weight w_1 from assembly 1. The normalization constant is w_{sum}. The sampling time is 0.5 s, which is approximately equal to the average single neuron interspike interval. We divide the range of possible values of w_1, which lie between 0 and 1, into 50 equally-sized bins. Then, we calculate for each bin the average $\overline{\Delta w_1}$ and standard deviation $\mathrm{Std}(\Delta w_1)$ of the changes ensuing those w_1 that fall in it. We compute the potential $U(w_1)$ by integrating $-\overline{\Delta w_1}(w_1)$ over w_1.

5.A.3 Associative memory property and input-output functionality of assemblies

To test that the assemblies possess the associative memory property, we suppress the activity of the periphery neurons (Fig. 5.7a). The partial activation of an assembly by the background activity is still sufficient to elicit assembly reactivations, as is required for associative memory (see Section 2.1). Assembly reactivations occur more frequently when the activity of periphery neurons is not suppressed (Fig. 5.7b), because the additional background spiking of the periphery neurons amplify assembly activity. To test if the circuits of inputs, assemblies and outputs are functional, we strongly stimulate the input neurons of assembly 1 (Fig. 5.7c). This leads to the activation of assembly 1 and subsequently to the activation of the output neurons of assembly 1. Thus, the circuits provide basic input-output functionality. The other assemblies and periphery neurons are not activated, i.e. the activation is specific.

The simulations shown in Fig. 5.7 are done after the first complete remodeling of the network. To prevent the weight changes that would otherwise compensate the missing activity from the periphery

neurons in Fig. 5.7a,b, we freeze the connection weights. To activate the input neurons in Fig. 5.7c, we use externally stimulate them such that they are highly active for 0.5 s after $t = 4.75$ s. The circuit structure is not destroyed by the resulting network activity.

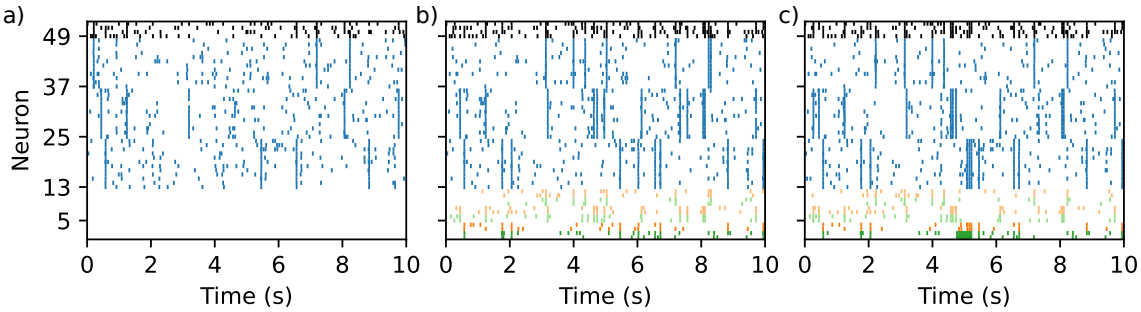

Figure 5.7: Associative memory property and basic input-output functionality of assemblies.
(a) The background activity of only the interior neurons is already sufficient to elicit assembly reactivations, which is required for associative memory.
(b) Assembly reactivations occur more frequently when the activity of periphery neurons is not suppressed.
(c) Assemblies provide basic input-output functionality. Stimulating a pair of input neurons (neurons 1,2) activates their assembly (neurons 13–24), which in turn specifically activates its output neurons (neurons 3,4). Spike trains are sorted according to their assembly membership at $t = 0$ s, starting with the periphery neurons of assemblies 1,2,3 (green: input neurons, orange:output neurons) and followed by the first twelve interior neurons of each assembly (blue). The spike trains of four inhibitory neurons are shown in black.

5.A.4 Assembly drift in a network without periphery neurons

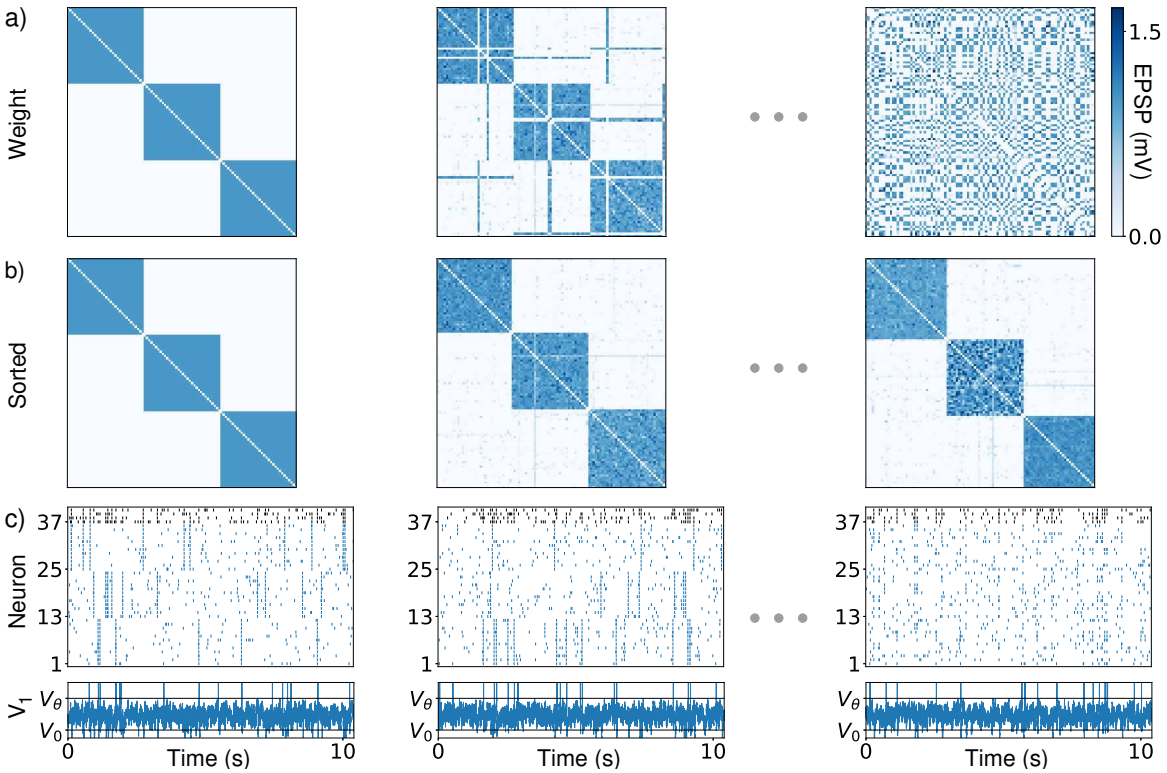

Figure 5.8: Assembly drift in a network without periphery neurons. Display is like in Fig. 5.3 but without periphery neurons and the second column shows simulation results after 15 minutes and the third column shows simulation results after 12h.

5.A.5 Spontaneous development of drifting assemblies

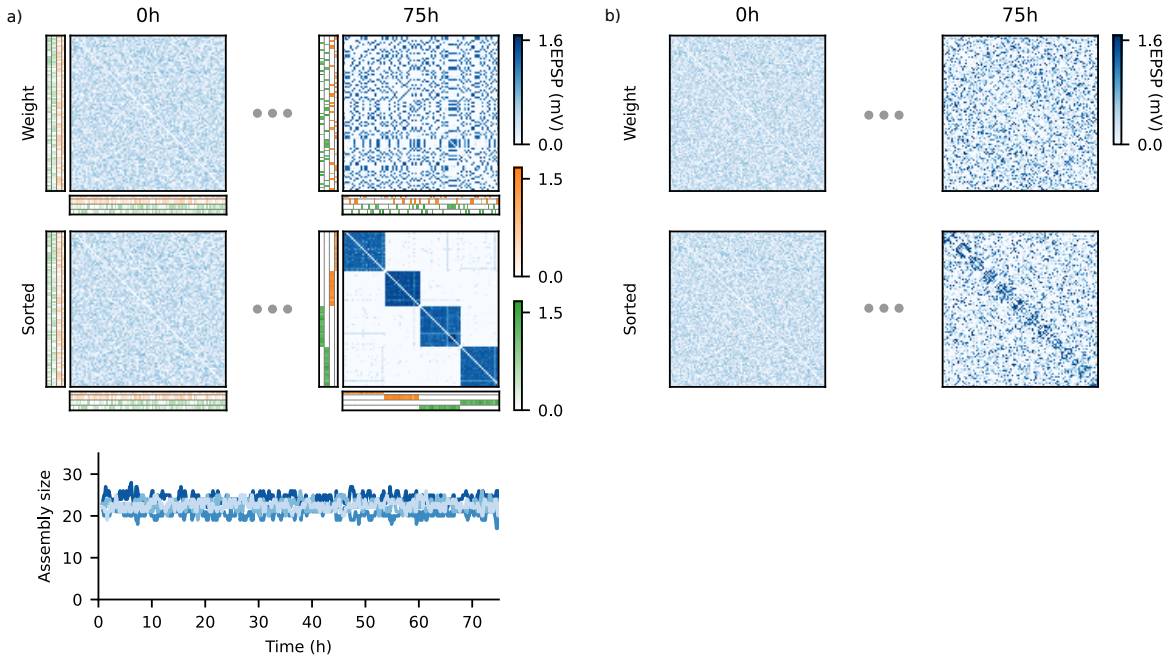

Figure 5.9: Spontaneous development of drifting assemblies.
(a) Four assemblies emerge spontaneously after randomly initializing networks with periphery neurons. Specifically, we initialize the networks by randomly drawing the weights of the connections between excitatory neurons from a uniform distribution and subsequently normalizing them (left column of top part, display like Fig. 5.3a,b). Four assemblies emerge within 70 min of simulated time. They persist and drift as indicated by the remodeled weight matrix after 75h of simulated time (right column of top part). Further, the assembly sizes fluctuate (bottom panel). The periphery neurons connect only randomly to the assemblies because of the random initial weights. The different number of periphery neurons per assembly is the reason why the mean assembly sizes differ from each other.
(b) No clear assemblies emerge after randomly initializing a network without periphery neurons. Display like the top part of (a).
These observations indicate together with Fig. 5.3 that not only the network parameters but also the initialization determines the number of assemblies in our networks. We confirmed our observations for five different network realizations. We note that while the periphery neurons mostly stayed connected to the same assemblies during the simulations of these five realizations, we observed one instance of a switch of a periphery neuron from an assembly originally connected to four periphery neurons to an assembly originally connected to only two periphery neurons (at about 13.8h).

5.A.6 Analysis of neuron transitions between assemblies for a network with three assemblies

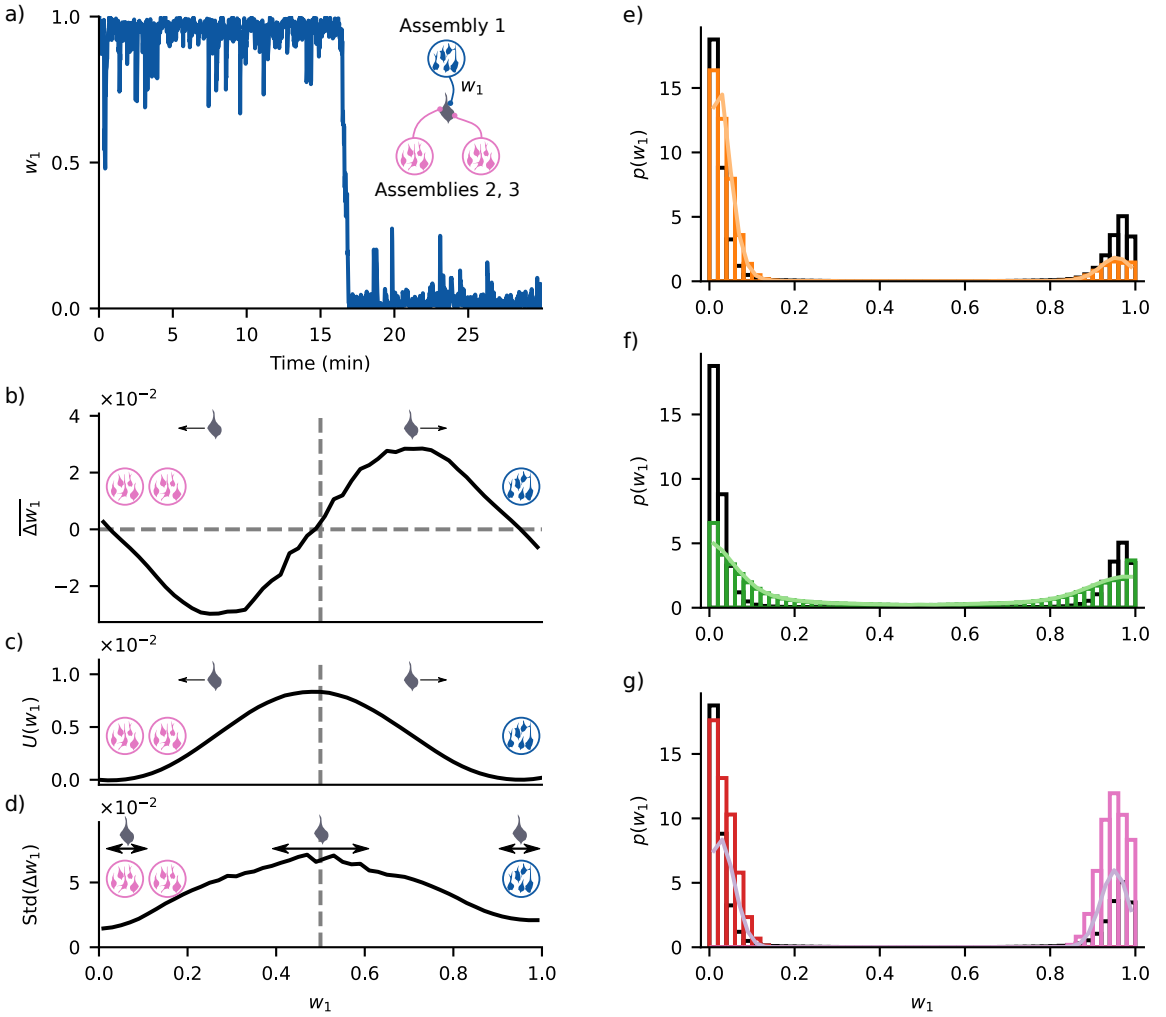

Figure 5.10: Weight dynamics and simplified model of neuron transitions between assemblies for a network with three assemblies.
(a–d) Display like Fig. 5.6a–d. The underlying network dynamics are those of Fig. 5.8. Like in Fig. 5.6a–d, $w_1 \approx 1$ means that the neuron is part of assembly 1. However, $w_1 \approx 0$ means that it is part of assembly 2 or 3. Thus, the summary statistics of the weight change are asymmetric.
(e–g) Display like Fig. 5.6e–g. For the same reason as for (a–d), the weightings of the high occupancy regions are asymmetric.

CHAPTER 6

Modeling feedback inhibition in epilepsy

This chapter consists of the following published article:

[4] L. Pothmann*, **C. Klos***, O. Braganza*, S. Schmidt, O. Horno, R.-M. Memmesheimer, H. Beck (* equal contribution)
Altered Dynamics of Canonical Feedback Inhibition Predicts Increased Burst Transmission in Chronic Epilepsy
Journal of Neuroscience **39** (2019) 8998–9012

The following sections contain the article with a shortened experimental part and further, minor changes. I contributed most of the modeling part of the study. In particular, I developed large parts of the model and the fitting procedure (Section 6.2.2), I performed the fitting (Section 6.3.4) and I simulated the network model when probed with external input (Section 6.3.5). Furthermore, I wrote large parts of the modeling related parts of the article.

6.1 Introduction

In the CNS, firing in neuronal ensembles is structured by the interaction of two fundamental categories of neurons: a majority of excitatory principal neurons and a minority ($\sim 10\,\%$ to $20\,\%$) of inhibitory, mostly GABAergic interneurons (see Sections 2.1.2 and 2.1.4). GABAergic interneurons mediate most of the shunting or hyperpolarizing inhibition in the adult brain, thereby powerfully controlling neuronal input-output behavior. Interneurons display a staggering diversity, with a large number of subtypes [223]. These different interneuron types are organized into two fundamental categories of inhibition. Feedforward inhibition of pyramidal neurons results from activation of interneurons by the same synaptic pathway that excites the pyramidal neuron. Feedback inhibition, on the other hand, is recruited by pyramidal cell firing and activation of recurrent inhibitory microcircuits. In the cortex and hippocampus, feedback inhibition exerts powerful control on output generation of pyramidal neurons [224, 225]. Different types of interneurons contribute to feedback inhibition, including interneuron classes mediating distal dendritic inhibition targeting the fine apical and basal branches of pyramidal neurons or proximal perisomatic inhibition. Intriguingly, somatically and dendritically targeting interneurons can be differentially recruited by different input frequencies, giving rise to

timed and domain-specific inhibition of CA1 pyramidal neurons [226]. Although feedback inhibition in the CA1 region is known to powerfully influence CA1 excitability, its alteration in chronic epilepsy and the resulting consequences for CA1 input-output transformations are unknown. Indeed, effects on CA1 input-output transformation may be particularly relevant under conditions of epileptiform activity of the upstream CA3 region. We have therefore examined the function of feedback inhibitory circuits in the normal and epileptic hippocampus. We find a pronounced change in the dynamics of inhibition of CA1 neurons that is rooted in both synaptic and intrinsic changes within the feedback circuit: Normally, synchronous activity of CA1 pyramidal cells recruits initially strong feedback inhibition that shows use-dependent depression. In contrast, initial inhibition is strongly reduced in chronic epilepsy. A biophysically constrained computational model suggests that these changed properties of feedback circuits promote the transmission of synchronous activity from CA3 via CA1 to other brain regions.

6.2 Material and Methods

6.2.1 Experiments

Pilocarpine model of epilepsy

Briefly, male Wistar rats were injected with pilocarpine hydrochloride. Within 60 min of injection, 30 % to 50 % of the animals developed a limbic status epilepticus (SE) that was terminated 40 min after SE onset. Only rats displaying at least one spontaneous seizure were included in this study. Experiments were conducted 4 to 8 weeks following experimentally induced SE. Sham-control animals were treated in an identical manner but were injected with saline instead of pilocarpine. Data from a subset of cells in the sham-control condition have already been used in ref. [227].

Slice preparation and patch-clamp recording

Transverse 300 µm thick hippocampal slices were prepared on a vibratome. Interneurons or pyramidal cells were visually identified under infrared difference interference contrast optics, and further characterized functionally as well as morphologically by biocytin labeling and reconstruction. All animal experiments were conducted in accordance with the guidelines of the Animal Care and Use Committee of the University of Bonn.

Analysis of intrinsic properties

The properties of postsynaptic currents (PSCs)/postsynaptic potentials (PSPs) were analyzed from an average of $\sim$ 10 sweeps. Input-output properties were assessed by applying successively increasing 1 s current pulses up to 800 pA (0, 10, 20, 30, 40, 60, 80, 100, 125, 150, . . . ,775, 800 pA). The maximal firing rate was determined as the maximally obtained rate with injections up to 800 pA. The current for half-maximal firing was determined through sigmoidal fits.

Analysis of feedback inhibitory circuit

For activation of CA1 feedback microcircuits, we stimulated CA1 axons by placing a electrode into the alveus adjacent to the subiculum and applying pulses with a duration of 0.1 ms. This stimulation

leads to antidromic activation of CA1 axons and recruitment of feedback inhibitory circuits. To prevent a direct monosynaptic excitation or inhibition, a cut was made at the CA1-subiculum border through strata lacunosummoleculare, radiatum, pyramidale, and oriens with only the alveus left intact [226]. A second cut was made at the CA1/CA3 border to limit spontaneous activity in CA1 neurons. Feedback EPSPs were recorded in inhibitory interneurons to monitor temporal summation. Stimulation strength was set according to the following two criteria (identically in sham-control and epileptic animals): (1) the stimulation should not elicit action potentials (APs), to allow proper estimation of the EPSP amplitudes; and (2) stimulation strength should be sufficiently large to elicit reliable EPSPs. The stimulation amplitudes were systematically varied at the beginning of each experiment until a stimulation strength that matched both criteria was found. The resultant stimulation strengths did not differ between sham-control and epileptic animals (mean(SD): 156(78) A and 186(96) A for cells from sham and epileptic slices respectively; $n = 34$ and $n = 37$, respectively; $p = 0.156$, unpaired Student's t test). Only interneurons that could be unambiguously identified based on morphology were included (81 of 171, see below). IPSCs were recorded from pyramidal neurons that result from activation of feedback interneurons. In these experiments, PSCs (rather than PSPs) were measured because they allowed to better distinguish individual IPSCs in train stimulations. IPSC amplitudes were measured from pre-stimulus-train baseline.

Cell classification and morphological analysis

BCs were identified via their distinctive axon distribution in the pyramidal cell layer. PD cells comprise cells whose axon ramified within stratum oriens and stratum radiatum (and potentially also stratum pyramidale). OLM cells were identified based on soma location in stratum oriens and visible axon in stratum lacunosum moleculare.

Statistical analysis

Average values in the text are expressed as mean ± SEM unless stated otherwise

6.2.2 Computational model

In the following, we give a detailed description of the models we used for the basket and pyramidal cells and their synapses and of the model of the complete feedback motif. Furthermore, we describe the fitting procedure we used. Model and fitting were implemented using MATLAB R2018a (The MathWorks).

Model of BCs

To model the excitation of the BC population, we used a nonlinear leaky integrator neuron model with a current-based synapse that exhibits short-term plasticity (STP). To model the STP, we used a modified version of the model introduced by Markram and Tsodyks [106] (see also Section 2.2.1). Specifically, the state of the population is represented by an effective average membrane voltage (or by the membrane potential of one representative neuron) $V_{\mathrm{BC}}(t)$, whose dynamics are governed by the

following:

$$\tau_{\mathrm{d,BC}} \frac{dV_{\mathrm{BC}}}{dt}(t) = -V_{\mathrm{BC}}(t) + \lambda V_{\mathrm{BC}}^2(t) + I_{\mathrm{e,BC}}(t), \tag{6.1}$$

where $\tau_{\mathrm{d,BC}}$ is the membrane time constant, λ specifies the strength of the nonlinearity, and $I_{\mathrm{e,BC}}(t)$ is the input current. Since the BCs do not spike in the experimental setting, we did not include a term covering spikes and resets in the model. The input current describes the effect of the synaptic input on the BC population. For the fit to the experimental data, we assume that the population of neurons receives similar inputs from a number of synapses that are excited in the same way by the stimulation. The total input is the linear superposition of the individual synaptic currents. Like a single synaptic current, it obeys the following:

$$\tau_{\mathrm{e,BC}} \frac{dI_{\mathrm{e,BC}}}{dt}(t) = -I_{\mathrm{e,BC}}(t) + \tau_{\mathrm{e,BC}} \hat{I}_{\mathrm{e,BC}} \tilde{I}_{\mathrm{e,BC}} u_{\mathrm{e,BC}}(t^-) x_{\mathrm{e,BC}}(t^-) \sum_{t_{\mathrm{sp}}} \delta(t - t_{\mathrm{sp}}), \tag{6.2}$$

where $\tau_{\mathrm{e,BC}}$ is the synaptic time constant, $u_{\mathrm{e,BC}}(t^-)$ is the fraction of available neurotransmitters released from the readily releasable pool when a spike arrives at the synapse, $x_{\mathrm{e,BC}}(t^-)$ is the current fraction of available neurotransmitters in the readily releasable pool when a spike arrives at the synapse, $\hat{I}_{\mathrm{e,BC}}$ specifies the impact strength of the released fraction of neurotransmitters, and $\tilde{I}_{\mathrm{e,BC}}$ is a normalization constant, which ensures that the EPSP amplitude is $\hat{I}_{\mathrm{e,BC}}$ if $\lambda = 0$ and if $u_{\mathrm{e,BC}}(t^-)$ and $x_{\mathrm{e,BC}}(t^-)$ assume their asymptotic values (as after an infinitely long preceding interspike interval). The t_{sp} 's are the times of spike arrival at the synapse, set according to the considered stimulation protocol. $x_{\mathrm{e,BC}}(t^-)$ introduces short-term synaptic depression and is determined by the following:

$$\tau_{\mathrm{RRP,e,BC}} \frac{dx_{\mathrm{e,BC}}}{dt}(t) = 1 - x_{\mathrm{e,BC}}(t) + \tau_{\mathrm{RRP,e,BC}} u_{\mathrm{e,BC}}(t^-) x_{\mathrm{e,BC}}(t^-) \sum_{t_{\mathrm{sp}}} \delta(t - t_{\mathrm{sp}}), \tag{6.3}$$

where $\tau_{\mathrm{RRP,e,BC}}$ is the time constant of the depression. $u_{\mathrm{e,BC}}(t)$ introduces short-term synaptic facilitation and is determined by the following:

$$\tau_{\mathrm{fac,e,BC}} \frac{du_{\mathrm{e,BC}}}{dt}(t) = u_{0,\mathrm{e,BC}} - u_{\mathrm{e,BC}}(t) + \tau_{\mathrm{fac,e,BC}} u_{\mathrm{f,e,BC}} (1 - u_{\mathrm{e,BC}}(t^-)) \sum_{t_{\mathrm{sp}}} \delta(t - t_{\mathrm{sp}}) \tag{6.4}$$

Here, $\tau_{\mathrm{fac,e,BC}}$ is the time constant of the facilitation, $u_{0,\mathrm{e,BC}}$ is the asymptotic release fraction corresponding to the release fraction of vesicles after an infinitely long preceding interspike interval, and $u_{\mathrm{f,e,BC}}$ characterizes the size of the jump toward 1 upon spike arrival. In the original version of this model [106], the jump size was also set to $u_{0,\mathrm{e,BC}}$, such that it served a double role (jump size and equilibrium fraction of released neurotransmitters), which does not seem justified for our data. All parameter values were determined through the fitting procedure described below for both the control and epileptic case.

Model of pyramidal cells

To model the pyramidal cell feedback inhibition, we first computed the effective membrane potential of the BC population in response to the stimuli with a model extending the model we used to reproduce

the BC data. The extended model incorporates a term that covers resets after spiking, depending on an estimate of the population-averaged firing rate of a single BC. We then used a synapse model, which again included STP, to determine the inhibitory current in the pyramidal cells as a function of the rate of the BC population. We computed the input current of the mean BC with Eqs. (6.2) to (6.4). To account for the reset of the membrane potential after spikes, we extended Eq. (6.1) to the following:

$$\tau_{\mathrm{d,BC}} \frac{dV_{\mathrm{BC}}}{dt}(t) = -V_{\mathrm{BC}}(t) + \lambda V_{\mathrm{BC}}^2(t) + I_{\mathrm{e,BC}}(t) - \tau_{\mathrm{d,BC}} V_{\mathrm{BC}}(t) r_{\mathrm{BC}}(t). \tag{6.5}$$

Here, $r_{\mathrm{BC}}(t)$ is the estimate of the population-averaged instantaneous firing rate of the BCs. It is given by the following:

$$r_{\mathrm{BC}}(t) = \hat{r}_{\mathrm{BC}} \log\left(1 + \exp\left(\frac{V_{\mathrm{BC}}(t) - V_{\mathrm{th,BC}}}{V_{\mathrm{w,BC}}}\right)\right), \tag{6.6}$$

where $V_{\mathrm{th,BC}}$ is the firing threshold, $V_{\mathrm{w,BC}}$ specifies the softness of the firing threshold, and $\hat{r}_{\mathrm{BC}}$ determines how fast the firing rate increases with increasing $V_{\mathrm{BC}}(t)$. With the exception of $\hat{I}_{\mathrm{e,BC}}$, all parameters that occur already in Eqs. (6.1) to (6.4) were set to the values found by the fit of the BC model to the cell-averaged BC data. We had to newly fit $\hat{I}_{\mathrm{e,BC}}$, since the level of stimulation of BCs was different (higher) in the current paradigm, such that they received a suprathreshold input. Furthermore, instead of fitting the parameters occurring in Eq. (6.6) to the data, we preset $V_{\mathrm{th,BC}} = 6\,\mathrm{mV}$, $V_{\mathrm{w,BC}} = 0.2\,\mathrm{mV}$ and $\hat{r}_{\mathrm{BC}} = 20\,\mathrm{Hz}$ as the quality of the fit to the pyramidal cell data is highly insensitive to the exact values of these parameters. This is because some of the parameters in the model have a similar effect on the output firing rate. For example, an increase of $\hat{I}_{\mathrm{e,BC}}$ and a decrease of $V_{\mathrm{th,BC}}$ both primarily led to a higher firing rate of the BCs. The fixed parameters could thus be chosen within a broad range where the overall activity of the BCs during the stimulation protocols is biologically plausible. In particular, while firing rates can be high during excursions of the voltage above threshold, such excursions are brief in our data, indicating the generation of only a few spikes. Using the firing rate of the BCs, we computed the inhibitory current in the pyramidal cells. The inhibitory current in the pyramidal cells $I_{\mathrm{i,PY}}(t)$ is determined by the following:

$$\tau_{\mathrm{i,PY}} \frac{dI_{\mathrm{i,PY}}}{dt}(t) = -I_{\mathrm{i,PY}}(t) + \tau_{\mathrm{i,PY}} \hat{I}_{\mathrm{i,PY}} \tilde{I}_{\mathrm{i,PY}} u_{\mathrm{i,PY}}(t) x_{\mathrm{i,PY}}(t) r_{\mathrm{BC}}(t), \tag{6.7}$$

where $\tau_{\mathrm{i,PY}}$ is the synaptic time constant, $u_{\mathrm{i,PY}}(t)$ is the release fraction of the neurotransmitters, $x_{\mathrm{i,PY}}(t)$ is the fraction of available neurotransmitters in the readily releasable pool, $\hat{I}_{\mathrm{i,PY}}$ specifies the impact strength of the released fraction of neurotransmitters, and $\tilde{I}_{\mathrm{i,PY}}$ is a normalization constant, which ensures that the IPSC amplitude is $\hat{I}_{\mathrm{i,PY}}$ if $u_{\mathrm{i,PY}}(t)$ and $x_{\mathrm{i,PY}}(t)$ assume their asymptotic values. $x_{\mathrm{i,PY}}(t)$ introduces short-term depression and is determined by the following:

$$\tau_{\mathrm{RRP,i,PY}} \frac{dx_{\mathrm{i,PY}}}{dt}(t) = 1 - x_{\mathrm{i,PY}}(t) + \tau_{\mathrm{RRP,i,PY}} u_{\mathrm{i,PY}}(t^-) x_{\mathrm{i,PY}}(t^-) r_{\mathrm{BC}}(t), \tag{6.8}$$

where $\tau_{\mathrm{RRP,i,PY}}$ is the time constant of the depression. $u_{\mathrm{i,PY}}(t)$ introduces short-term facilitation and is determined by the following:

$$\tau_{\mathrm{fac,i,PY}} \frac{du_{\mathrm{i,PY}}}{dt}(t) = u_{0,\mathrm{i,PY}} - u_{\mathrm{i,PY}}(t) + \tau_{\mathrm{fac,i,PY}} u_{\mathrm{f,i,PY}} (1 - u_{\mathrm{i,PY}}(t^-)) r_{\mathrm{BC}}(t). \tag{6.9}$$

Here, $\tau_{\mathrm{fac,i,PY}}$ is the time constant of the facilitation, $u_{0,\mathrm{i,PY}}$ is the asymptotic release fraction corresponding to the release fraction of vesicles after an infinitely long preceding interval with no presynaptic activity, and $u_{\mathrm{f,i,PY}}$ characterizes the increase of the release fraction of the neurotransmitters. All so far unspecified parameter values were determined through the fitting procedure described below for both the control and epileptic case.

Model of the complete feedback motif

To determine how the complete inhibitory feedback loop reacts to input, we combined the basket and pyramidal cell models. The two population models are recurrently connected and have firing rates $r_{\mathrm{BC}}(t)$ and $r_{\mathrm{PY}}(t)$. The pyramidal cell population additionally receives as excitatory input a firing rate $r_{\mathrm{IN}}(t)$ from CA3. We computed the firing rates of the basket and pyramidal cells using the experimentally measured relations between input current and firing rate (see Fig. 4B).

Pyramidal cells

The firing rate of an average pyramidal neuron as a function of the input current $Ipy(t)$ is given by the following:

$$r_{\mathrm{PY}}(t) = r_{0,\mathrm{PY}} + \frac{r_{1,\mathrm{PY}}}{1 + \exp\left(-\frac{I_{\mathrm{PY}}(t) - I_{\frac{1}{2},\mathrm{PY}}}{I_{\mathrm{w,PY}}}\right)} \tag{6.10}$$

For the control case, fitting the experimental data yielded the parameters $r_{0,\mathrm{PY}} = 1.60\,\mathrm{Hz}$, $r_{1,\mathrm{PY}} = 23.03\,\mathrm{Hz}$, $I_{\frac{1}{2},\mathrm{PY}} = 259.91\,\mathrm{pA}$ and $I_{\mathrm{w,PY}} = 96.38\,\mathrm{pA}$. For the epileptic case, we obtained $r_{0,\mathrm{PY}} = 1.74\,\mathrm{Hz}$, $r_{1,\mathrm{PY}} = 31.63\,\mathrm{Hz}$, $I_{\frac{1}{2},\mathrm{PY}} = 377.32\,\mathrm{pA}$ and $I_{\mathrm{w,PY}} = 132.75\,\mathrm{pA}$. The input current to the pyramidal cells is the sum of an excitatory current $-I_{\mathrm{e,PY}}(t)$ and an inhibitory current $-I_{\mathrm{i,PY}}(t)$. As there are no experimental data available about the STP of the excitatory synapses between CA3 neurons and CA1 in epilepsy, we assumed for simplicity that this synapse generally does not exhibit STP. Thus, the excitatory current is governed by the following:

$$\tau_{\mathrm{e,PY}} \frac{dI_{\mathrm{e,PY}}}{dt}(t) = -I_{\mathrm{e,PY}}(t) + \tau_{\mathrm{e,PY}} \hat{I}_{\mathrm{e,PY}} r_{\mathrm{IN}}(t), \tag{6.11}$$

where $\tau_{\mathrm{e,PY}}$ is the synaptic time constant and $\hat{I}_{\mathrm{e,PY}}$ is the coupling strength between the external input and the pyramidal cells. For both the control and the epileptic case, we set $\tau_{\mathrm{e,PY}} = 5\,\mathrm{ms}$ in rough accordance to experimental data [228, 229] and $\hat{I}_{\mathrm{e,PY}} = 2\,000\,\mathrm{pA}$. We selected the values of this and all further coupling strengths to achieve biologically plausible activity levels, with the same coupling strength in the control and epileptic case. To determine the inhibitory current $I_{\mathrm{i,PY}}(t)$, we used Eqs. (6.7) to (6.9) with the parameters set to the values obtained from the fit to the cell-averaged pyramidal cell data, except for $\hat{I}_{\mathrm{i,PY}}$, as the inhibition is not a result of external stimulation anymore but models biological neuronal activity. Specifically, we set $\hat{I}_{\mathrm{i,PY}} = 1\,000\,\mathrm{pA}$ in both the control and the epileptic case. We used the same value for both cases as the fit to the cell-averaged pyramidal cell data also yielded similar values for $\hat{I}_{\mathrm{i,PY}}$ in the control and epileptic case.

BCs

The firing rate of an average BC as a function of the input current $I_{\mathrm{e,BC}}(t)$ is given by the following:

$$r_{\mathrm{BC}}(t) = r_{0,\mathrm{PY}} + \frac{r_{1,\mathrm{BC}}}{1 + \exp\left(-\frac{I_{\mathrm{e,BC}}(t) - I_{\frac{1}{2},\mathrm{BC}}}{I_{\mathrm{w,BC}}}\right)} \tag{6.12}$$

For the control case, we obtained the parameters $r_{0,\mathrm{BC}} = 12.15\,\mathrm{Hz}$, $r_{1,\mathrm{BC}} = 141.02\,\mathrm{Hz}$, $I_{\frac{1}{2},\mathrm{BC}} = 383.44\,\mathrm{pA}$ and $I_{\mathrm{w,BC}} = 162.37\,\mathrm{pA}$ and for the epileptic case the parameters $r_{0,\mathrm{BC}} = 1.09\,\mathrm{Hz}$, $r_{1,\mathrm{BC}} = 104.58\,\mathrm{Hz}$, $I_{\frac{1}{2},\mathrm{BC}} = 487.64\,\mathrm{pA}$ and $I_{\mathrm{w,BC}} = 107.16\,\mathrm{pA}$. We computed the input current $I_{\mathrm{e,BC}}(t)$ similarly to Eqs. (6.2) to (6.4). Specifically, we replaced $\sum_{t_{\mathrm{sp}}} \delta(t - t_{\mathrm{sp}})$ with $r_{\mathrm{PY}}(t)$ in Eqs. (6.2) to (6.4) and additionally replaced $\hat{I}_{\mathrm{e,BC}}\tilde{I}_{\mathrm{e,BC}}$ with $\hat{I}_{\mathrm{e,BC}}/u_{0,\mathrm{e,BC}}$ in Eq. (6.2). The parameters are set to the values obtained from the fit to the cell-averaged BC data, except for the coupling strength $\hat{I}_{\mathrm{e,BC}}$, for the same reason as before. We set $\hat{I}_{\mathrm{e,BC}} = 15\,000\,\mathrm{pA}$ in the control case and $\hat{I}_{\mathrm{e,BC}} = 10\,000\,\mathrm{pA}$ in the epileptic case. We chose these different values as the coupling strength was ~ 1.5 times higher in the control compared with the epileptic case for the fit to the cell-averaged BC data. We extracted the coupling strength from the fit to the cell-averaged BC data by dividing $\hat{I}_{\mathrm{e,BC}}\tilde{I}_{\mathrm{e,BC}}$ by the mean input resistance of the cells used for the fitting.

Fitting procedure

We fitted the BC model to the voltage traces measured in BCs and the pyramidal cell model to the current traces measured in pyramidal cells during both the 50 Hz and theta burst protocol. As described above, we used the fit to the cell-averaged data to describe the behavior of the cell populations as a whole, which is the characteristic relevant for modeling the feedback circuit. The goal of the fitting procedure was to minimize following model error:

$$E = \sqrt{\frac{1}{N_{50}} \sum_{t_{50}} (Y_{50}(t_{50}) - X_{50}(t_{50}))^2 + \frac{1}{N_{\mathrm{TB}}} \sum_{t_{\mathrm{TB}}} (Y_{\mathrm{TB}}(t_{\mathrm{TB}}) - X_{\mathrm{TB}}(t_{\mathrm{TB}}))^2}, \tag{6.13}$$

where t_{50} and t_{TB} are the time steps in the 50 Hz and theta burst protocol and N_{50} and N_{TB} are their numbers. $Y_{50}(t_{50})$ is the experimentally observed potential of the BCs (current in the pyramidal cells) at time step t_{50} in the 50 Hz protocol and $X_{50}(t_{50})$ is the corresponding value in our model. Likewise, $Y_{\mathrm{TB}}(t_{\mathrm{TB}})$ is the experimentally observed potential of the BCs (current in the pyramidal cells) at time step t_{TB} in the theta burst protocol and $X_{\mathrm{TB}}(t_{\mathrm{TB}})$ is the corresponding value in our model. To validate the fitted parameter combinations in the biologically plausible range of parameters, we generated error surfaces. This approach safeguards against local minima and allows to display the parameter region that yields good fits and its localization (see Fig. 6.4C,D). The first step was to compute the model error on a regular grid in the parameter space. This grid was located in a region of biologically plausible parameter values (see Tables 6.1 and 6.2). In the case of the BC model, this grid consisted of 10^8 points, with 10 regularly spaced values for each of the 8 parameters; and in the case of the pyramidal cell model, it consisted of 13^7 points, with 13 regularly spaced values for each of the 7 parameters (see Fig. 6.4C,D). In the second step, we first determined the local minima of the model error on this grid. Starting from each of the found minima, we then performed a localized grid search to further minimize the model error. Typically, the grid search yielded two parameter sets with similarly small fitting

error, whose only substantial differences were that one of the sets exhibited no short-term facilitation whereas the other point exhibited a small amount of facilitation (e.g., large $u_{\mathrm{f,e,BC}}$ but small $\tau_{\mathrm{fac,e,BC}}$). This indicates that facilitation is negligible. We thus chose the point corresponding to the simpler model without short-term facilitation for the further analysis.

6.3 Results

All canonical feedback circuits consist of the same key elements: excitatory connections from a principal cell population that synaptically recruit inhibitory interneurons, which then inhibit the principal cell population. We have systematically examined the properties of the key elements of feedback circuits in the hippocampal CA1 region, and how they change in chronic epilepsy.

6.3.1 Altered activation of CA1 interneurons within feedback microcircuits

We first determined the properties of feedback activation of three categories of CA1 interneurons targeting different areas of the somatodendritic axis of pyramidal cells (Fig. 6.1A,B). Only interneurons that could be unambiguously classified into one of these three categories based on their axon morphology were included in this study (see Materials and Methods). The first category consisted of BCs that innervate pyramidal cell somata (Fig. 6.1A left). A second group included cells that target the proximal dendrites of pyramidal cells in stratum radiatum and oriens (i.e., bistratified cells) [80], as well as cells that additionally innervate stratum pyramidale (i.e., trilaminar cells) and were collectively termed PDs. Third, we examined OLM interneurons with somata located in stratum oriens and axonal projections innervating the distal pyramidal cell dendrites in stratum lacunosum moleculare. Finally, we also examined CA1 pyramidal neurons in sham-control and epileptic animals (Fig. 6.1A right). As BCs provide the dominant perisomatic inhibition to pyramidal neurons and showed the most robust changes in feedback activation and intrinsic properties in chronic epilepsy, we focus on them in this thesis.

In the three groups of interneurons, we then asked how they are recruited by feedback excitatory inputs from CA1 pyramidal neurons. Therefor, we first considered a 50 Hz stimulus train (10 stimulations; Fig. 6.1C left). Feedback excitation elicited with a stimulus electrode placed into the alveus revealed distinctive forms of short-term plasticity in the different types of CA1 interneurons in sham-control animals, as described previously [227]. BCs received a large amplitude excitatory input at the beginning of a 50 Hz stimulus train (10 stimulations) that was followed by synaptic depression (by −47.3(115) %, $n = 7$; Fig. 6.1D top left). In chronically epileptic animals, BCs displayed no depression in EPSP size over the stimulus train (0.1(118) %, $n = 13$; Fig. 6.1D bottom left). Out of all considered interneuron types, only BCs showed significant differences between control and epileptic animals ($p = 0.0084$; Mann–Whitney U test).

During exploratory behavior and REM sleep, firing of hippocampal pyramidal cells is phase-locked to the theta rhythm, a field potential oscillation of 5 Hz to 10 Hz [230]. We therefore explored the excitation of the different types of interneurons within feedback circuits with a theta burst protocol consisting of a high-frequency component (three stimuli at 100 Hz) repeated 10 times at a frequency of 5 Hz (Fig. 6.1C right). The peak response obtained during theta patterned bursts decreased strongly during the train in BCs of control animals (by −34.6(61) %, $n = 7$; Fig. 6.1D top right). In epileptic animals, BCs displayed a loss of depression during the theta stimulation train (−8.7(87) %, $n = 13$; Fig. 6.1D bottom right). This significantly differed from sham control animals ($p = 0.0123$;

Mann–Whitney U test).

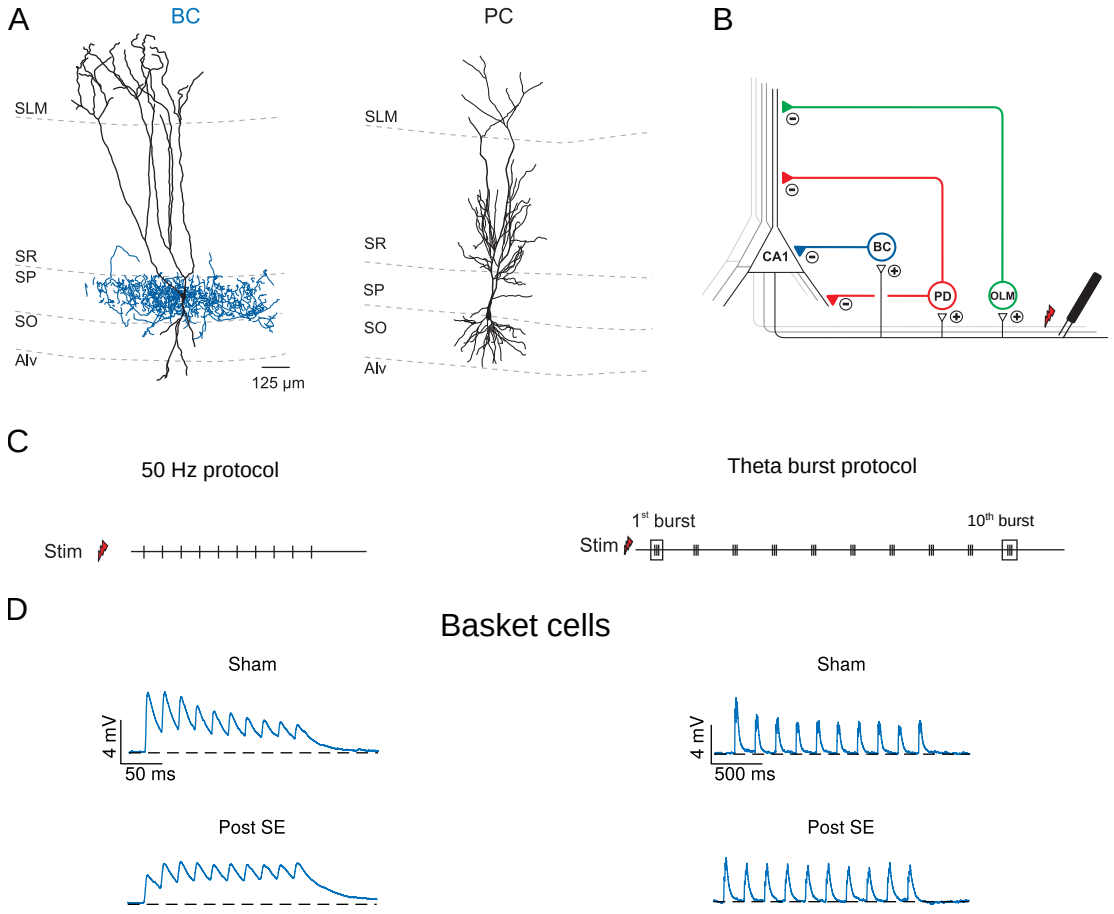

Figure 6.1: Epilepsy-associated changes in feedback recruitment of basket cells and feedback inhibition onto pyramidal cells.
(A) Representative morphological reconstructions of a basket and pyramidal cells in sham animals (same as Fig. 2.1). For the basket cell, axons are shown in blue and dendrites in black. No significant morphological differences between sham and epileptic animals were found. SLM: stratum lacunosum-moleculare, SR: stratum radiatum, SP: stratum pyramidiale, SO: stratum oriens, Alv: Alveus.
(B) Schematic diagram illustrating the recorded cell types as well as placement of the stimulus electrode.
(C) Schematics of the stimulation protocols.
(D) Cell-averaged EPSPs measured in basket cells used for the model fitting in response to the 50 Hz protocol (left) and the theta burst protocol (right) (control and epileptic animals, respectively, $n = 5$ and $n = 7$; for technical reasons, only a subset of all recorded cells could be used for the model fitting).

6.3.2 Altered firing behavior of interneurons

In addition to the dynamics of the synaptic excitatory drive, the recruitment of interneurons within inhibitory networks also depends strongly on their intrinsic properties. We thus examined the firing behavior induced by increasing 1 s current injections in sham-control and epileptic animals for each

of the cell categories (Fig. 6.2; BCs: $n = 9$ and $n = 6$ for sham and epileptic animals, respectively). The maximally obtained firing rate (with up to 800 pA current injections) was significantly increased only in pyramidal neurons, but not interneurons ($n = 13$, $n = 14$; 22.2(12) Hz vs 29.4(19) Hz for sham and epileptic animals, respectively; $p = 0.0070$; Student's t test). We then determined the current injection needed to achieve half-maximal discharge rates through sigmoidal fits to the individual cells. We found a pronounced shift of the input-output relation in BCs (316(55) pA vs 519(55) pA for sham and epileptic animals, respectively, $p = 0.0266$; Student's t tests). A similar change was seen in pyramidal neurons (259(19) pA vs 382(26) pA for sham and epileptic animals, respectively, $p = 0.0009$, Student's t test). Thus, in addition to changes in the properties of excitatory synapses driving them, there are large differences in intrinsic properties that reduce the ability of BCs to be recruited synaptically. Furthermore, pyramidal cells in epileptic animals can in principle reach higher activity levels.

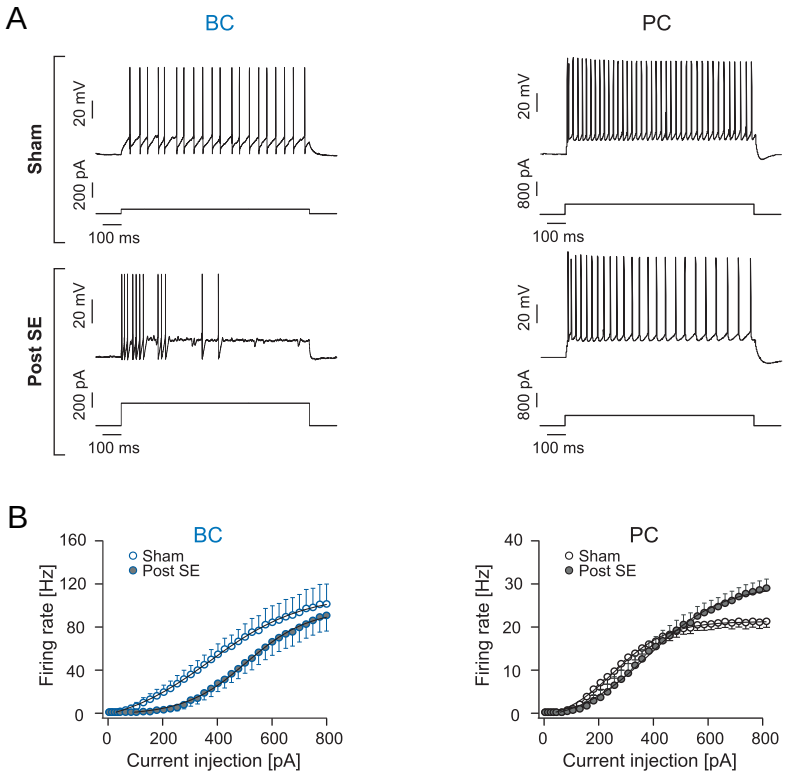

Figure 6.2: Excitability of basket and pyramidal cells in epileptic animals is altered.
(A) Representative discharge responses of a basket (left) and pyramidal (right) cell to 1 s current injections in sham (top) and epileptic (bottom) animals.
(B) Input-output relation of basket and pyramidal cells. The average firing rate was calculated and plotted versus the current injection. Solid lines indicate sigmoidal fits to the average data.

6.3.3 Altered recruitment of feedback inhibition onto pyramidal cells in chronic epilepsy

How do these changes collectively impact the time course and magnitude of feedback inhibition onto CA1 pyramidal cells? We directly examined this question by recording feedback inhibition elicited by alveus stimulation in control and epileptic animals in CA1 pyramidal neurons (Fig. 6.3). During 50 Hz stimulation (Fig. 6.3B,C left), the amplitude of IPSCs in control animals decreased strongly from the first to the 10th stimulus, with an average decrease of −77.8(17) % (n = 25). In pilocarpine-treated animals, the depression in amplitude was significantly diminished (average decrease: −44.9(76) %, n = 15; p = 0.0001, Mann–Whitney U test).

We next compared the absolute amplitude of the IPSCs elicited at the beginning and the end of the stimulus trains in pilocarpine-treated and control animals. In control animals, the first IPSC had an average amplitude of 197.5(237) pA, whereas in epileptic animals, IPSC amplitude was significantly smaller (85.0(171) pA, $p < 0.0001$, Bonferroni's multiple-comparisons post test). In contrast, the amplitude of the 10th IPSC in the stimulus train was not significantly changed in epileptic animals ($p > 0.9999$, Bonferroni's multiple-comparisons post test).

During theta patterned feedback stimulation, the differences in feedback inhibitory dynamics between control and epileptic animals were even more pronounced than during 50 Hz stimulation (Fig. 6.3B,C right). In control animals, the IPSC amplitude decreased by −51.44(381) % (n = 9) from the first to the 10th burst. In epileptic animals, this phenomenon was strongly attenuated, with an average reduction of peak IPSC amplitude of only −10.16(1101) % (n = 9, p = 0.0040, Mann–Whitney U test). As for the 50 Hz stimulation trains, epileptic animals displayed a strong reduction of IPSC amplitudes at the first burst stimulation in the train, but not the last (Bonferroni's multiple-comparison post test of control vs treatment: $p < 0.0001$ and $p = 0.6395$ for first and 10th stimulus, respectively). Thus, net inhibition of the feedback circuit shows markedly changed dynamics.

6.3.4 Generation of a computational model of the feedback circuit

Generation of a simple population model allowed us to further interpret our data. The changes between the response properties of neuron populations in sham-control and epileptic animals were reflected by quantitative and qualitative changes of model parameters. A mentioned earlier, we focused on inhibitory BC populations because we expected them to be most influential for controlling spike output of CA1 neurons.

The pyramidal neuron and BC population activities are represented by two population rates, $r_{\mathrm{PY}}(t)$ and $r_{\mathrm{BC}}(t)$, respectively (see Fig. 6.5A; see Section 6.2). We first modeled the feedback excitatory synapse activating BCs and the average BC population response to excitatory stimulation using a current-based synapse that exhibits STP and a simple leaky integrator model of the BC population. Short-term depression is modeled as depletion of a continuously replenishing pool of vesicles, where the "asymptotic release fraction" corresponds to the release fraction of vesicles assuming an infinitely long preceding interspike interval and the "timescale" measures how long replenishment takes. We determined the values of the model parameters that best capture the dynamics of activation of BCs by fitting the model to the average of the voltage traces measured in response to the stimulation of feedback excitatory synapses (as shown in Fig. 6.1). This provided an accurate description of the excitatory drive to the population of BCs (see Section 6.2).

The model allowed a good fit to the experimental data both in control animals and epileptic animals

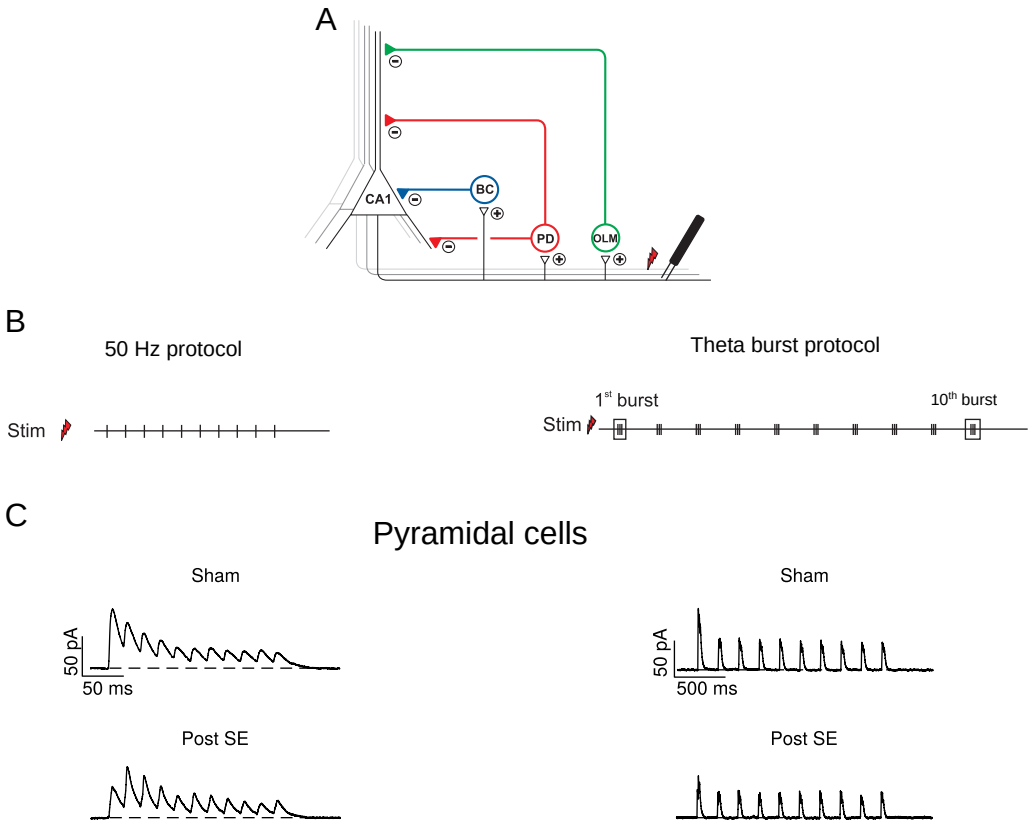

Figure 6.3: Epilepsy-associated changes in feedback inhibition onto pyramidal cells.
(A,B) Same as Fig. 6.1B,C.
(C) Cell-averaged IPSCs measured in pyramidal cells in response to the 50 Hz protocol (left) and the theta burst protocol (right) (control and epileptic animals, respectively, $n = 9$ and $n = 7$; for technical reasons, only a subset of all recorded cells could be used for the model fitting).

(Fig. 6.4A, left vs right; $R^2 = 0.98$ for both control and epileptic animals) and yields quantitative estimates for the effective characteristics of the BC population, such as the asymptotic release fraction and the timescale of replenishment. To verify that the fitting procedure was not trapped in a local minimum and to determine how sensitive the model error is to deviations from the found minimum, we computed the model error on a coarse-grained grid in parameter space. To visualize this approximate error surface, we projected the grid on the two key parameters describing short-term depression, asymptotic release fraction and timescale of replenishment, where all other parameters are optimized for a given combination of the two. The model only reproduces the data in control animals well if the asymptotic release fraction and the replenishment timescale lie close to the minimum found by the fit. In contrast, in the epileptic case, the model yields good results if either the asymptotic release fraction or the replenishment timescale is near minimal (Fig. 6.4C). This qualitative difference reflects the fact that short-term depression becomes negligible in epileptic animals. Incorporation of synaptic facilitation did not markedly improve the fit in sham and epileptic animals, indicating that it is negligible in the mean population output (Table 6.1).

Next, we considered the BC-to-pyramidal cell synapse, again using a current-based model including

STP. The already obtained model for the BC responses to excitatory stimulation allowed us to fit the model parameters describing the pyramidal cells and the BC-to-pyramidal cell synapse to the IPSC traces measured in response to stimulation in the alveus (Fig. 6.3). Again, the fit to data obtained from both control animals and epileptic animals was good (Fig. 6.4B; $R^2 = 0.95$ and $R^2 = 0.93$ for control and epileptic animals, respectively). Using the same approach as before, we determined an approximate error surface over the parameters describing the short-term depression of the BC-to-pyramidal cell inhibitory synapse, which was not directly experimentally measured in this study (Fig. 6.4D, Table 6.2). The results indicate that short-term depression is negligible in epileptic and weak in control animals.

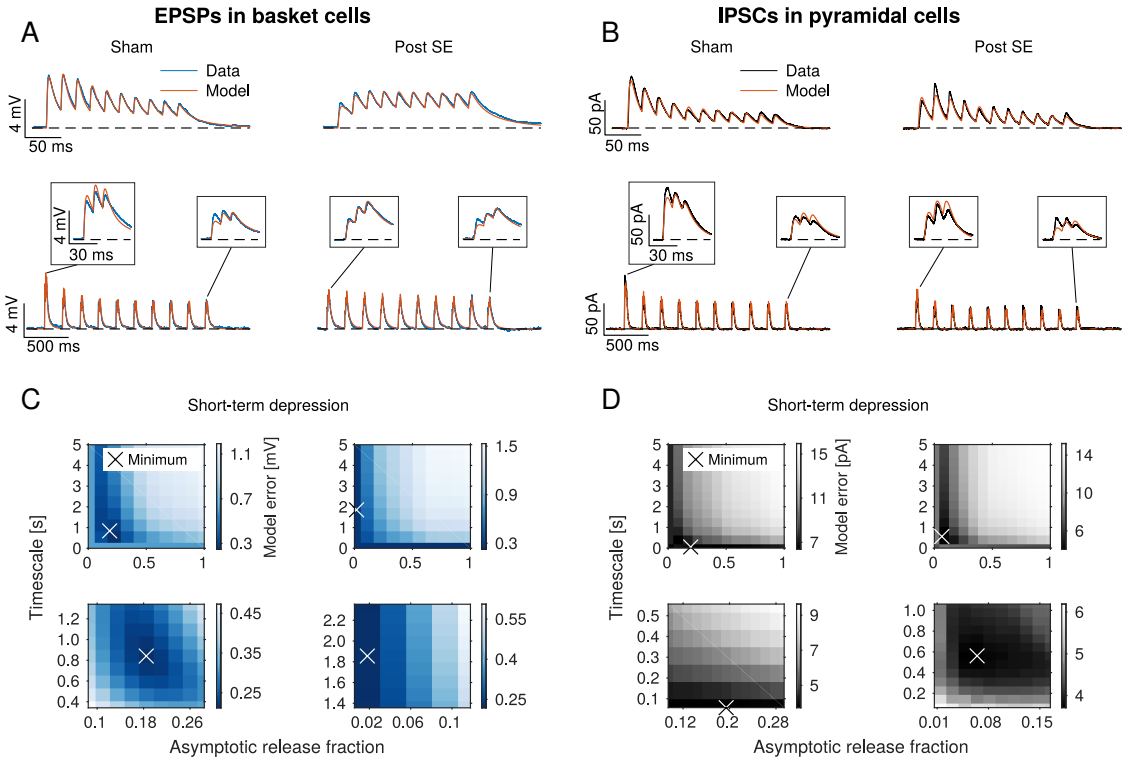

Figure 6.4: Modeling of a feedback inhibitory circuit.
(A) Cell-averaged EPSPs measured in BCs (blue traces; control and epileptic animals, respectively, $n = 5$ and $n = 7$) and EPSPs obtained from the fit of the model to these data (orange traces). Data and fit for both the 50 Hz protocol (top) and the theta burst protocol (bottom) are shown. Scale bars hold for all EPSPs that are in the same row.
(C) Error between experimentally measured EPSPs and EPSPs obtained from the model as a function of the parameters describing short-term depression. Bottom, close-up view of the error surface around the parameters obtained from the fit of the model to the data (crosses).
(B,D) Same as in (A,C), but for the IPSCs measured in pyramidal cells in control and epileptic animals ($n = 9$ and $n = 7$).

	$\tau_{d,BC}$ (ms)	λ	$\tau_{e,BC}$ (ms)	$\hat{I}_{e,BC}$ (mV)	$\tau_{RRP,e,BC}$ (ms)	$u_{0,e,BC}$	$\tau_{fac,e,BC}$ (ms)	$u_{f,e,BC}$	Model error (mV)
Search space	10 – 100	-1 – 0	0.005 – 5	0.02 – 20	5 – 5000	0.001 – 1	5 – 5000	0 – 1	
Sham-control	55.5	-0.952	1.790	6.82	841	0.185	5	0.000	0.212
Epileptic	56.8	-0.461	1.938	2.44	1856	0.018	5	0.000	0.219

Table 6.1: Parameters values found by the fitting procedure for the basket cell data in the control and epileptic case. The first row shows the parameter bounds within which the parameter space was searched for optimal values. $\tau_{d,BC}$, Membrane time constant; λ, strength of nonlinearity of membrane potential; $\tau_{e,BC}$, synaptic time constant; $\hat{I}_{e,BC}$, impact strength of released fraction of neurotransmitters; $\tau_{RRP,e,BC}$, time constant of synaptic depression; $u_{0,e,BC}$, asynptotic release fraction; $\tau_{fac,e,BC}$, time constant of synaptic facilitation; $u_{f,e,BC}$, increase of release fraction after a presynaptic spike; Model error, error between data and model. Parameters correspond to the basket cell population or the pyramidal cell-to-basket cell synapse.

	$\tau_{i,PY}$ (ms)	$\hat{I}_{i,PY}$ (pA)	$\tau_{RRP,i,PY}$ (ms)	$u_{0,i,PY}$	$\tau_{fac,i,PY}$ (ms)	$u_{f,i,PY}$	$\hat{I}_{e,BC}$ (mV)	Model error (mV)
Search space	0.015 – 15	0.1 – 500	5 – 5000	0.001 – 1	5 – 5000	0 – 1	3 – 300	
Sham-control	12.88	33.0	57	0.194	5	0.0	74.3	3.792
Epileptic	9.66	30.2	561	0.064	5	0.0	33.5	3.711

Table 6.2: Parameters values found by the fitting procedure for the pyramidal cell data in the control and epileptic case. The first row shows the parameter bounds within which the parameter space was searched for optimal values. $\tau_{i,PY}$, synaptic time constant; $\hat{I}_{i,PY}$, impact strength of released fraction of neurotransmitters; $\tau_{RRP,i,PY}$, time constant of synaptic depression; $u_{0,i,PY}$, asynptotic release fraction; $\tau_{fac,i,PY}$, time constant of synaptic facilitation; $u_{f,i,PY}$, increase of release fraction after a presynaptic spike; Model error, error between data and model. Parameters correspond to the pyramidal cell population or the basket cell-to-pyramidal cell synapse.

6.3.5 Consequences of altered feedback circuits: altered burst transmission from CA3 via CA1

We then asked what the consequences of the observed changes in the feedback circuit are for input-output conversion in the CA1 region. The experimental data show that recruitment of perisomatic inhibition in feedback networks is substantially altered, via both changes in intrinsic interneuron properties and their synaptic recruitment. These changes predict a substantial decrease in the initial inhibition that can be recruited when the CA1 pyramidal ensemble begins to discharge synchronously. Under conditions under which CA3 cells are synchronously active, as during epileptiform burst activity, feedback inhibition may then be particularly inefficient in controlling CA1 pyramidal cell excitability.

To test this prediction, we generated a model of the complete inhibitory feedback motif for the control and epileptic case, using the components described above (Fig. 6.5A; see Section 6.2). The feedback circuit model was then probed with inputs from CA3 that were systematically varied. Specifically, the input was represented by a rate, which increased with various degrees of steepness, and then stayed constant for 80 ms, thereafter dropping to zero again (Fig. 6.5B; r_{IN}, example for rate increase from 0 Hz to 70 Hz over 150 ms). These parameters were chosen based on the frequency and duration of CA3 ripples or epileptic bursts [96, 97, 231]. The entirety of the epilepsy-induced changes led to decreased BC activity and increased pyramidal cell activity (Fig. 6.5B; r_{BC} and r_{PY}, respectively; Fig. 6.5C,D). These changes were robust over a wide range of input rise times and maximal rates (Fig. 6.5C,D). Notably, the increase of pyramidal cell activity was especially pronounced for short rise times, which are typical for the initial phase of epileptic bursts (Fig. 6.5D, right). Hence, these results

predict that the changes in CA1 during development of epilepsy promote the transmission of epileptic bursts from CA3 to other parts of the brain.

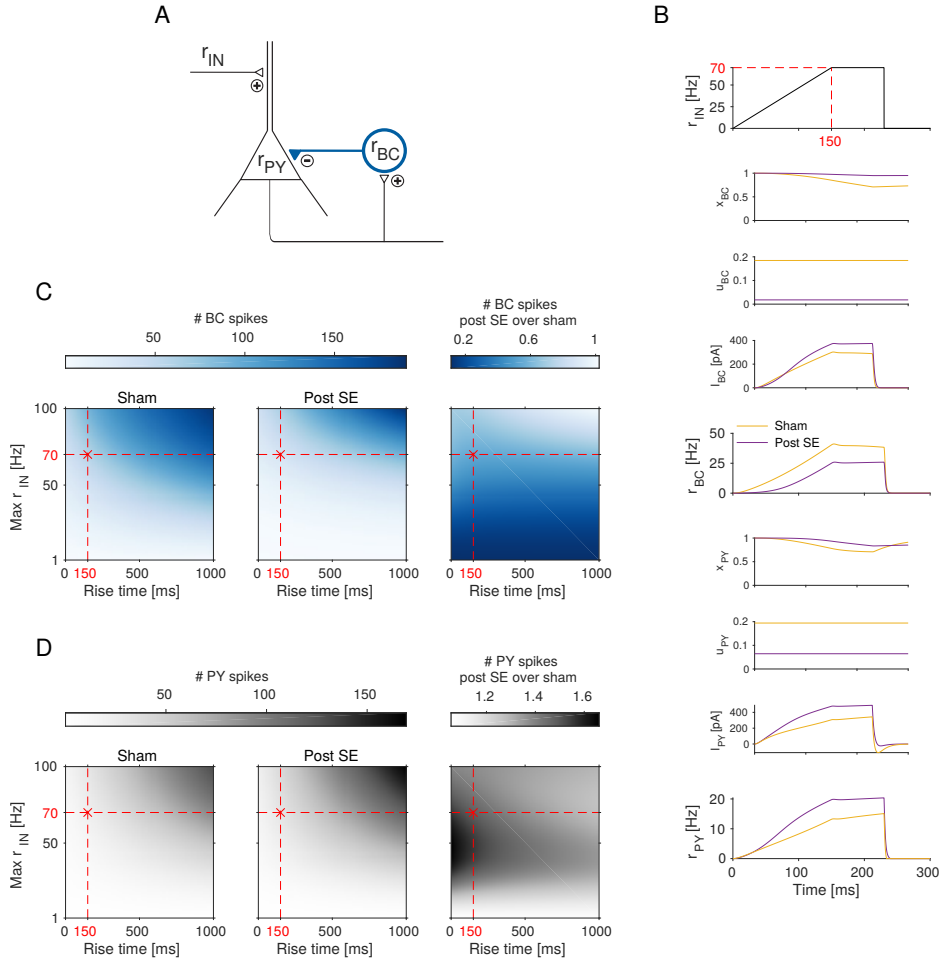

Figure 6.5: Increased burst transmission caused by altered dynamics of feedback inhibition.
(A) Schematic diagram of the model components.
(B) Example time evolution of all model variables in the control (orange) and epileptic case (purple) for an example input (top). x_{BC} is the fraction of available neurotransmitter in the readily releasable pool of vesicles, and u_{BC} is the release fraction of neurotransmitter for the pyramidal cell-to-BC synapse. IBC is the total input current to the BCs. x_{PY}, u_{PY}, and I_{PY} are the corresponding quantities for the BC-to-pyramidal cell synapse and the pyramidal cells (see Section 6.2).
(C) Number of spikes of the BCs in the epileptic case and in the control case (left subpanels) and their ratio for different input rise times and maximal firing rates (right subpanel). Red lines indicate the parameters shown in (B).
(D) Same as (C) but for pyramidal cells.

6.4 Discussion

In this chapter, we demonstrate and quantify major functional changes in feedback inhibition in chronic epilepsy, consisting of both intrinsic and synaptic changes. Collectively, these changes result in altered dynamics of CA1 feedback inhibition and predict decreased filtering of burst-like activity from the CA3 region.

Feedback inhibition plays a crucial role in controlling excitability of pyramidal neurons. Feedback excitation of perisomatically inhibiting interneurons normally shows strong short-term depression, as shown previously [226, 227, 232]. This is markedly changed in epilepsy, leading to a profound reduction of initial perisomatic inhibition. To assess the consequences of these and the other observed changes on the input-output properties of the CA1 region, we developed a simple, biologically plausible CA1 circuit model of the feedback inhibitory motif. The model was systematically fitted to our experimental data generated for the different elements in the feedback circuit. Probing it with inputs from CA3, we find that the epilepsy-associated changes in the feedback inhibitory motif cause an increased CA1 output. Notably, the increase is particularly pronounced in the case of steep rises of the input signal from CA3, which are typical for the initial phase of epileptic bursts. This indicates that the changes in CA1 during development of epilepsy foster the transmission of epileptic bursts from CA3 to other parts of the brain.

These findings may also be relevant to burst-like inputs occurring during sharp-wave ripple oscillations, which are driven by CA3 pyramidal neurons. In the normal brain, both basket and PD interneurons are efficiently recruited by such activity patterns. Accordingly, feedback inhibition impinging on the soma and proximal dendrites during these input patterns is strong. In epileptic animals, sharp-wave ripples are also observed, with distinct and more variable spectral features [98, 99]. In keeping with the predictions of our model, CA1 cells fired more in epileptic animals during sharp-wave ripples, as well as participating indiscriminately in multiple types of sharp-wave ripple events [99]. Moreover, these changes seemed to be due to altered excitation-inhibition balance, according to intracellular measurements of synaptic conductances, as well as pharmacological experiments. Thus, the circuit abnormality we have demonstrated appears to be relevant in vivo. Moreover, the deficit in inhibitory efficacy and timing of inhibition may interfere with proper ensemble selection of CA1 ensembles during sharp-wave ripples, which would be expected to degrade the information capacity of CA1 networks [99]. Because precise activation of CA1 ensembles during sharp-wave ripples is important for memory formation [75, 233], these findings are likely also relevant for impaired memory formation in the epileptic hippocampus.

In addition to the changes in short-term dynamics, we describe a pronounced decrease in intrinsic excitability of BCs. Notably, in contrast to the synaptic mechanisms described above, this change will impact interneuron excitability regardless of whether they are being recruited in feedback or feedforward circuits.

In the pilocarpine model, a number of studies have addressed the dysfunction of GABAergic systems. Consistent with our results regarding GABAergic bouton density (see our article [4]), the number of GABAergic boutons innervating CA1 pyramidal cell somata was not reduced in the pilocarpine model [77] as well as human tissue [234]. There was also no loss of spontaneous GABAergic IPSCs in CA1 pyramidal neurons [235–237]. However, there are more subtle effects regarding changes in specific subtypes of perisomatic interneurons in this model. There is a selective reduction in perisomatic CA1 pyramidal cell innervation from CCK-expressing BCs, with a loss of CCK and CB1 receptor-expressing boutons, whereas PV-expressing boutons [77] and PV-expressing somata in the

pyramidal cell layer [76] were unchanged. We did not differentiate between PV and CCK BCs in our analyses, but our finding of altered dynamics of feedback excitation may compound changes due to an altered composition of perisomatic inhibition.

The changes we describe interact with changes in excitatory neurons. One prominent change in the CA1 region is an increase in intrinsic bursting behavior [238–240]. This change is likely relevant for seizure initiation because spontaneously occurring burst discharges precede seizure-like activity in hippocampal slices [238]. Our results predict that synchronized bursting of pyramidal neurons would be less controlled by recurrent inhibition. This would hold true if the bursting is driven by rapid increases of CA3 activity.

In conclusion, we demonstrate multiple changes in interneurons involved in feedback inhibition and in their recruitment. These conspire to cause pronounced impairment of a canonical inhibitory motif in chronic epilepsy, which may be an important contributor to the generation and spread of aberrant activity.

CHAPTER 7

Summary and outlook

In this thesis, we have investigated different forms of connection weight changes in network models of neural circuits from a dynamical systems' perspective. We considered the ability of weight changes to shape network dynamics taking into account, for example, the stability and dimensionality of the dynamics and emergent phenomena such as synchronous activity. Our results provide novel insights for the roles of synaptic plasticity for learning, for the drift of neural representations and for epilepsy-associated changes of neural dynamics.

In Chapter 3 we have developed a novel scheme for dynamical supervised learning. We constructed networks that learn dynamically by pretraining their connection weights. Crucially and in contrast to previous approaches [33–46], our scheme does not depend on continuous target feedback after the dynamical learning has finished. We achieved this by stabilizing the network dynamics after learning by fixing the feedback of a context neuron. We demonstrated the feasibility of our approach by applying it to a set of example tasks, which are of comparable difficulty as those used to introduce FORCE learning [125]. Further, we studied the mechanisms and thoroughly investigated the properties of our dynamical learning.

Our scheme provides a novel candidate mechanism for learning in the brain when the usual paradigm of weight learning might be too slow and/or when similar tasks have been experienced before. While there is substantial evidence that the brain learns to learn [25, 129], there are rarely any experimental studies that directly consider learning without weight changes [28]. Our results indicate that dynamical supervised learning of complicated, even chaotic dynamics is possible. Further, previous work showed that dynamical reinforcement learning is possible [32, 130, 131]. We thus think that dynamical learning is an interesting field for future experimental studies. Theoretical studies on dynamical learning have so far been conducted in rather abstract neural networks. Bringing them closer to real neural networks is an important task for the future.

Besides being a model for biological neural networks, our scheme may also be useful for neuromorphic and physical reservoir computing, where it could replace the performance-limiting weight learning [49–53, 163]. For this, it would be advantageous if the capabilities of our approach could be expanded. This holds in particular for the learning of dynamics that differ considerably from the pretrained ones or are variable in a high-dimensional parameter. A starting point could be to adjust the network architecture, since recent work on similar networks indicate that a low-rank weight matrix improves the generalization performance [241].

While much research on weight learning is focused on activity-dependent update rules, in Chapter 4

we have considered WP learning, which utilizes random weight changes. We have shown that it performs better than NP if the task consists of trials that have a long duration and capture most of the task's content. Specifically, using Wick's theorem, we derived analytical expressions for the error dynamics of WP, NP and gradient descent for single-layer linear perceptrons performing a temporally extended, student-teacher task. We then demonstrated numerically that our analytical results qualitatively extend to more complex networks and standard biological and machine learning tasks. In fact, we find that WP's performance is more robust to deviations of the network architecture from single-layer linear networks compared to NP's.

Our results dispute the long-standing belief that NP is to be preferred over WP with regard to performance. NP is regularly used as a simple reference algorithm for other learning rules that are suggested for biologically plausible learning and are typically more complex [113, 196]. Our findings indicate that WP might often be the better choice. With regard to biological plausibility, WP is also attractive as recent experimental studies indicate strong, spontaneous weight changes [22–24]. Such weight changes may be caused by an implementation of WP in the brain.

Remarkably, we found that WP's performance does not suffer much from increasing the number of layers in the MNIST task. A natural next step would be to check how it performs for more complex tasks in deep networks with more than three layers. While it is not to be expected that WP could compete with gradient descent given our analytical and numerical findings, this would further elucidate WP's applicability for learning in the brain and learning in general.

In Chapter 5, we have shown how the combination of STDP and homeostatic normalization together with noisy spiking can give rise to drifting assemblies. To this end, we first used numerical simulations and analyzed the simulation results. Then, we constructed a simplified model of the weight dynamics, which showed that the neuron switching dynamics can be viewed as a random walk between meta-stable states with noise-induced transitions between them. It further showed that the inhomogeneous noise alone is sufficient to induce multistability.

Our model stands in profound contrast to previous models of memories, which assume static assemblies [67]. It accounts for the recent experimental observations that neural representations of memories are changing while behavior is stable [63–65]. It is based on synaptic plasticity mechanisms, which drive the assembly drift but also implement compensatory learning, thus keeping the assemblies and associated memories intact. For the future, it would be important to study how novel memories can be learned in our model. Imprinting novel assemblies into the network could provide a solution. Another interesting question is if other neural structures than distinct assemblies, such as sequences, can drift. This could be relevant, for example, for the learning of temporal visual sequences: In an experimental study it was shown that the neural representations of such sequences vanish in primary visual cortex [242], possibly due to ongoing synaptic plasticity as suggested by a modeling study [243]. It may be that the long-term storage of such sequences relies on the drift of the neural representations into other brain regions, where they persist.

Finally, in Chapter 6, we have considered the feedback inhibitory motif in epilepsy. We constructed an effective network model for feedback inhibition in CA1 based on experimental results from healthy and epileptic rats. It consists of an excitatory-inhibitory feedback loop with synapses that exhibit STP. We fitted its individual parts to the experimental data using a grid search in parameter space. Afterwards, we tested how it behaves under external stimulation mimicking input from CA3. We found that the epilepsy-associated changes appear to promote the transmission of burst-like activity from CA3 to other parts of the brain.

Our combined experimental and theoretical approach is the first to show that altered feedback

inhibition might be a crucial component for the spread and generation of epilepsy-associated activity. Experimentally, it would be interesting to check our model predictions directly by recording the output of pyramidal cells in CA1 in response to burst-like input. Theoretically, one could take our results on how model parameters change in epilepsy and use them in detailed models of SPW/Rs in CA1 (e.g. ref. [81]). This would allow comparing in detail how SPW/Rs change in models with how they change in experiments due to epilepsy [99].

Bibliography

[1] C. Klos et al., *Dynamical Learning of Dynamics*, Phys. Rev. Lett. **125** (2020) 088103.

[2] P. Züge, C. Klos, and R.-M. Memmesheimer, *Weight perturbation learning outperforms node perturbation on broad classes of temporally extended tasks*, 2021, bioRxiv: 2021.10.04.463055.

[3] Y. F. Kalle Kossio et al., *Drifting assemblies for persistent memory: Neuron transitions and unsupervised compensation*, PNAS **118** (2021).

[4] L. Pothmann et al., *Altered Dynamics of Canonical Feedback Inhibition Predicts Increased Burst Transmission in Chronic Epilepsy*, Journal of Neuroscience **39** (2019) 8998.

[5] E. R. Kandel et al., eds., *Principles of Neural Science*, 5th ed., Mcgraw-Hill, 2012.

[6] M. F. Bear, B. W. Connors, and M. A. Paradiso, *Neuroscience — Exploring the Brain*, 3rd ed., Wolters Kluwer, 2016.

[7] S. Herculano-Houzel, *The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost*, PNAS **109** (2012) 10661.

[8] M. E. J. Newman, *Networks*, 2nd ed., Oxford University Press, 2018.

[9] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, 1st ed., MIT Press, 2001.

[10] W. Gerstner et al., *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*, 1st ed., Cambridge University Press, 2014.

[11] M. E. J. Newman, *Resource Letter CS–1: Complex Systems*, American Journal of Physics **79** (2011) 800.

[12] J. J. Hopfield, *Neural networks and physical systems with emergent collective computational abilities*, PNAS **79** (1982) 2554.

[13] H. Sompolinsky, A. Crisanti, and H. J. Sommers, *Chaos in Random Neural Networks*, Physical Review Letters **61** (1988) 259.

[14] C. van Vreeswijk and H. Sompolinsky, *Chaos in Neuronal Networks with Balanced Excitatory and Inhibitory Activity*, Science **274** (1996) 1724.

[15] N. Brunel, *Dynamics of Sparsely Connected Networks of Excitatory and Inhibitory Spiking Neurons*, Journal of Computational Neuroscience **8** (2000) 183.

[16] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed.,
Cambridge University Press, 2008.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed.,
`http://www.deeplearningbook.org`, MIT Press, 2016.

[18] D. Hassabis et al., *Neuroscience-Inspired Artificial Intelligence*, Neuron **95** (2017) 245.

[19] T. P. Lillicrap et al., *Backpropagation and the brain*,
Nature Reviews Neuroscience **21** (2020) 335.

[20] D. L. K. Yamins and J. J. DiCarlo,
Using goal-driven deep learning models to understand sensory cortex,
Nature Neuroscience **19** (2016) 356.

[21] J. Sjöström and W. Gerstner, *Spike-timing dependent plasticity*,
Scholarpedia **5** (2010) 1362, revision #184913.

[22] H. Kasai et al., *Spine dynamics in the brain, mental disorders and artificial neural networks*,
Nature Reviews Neuroscience **22** (2021) 407.

[23] K. P. Berry and E. Nedivi, *Spine Dynamics: Are They All the Same?* Neuron **96** (2017) 43.

[24] N. E. Ziv and N. Brenner,
Synaptic Tenacity or Lack Thereof: Spontaneous Remodeling of Synapses,
Trends in Neurosciences **41** (2018) 89.

[25] D. A. Braun, C. Mehring, and D. M. Wolpert, *Structure learning in action*,
Behavioural Brain Research **206** (2010) 157.

[26] K. Oberauer, *Is Rehearsal an Effective Maintenance Strategy for Working Memory?*
Trends in Cognitive Sciences **23** (2019) 798.

[27] R. Froemke, D. Debanne, and G.-Q. Bi,
Temporal modulation of spike-timing-dependent plasticity,
Frontiers in Synaptic Neuroscience **2** (2010) 19.

[28] M. G. Perich, J. A. Gallego, and L. E. Miller,
A Neural Population Mechanism for Rapid Learning, Neuron **100** (2018) 964.

[29] K.-i. Funahashi and Y. Nakamura,
Approximation of dynamical systems by continuous time recurrent neural networks,
Neural Networks **6** (1993) 801.

[30] N. E. Cotter and P. R. Conwell, "Fixed-weight Networks Can Learn,"
1990 IJCNN International Joint Conference on Neural Networks, 1990 553.

[31] N. E. Cotter and P. R. Conwell, "Learning Algorithms and Fixed Dynamics,"
IJCNN-91-Seattle International Joint Conference on Neural Networks, 1991 799.

[32] J. X. Wang et al., *Prefrontal cortex as a meta-reinforcement learning system*,
Nature Neuroscience **21** (2018) 860.

[33] G. Bellec et al.,
Long short-term memory and learning-to-learn in networks of spiking neurons, 2018,
arXiv: `1803.09574 [cs.NE]`.

[34] L. Feldkamp, G. Puskorius, and P. Moore, "Adaptation from fixed weight dynamic networks," *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 1, 1996 155.

[35] L. Feldkamp, G. Puskorius, and P. Moore, *Adaptive behavior from fixed weight networks*, Information Sciences **98** (1997) 217.

[36] L. Feldkamp and G. Puskorius, "Fixed-weight controller for multiple systems," *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 2, 1997 773.

[37] A. Younger, P. Conwell, and N. Cotter, *Fixed-weight on-line learning*, IEEE Transactions on Neural Networks **10** (1999) 272.

[38] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to Learn Using Gradient Descent," *Artificial Neural Networks — ICANN 2001*, ed. by G. Dorffner, H. Bischof, and K. Hornik, Springer Berlin Heidelberg, 2001 87.

[39] A. Younger, S. Hochreiter, and P. Conwell, "Meta-learning with backpropagation," *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, vol. 3, 2001 2001.

[40] L. A. Feldkamp, D. V. Prokhorov, and T. M. Feldkamp, *Simple and conditioned adaptive behavior from Kalman filter trained recurrent networks*, Neural Networks **16** (2003) 683.

[41] R. Santiago, "Context discerning multifunction networks: reformulating fixed weight neural networks," *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 1, 2004 189.

[42] M. Lukoševičius, *Echo State Networks with Trained Feedbacks*, tech. rep. Technical Report No. 4, Jacobs University Bremen, 2007.

[43] A. Santoro et al., "Meta-Learning with Memory-Augmented Neural Networks," *Proceedings of The 33rd International Conference on Machine Learning*, ed. by M. F. Balcan and K. Q. Weinberger, vol. 48, Proceedings of Machine Learning Research, PMLR, 2016 1842.

[44] L. Feldkamp and G. Puskorius, "Training of robust neural controllers," *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, vol. 3, 1994 2754.

[45] L. Feldkamp and G. Puskorius, "Training controllers for robustness: multi-stream DEKF," *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 4, 1994 2377.

[46] M. Oubbati and G. Palm, *A neural framework for adaptive robot control*, Neural Computing and Applications **19** (2010) 103.

[47] J. Pathak et al., *Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach*, Phys. Rev. Lett. **120** (2018) 024102.

[48] R. S. Zimmermann and U. Parlitz, *Observing Spatio-temporal Dynamics of Excitable Media Using Reservoir Computing*, Chaos **28** (2018) 043118.

[49] G. Tanaka et al., *Recent advances in physical reservoir computing: A review*, Neural Networks **115** (2019) 100.

[50] M. Rafayelyan et al., *Large-Scale Optical Reservoir Computing for Spatiotemporal Chaotic Systems Prediction*, Phys. Rev. X **10** (2020) 041037.

[51] F. Duport et al., *Fully analogue photonic reservoir computer*, Scientific Reports **6** (2016) 22381.

[52] A. N. Tait et al., *Neuromorphic photonic networks using silicon photonic weight banks*, Scientific Reports **7** (2017) 7430.

[53] P. Antonik, M. Haelterman, and S. Massar, *Brain-Inspired Photonic Signal Processor for Generating Periodic Patterns and Emulating Chaotic Systems*, Phys. Rev. Applied **7** (2017) 054014.

[54] A. Dembo and T. Kailath, *Model-free distributed learning*, IEEE Transactions on Neural Networks **1** (1990) 58.

[55] G. Cauwenberghs, "A Fast Stochastic Error-Descent Algorithm for Supervised Learning and Optimization," *Advances in Neural Information Processing Systems*, ed. by S. Hanson, J. Cowan, and C. Giles, vol. 5, 1993.

[56] J. Werfel, X. Xie, and H. S. Seung, *Learning Curves for Stochastic Gradient Descent in Linear Feedforward Networks*, Neural Computation **17** (2005) 2699.

[57] I. R. Fiete and H. S. Seung, *Gradient Learning in Spiking Neural Networks by Dynamic Perturbation of Conductances*, Phys. Rev. Lett. **97** (2006) 048104.

[58] R. Legenstein et al., *A Reward-Modulated Hebbian Learning Rule Can Explain Experimentally Observed Network Reorganization in a Brain Control Task*, Journal of Neuroscience **30** (2010) 8400.

[59] G. M. Hoerzer, R. Legenstein, and W. Maass, *Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning*, Cerebral Cortex **24** (2012) 677.

[60] T. Miconi, *Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks*, eLife **6** (2017) e20899.

[61] I. R. Fiete, M. S. Fee, and H. S. Seung, *Model of Birdsong Learning Based on Gradient Estimation by Dynamic Perturbation of Neural Conductances*, Journal of Neurophysiology **98** (2007) 2038.

[62] H. Saito et al., *Statistical mechanics of structural and temporal credit assignment effects on learning in neural networks*, Phys. Rev. E **83** (2011) 051125.

[63] L. A. DeNardo et al., *Temporal evolution of cortical ensembles promoting remote memory retrieval*, Nature Neuroscience **22** (2019) 460.

[64] C. Clopath et al., *Variance and invariance of neuronal long-term representations*, Philosophical Transactions of the Royal Society B: Biological Sciences **372** (2017) 20160161.

[65] M. E. Rule, T. O'Leary, and C. D. Harvey, *Causes and consequences of representational drift*, Current Opinion in Neurobiology **58** (2019) 141.

[66] G. Buzsáki, *Neural Syntax: Cell Assemblies, Synapsembles, and Readers*, Neuron **68** (2010) 362.

[67] G. Mongillo, S. Rumpel, and Y. Loewenstein, *Intrinsic volatility of synaptic connections — a challenge to the synaptic trace theory of memory*, Current Opinion in Neurobiology **46** (2017) 7.

[68] C. W. Gardiner, *Handbook of Stochastic Methods*, 3rd ed., Springer, 2004.

[69] L. Arnold, W. Horsthemke, and R. Lefever, *White and coloured external noise and transition phenomena in nonlinear systems*, Zeitschrift für Physik B Condensed Matter and Quanta **29** (1978) 367.

[70] W. Horsthemke and R. Lefever, *Noise-Induced Transitions*, Springer, 1984.

[71] G. Jetschke, *Mathematik der Selbstorganisation*, VEB Deutscher Verlag der Wissenschaften, 1989.

[72] T. Biancalani, L. Dyson, and A. J. McKane, *Noise-Induced Bistable States and Their Mean Switching Time in Foraging Colonies*, Physical Review Letters **112** (2014).

[73] R. S. Fisher et al., *Epileptic Seizures and Epilepsy: Definitions Proposed by the International League Against Epilepsy (ILAE) and the International Bureau for Epilepsy (IBE)*, Epilepsia **46** (2005) 470.

[74] P. Andersen et al., eds., *The Hippocampus Book*, 1st ed., Oxford University Press, 2007.

[75] G. Buzsáki, *Hippocampal sharp wave-ripple: A cognitive biomarker for episodic memory and planning*, Hippocampus **25** (2015) 1073.

[76] V. André et al., *Alterations of hippocampal GABAergic system contribute to development of spontaneous recurrent seizures in the rat lithium-pilocarpine model of temporal lobe epilepsy*, Hippocampus **11** (2001) 452.

[77] M. S. Wyeth et al., *Selective Reduction of Cholecystokinin-Positive Basket Cell Innervation in a Model of Temporal Lobe Epilepsy*, Journal of Neuroscience **30** (2010) 8993.

[78] Leonie Pothmann, *Changes of inhibitory micronetworks in the epileptic hippocampus and their response to anticonvulsant drugs*, PhD thesis: Rheinische Friedrich-Wilhelms-Universität Bonn, 2015, URL: `https://hdl.handle.net/20.500.11811/6555`.

[79] T. Freund and S. Kali, *Interneurons*, Scholarpedia **3** (2008) 4720, revision #89023.

[80] P. Somogyi and T. Klausberger, *Defined types of cortical interneurone structure space and spike timing in the hippocampus*, The Journal of Physiology **562** (2005) 9.

[81] W. Braun and R.-M. Memmesheimer, *High-frequency oscillations and replay in a two-population model of hippocampal region CA1*, 2021, bioRxiv: 2021.06.08.447523.

[82] M. J. Bezaire and I. Soltesz, *Quantitative assessment of CA1 local circuits: Knowledge base for interneuron-pyramidal cell connectivity*, Hippocampus **23** (2013) 751.

[83] H. Seung,
Learning in Spiking Neural Networks by Reinforcement of Stochastic Synaptic Transmission,
Neuron **40** (2003) 1063.

[84] R. K. Mishra et al., *Symmetric spike timing-dependent plasticity at CA3–CA3 synapses optimizes storage and recall in autoassociative networks*,
Nature Communications **7** (2016) 11552.

[85] Y. Humeau and D. Choquet, *The next generation of approaches to investigate the link between synaptic plasticity and learning*, Nature Neuroscience **22** (2019) 1536.

[86] M. Tsodyks and S. Wu, *Short-term synaptic plasticity*,
Scholarpedia **8** (2013) 3153, revision #182521.

[87] G.-q. Bi and M.-m. Poo, *Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type*,
Journal of Neuroscience **18** (1998) 10464.

[88] G. Turrigiano, *Homeostatic Synaptic Plasticity: Local and Global Mechanisms for Stabilizing Neuronal Function*, Cold Spring Harbor Perspectives in Biology **4** (2012) a005736.

[89] M. Letellier et al., *Differential role of pre- and postsynaptic neurons in the activity-dependent control of synaptic strengths across dendrites*, PLOS Biology **17** (2019) e2006223.

[90] S. Rumpel and J. Triesch, *The dynamic connectome*, e-Neuroforum **7** (3 2016) 48.

[91] R. J. Sutherland et al., *Has multiple trace theory been refuted?* Hippocampus **30** (2020) 842.

[92] A. Renart et al., *The Asynchronous State in Cortical Circuits*, Science **327** (2010) 587.

[93] A. Kumar et al., *The High-Conductance State of Cortical Networks*,
Neural Computation **20** (2008) 1.

[94] Y. Ahmadian and K. D. Miller, *What is the dynamical regime of cerebral cortex?* 2019,
arXiv: 1908.10101 [q-bio.NC].

[95] G. Buzsáki and A. Draguhn, *Neuronal Oscillations in Cortical Networks*,
Science **304** (2004) 1926.

[96] G. Foffani et al., *Reduced Spike-Timing Reliability Correlates with the Emergence of Fast Ripples in the Rat Epileptic Hippocampus*, Neuron **55** (2007) 930.

[97] P. Jiruska et al.,
Epileptic high-frequency network activity in a model of non-lesional temporal lobe epilepsy,
Brain **133** (2010) 1380.

[98] J. M. Ibarz et al., *Emergent Dynamics of Fast Ripples in the Epileptic Hippocampus*,
Journal of Neuroscience **30** (2010) 16249.

[99] M. Valero et al., *Mechanisms for Selective Single-Cell Reactivation during Offline Sharp-Wave Ripples and Their Distortion by Fast Ripples*, Neuron **94** (2017) 1234.

[100] M. Jazayeri and S. Ostojic, *Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity*, 2021, arXiv: 2107.04084 [q-bio.NC].

[101] S. Chung and L. F. Abbott, *Neural population geometry: An approach for understanding biological and artificial neural networks*, 2021, arXiv: 2104.07059 [q-bio.NC].

[102] P. Gao et al., *A theory of multineuronal dimensionality, dynamics and measurement*, 2017, bioRxiv: 214262.

[103] J. A. Gallego et al., *Neural Manifolds for the Control of Movement*, Neuron **94** (2017) 978.

[104] C. J. Cueva et al., *Low-dimensional dynamics for working memory and time encoding*, PNAS **117** (2020) 23021.

[105] A. A. Russo et al., *Neural Trajectories in the Supplementary Motor Area and Motor Cortex Exhibit Distinct Geometries, Compatible with Different Classes of Computation*, Neuron **107** (2020) 745.

[106] H. Markram, Y. Wang, and M. Tsodyks, *Differential signaling via the same axon of neocortical pyramidal neurons*, PNAS **95** (1998) 5323.

[107] C. Tetzlaff et al., *Synaptic Scaling in Combination with Many Generic Plasticity Mechanisms Stabilizes Circuit Connectivity*, Frontiers in Computational Neuroscience **5** (2011) 47.

[108] F. Zenke, W. Gerstner, and S. Ganguli, *The temporal paradox of Hebbian learning and homeostatic plasticity*, Current Opinion in Neurobiology **43** (2017) 166.

[109] C. van Vreeswijk and H. Sompolinsky, *Chaotic Balanced State in a Model of Cortical Circuits*, Neural Computation **10** (1998) 1321.

[110] G. Hennequin, T. P. Vogels, and W. Gerstner, *Optimal Control of Transient Dynamics in Balanced Networks Supports Generation of Complex Movements*, Neuron **82** (2014) 1394.

[111] J. P. Stroud et al., *Motor primitives in space and time via targeted gain modulation in cortical networks*, Nature Neuroscience **21** (2018) 1774.

[112] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, Nature **323** (1986) 533.

[113] T. P. Lillicrap et al., *Random synaptic feedback weights support error backpropagation for deep learning*, Nature Communications **7** (2016) 13276.

[114] F. Zenke and S. Ganguli, *SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks*, Neural Computation **30** (2018) 1514.

[115] J. M. Murray, *Local online learning in recurrent networks with random feedback*, eLife **8** (2019) e43299.

[116] G. Bellec et al., *A solution to the learning dilemma for recurrent networks of spiking neurons*, Nature Communications **11** (2020) 3625.

[117] M. Lukoševičius and H. Jaeger,
Reservoir computing approaches to recurrent neural network training,
Computer Science Review **3** (2009) 127.

[118] H. Jaeger, *Echo state network*, Scholarpedia **2** (2007) 2330, revision #196567.

[119] T. M. Cover, *Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition*,
IEEE Transactions on Electronic Computers **EC-14** (1965) 326.

[120] W. Maass, T. Natschläger, and H. Markram, *Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations*,
Neural Computation **14** (2002) 2531.

[121] H. Jaeger, *The "echo state" approach to analysing and training recurrent neural networks*,
tech. rep. GMD Report 148,
German National Research Center for Information Technology, 2001.

[122] H. Jaeger and H. Haas, *Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication*, Science **304** (2004) 78.

[123] D. V. Buonomano and M. M. Merzenich, *Temporal Information Transformed into a Spatial Code by a Neural Network with Realistic Properties*, Science **267** (1995) 1028.

[124] P. F. Dominey, *Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning*, Biological Cybernetics **73** (1995) 265.

[125] D. Sussillo and L. Abbott,
Generating Coherent Patterns of Activity from Chaotic Neural Networks,
Neuron **63** (2009) 544.

[126] A. Lazar, G. Pipa, and J. Triesch, *SORN: a self-organizing recurrent neural network*,
Frontiers in Computational Neuroscience **3** (2009) 23.

[127] S. Thrun and L. Pratt, eds., *Learning to Learn*, 1st ed., Springer US, 1998.

[128] J. Vanschoren, *Meta-Learning: A Survey*, 2018, arXiv: 1810.03548 [cs.LG].

[129] B. J. Lansdell and K. P. Kording, *Towards learning-to-learn*,
Current Opinion in Behavioral Sciences **29** (2019) 45.

[130] Y. Duan et al., *RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning*, 2016,
arXiv: 1611.02779 [cs.AI].

[131] A. Nagabandi et al., *Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning*, 2019, arXiv: 1803.11347 [cs.LG].

[132] H. Jaeger et al.,
Optimization and Applications of Echo State Networks with Leaky-integrator Neurons,
Neural Networks **20** (2007) 335.

[133] D. Sussillo and O. Barak, *Opening the Black Box: Low-dimensional Dynamics in High-dimensional Recurrent Neural Networks*, Neural Comput. **25** (2013) 626.

[134] M. Lukoševičius, H. Jaeger, and B. Schrauwen, *Reservoir Computing Trends*,
KI - Künstliche Intelligenz **26** (2012) 365.

[135] M. Y. Ismail and J. C. Principe,
"Equivalence between RLS Algorithms and the Ridge Regression Technique,"
Proc. Systems and Computers Conf. Record of The Thirtieth Asilomar Conf. Signals, 1996 1083.

[136] V. Mante et al.,
Context-dependent Computation by Recurrent Dynamics in Prefrontal Cortex.,
Nature **503** (2013) 78.

[137] L. F. Abbott, B. DePasquale, and R.-M. Memmesheimer,
Building functional networks of spiking model neurons, Nature Neuroscience **19** (2016) 350.

[138] L. Abbott, K. Rajan, and H. Sompolinsky,
"Interactions between Intrinsic and Stimulus-Evoked Activity in Recurrent Neural Networks,"
The Dynamic Brain: An Exploration of Neuronal Variability and Its Functional Significance, Oxford University Press, 2011.

[139] M. Oubbati, P. Levi, and M. Schanz,
"Meta-Learning for Adaptive Identification of Non-Linear Dynamical Systems,"
Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005. 2005 473.

[140] H. Jaeger and D. Eck,
"Can't Get You Out of My Head: A Connectionist Model of Cyclic Rehearsal,"
Modeling Communication with Robots and Virtual Humans,
ed. by I. Wachsmuth and G. Knoblich, Springer Berlin Heidelberg, 2008 310.

[141] F. wyffels et al., *Frequency modulation of large oscillatory neural networks*,
Biological Cybernetics **108** (2014) 145.

[142] K. J. Boström et al.,
Model for a flexible motor memory based on a self-active recurrent neural network,
Human Movement Science **32** (2013) 880.

[143] M. I. Jordan and D. E. Rumelhart,
Forward Models: Supervised Learning with a Distal Teacher,
Cognitive Science **16** (1992) 307.

[144] M. Westover, C. Eliasmith, and C. H. Anderson,
Linearly decodable functions from neural population codes,
Neurocomputing **44-46** (2002) 691.

[145] B. DePasquale et al., *Full-force: A Target-based Method for Training Recurrent Networks*,
PLoS One **13** (2018) e0191527.

[146] *Array programming with NumPy*, Nature **585** (2020) 357.

[147] https://github.com/chklos/dynamical-learning.

[148] O. Schütze et al., *Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multiobjective Optimization.*,
IEEE Trans. Evolutionary Computation **16** (2012) 504.

[149] H. Jaeger, *Controlling Recurrent Neural Networks by Conceptors*, 2017,
arXiv: `1403.3369 [cs.NE]`.

[150] H. Jaeger, *Using Conceptors to Manage Neural Long-Term Memories for Temporal Patterns*, Journal of Machine Learning Research **18** (2017) 1.

[151] M. Jabri and B. Flower, *Weight perturbation: an optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks*, IEEE Transactions on Neural Networks **3** (1992) 154.

[152] C. Beer and O. Barak, *One step back, two steps forward: interference and learning in recurrent neural networks*, 2019, arXiv: 1805.09603 [q-bio.NC].

[153] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *Proceedings of the 3rd International Conference on Learning Representations*, vol. 3, 2015.

[154] R. L. Redondo and R. G. M. Morris, *Making memories last: the synaptic tagging and capture hypothesis*, Nature Reviews Neuroscience **12** (2011) 17.

[155] Z. Yu et al., *CaMKII activation supports reward-based neural network optimization through Hamiltonian sampling*, 2018, arXiv: 1606.00157 [cs.NE].

[156] W. C. Abraham, *Metaplasticity: tuning synapses and networks for plasticity*, Nat. Rev. Neurosci. **9** (2008) 387.

[157] G. Puskorius and L. Feldkamp, *Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks*, IEEE Transactions on Neural Networks **5** (1994) 279.

[158] E. D. Sontag, "Neural Nets As Systems Models and Controllers," *Proc. Seventh Yale Workshop on Adaptive and Learning Systems*, 1992 73.

[159] K.-K. K. Kim, E. R. Patrón, and R. D. Braatz, *Standard Representation and Unified Stability Analysis for Dynamic Artificial Neural Network Models*, Neural Networks **98** (2018) 251.

[160] S. Lim and M. S. Goldman, *Balanced Cortical Microcircuitry for Maintaining Information in Working Memory*, Nat. Neurosci. **16** (2013) 1306.

[161] W. Maass, P. Joshi, and E. D. Sontag, *Computational Aspects of Feedback in Neural Circuits*, PLOS Computational Biology **3** (2007) 1.

[162] R. Pascanu and H. Jaeger, *A neurodynamical model for working memory*, Neural Networks **24** (2011) 199.

[163] E. Chicca et al., *Neuromorphic Electronic Circuits for Building Autonomous Cognitive Systems*, Proceedings of the IEEE **102** (2014) 1367.

[164] D. Thalmeier et al., *Learning Universal Computations with Spikes*, PLOS Computational Biology **12** (2016) 1.

[165] C. D. Schuman et al., *A Survey of Neuromorphic Computing and Neural Networks in Hardware*, 2017, arXiv: 1705.06963 [cs.NE].

[166] K. K. Sreenivasan and M. D'Esposito, *The what, where and how of delay activity*, Nature Reviews Neuroscience **20** (2019) 466.

[167] R. J. Williams, *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, Machine Learning **8** (1992) 229.

[168] D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan, *Principles of sensorimotor learning*, Nature Reviews Neuroscience **12** (2011) 739.

[169] R. Mooney, J. Prather, and T. Roberts, "Neurophysiology of birdsong learning," *Learning and Memory*, Elsevier, 2007 441.

[170] N. Y. Masse et al., *Circuit mechanisms for the maintenance and manipulation of information in working memory*, Nature Neuroscience **22** (2019) 1159.

[171] Y. LeCun et al., *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998) 2278.

[172] K. Doya and T. J. Sejnowski, "A Computational Model of Birdsong Learning by Auditory Experience and Auditory Feedback," *Central Auditory Processing and Neural Modeling*, Springer US, 1998 77.

[173] B. Widrow and M. Lehr, *30 years of adaptive neural networks: perceptron, Madaline, and backpropagation*, Proceedings of the IEEE **78** (1990) 1415.

[174] M. Helias and D. Dahmen, eds., *Statistical Field Theory for Neural Networks*, 1st ed., Springer, 2020.

[175] R. Kawai et al., *Motor Cortex Is Required for Learning but Not for Executing a Motor Skill*, Neuron **86** (2015) 800.

[176] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in Neural Information Processing Systems*, ed. by H. Wallach et al., vol. 32, 2019 8024.

[177] A. Destexhe, M. Rudolph, and D. Paré, *The high-conductance state of neocortical neurons in vivo*, Nature Reviews Neuroscience **4** (2003) 739.

[178] J. D. Murray et al., *A hierarchy of intrinsic timescales across primate cortex*, Nature Neuroscience **17** (2014) 1661.

[179] R. H. R. Hahnloser, A. A. Kozhevnikov, and M. S. Fee, *An ultra-sparse code underlies the generation of neural sequences in a songbird*, Nature **419** (2002) 65.

[180] T. Teşileanu, B. Ölveczky, and V. Balasubramanian, *Rules and mechanisms for efficient two-stage learning in neural circuits*, eLife **6** (2017) e20944.

[181] J. M. Murray and G. S. Escola, *Learning multiple variable-speed sequences in striatum via cortical tutoring*, eLife **6** (2017) e26084.

[182] A. K. Dhawale, M. A. Smith, and B. P. Ölveczky, *The Role of Variability in Motor Learning*, Annual Review of Neuroscience **40** (2017) 479.

[183] S. Cheng, *The CRISP theory of hippocampal function in episodic memory*, Frontiers in Neural Circuits **7** (2013) 88.

[184] A. Maes, M. Barahona, and C. Clopath, *Learning spatiotemporal signals using a recurrent spiking network that discretizes time*, PLOS Computational Biology **16** (2020) 1.

[185] Ł. Kuśmierz, T. Isomura, and T. Toyoizumi, *Learning with three factors: modulating Hebbian plasticity with errors*, Current Opinion in Neurobiology **46** (2017) 170.

[186] Q. Zhou, H. W. Tao, and M.-m. Poo, *Reversal and Stabilization of Synaptic Modifications in a Developing Visual System*, Science **300** (2003) 1953.

[187] P. Suszyński and P. Wawrzyński, "Learning population of spiking neural networks with perturbation of conductances," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013.

[188] T. Cho et al., *Node perturbation learning without noiseless baseline*, Neural Networks **24** (2011) 267.

[189] P. Mazzoni, R. A. Andersen, and M. I. Jordan, *A more biologically plausible learning rule for neural networks.*, PNAS **88** (1991) 4433.

[190] R. Darshan, A. Leblois, and D. Hansel, *Interference and Shaping in Sensorimotor Adaptations with Rewards*, PLOS Computational Biology **10** (2014) 1.

[191] E. Vasilaki et al., *Learning flexible sensori-motor mappings in a complex network*, Biological Cybernetics **100** (2009) 147.

[192] K. Takiyama and M. Okada, *Maximization of Learning Speed in the Motor Cortex Due to Neuronal Redundancy*, PLOS Computational Biology **8** (2012) 1.

[193] M. N. Abdelghani, T. P. Lillicrap, and D. B. Tweed, *Sensitivity Derivatives for Flexible Sensorimotor Learning*, Neural Computation **20** (2008) 2085.

[194] B. B. Vladimirskiy et al., *Stimulus sampling as an exploration mechanism for fast reinforcement learning*, Biological Cybernetics **100** (2009) 319.

[195] J. Friedrich, R. Urbanczik, and W. Senn, *Code-specific learning rules improve action selection by populations of spiking neurons*, International Journal of Neural Systems **24** (2014) 1450002.

[196] A. Payeur et al., *Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits*, Nature Neuroscience **24** (2021) 1010.

[197] K. Doya, "Bifurcations in the learning of recurrent neural networks," *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, vol. 6, 1992 2777.

[198] A. R. Chambers and S. Rumpel, *A stable brain from unstable components: Emerging concepts and implications for neural computation*, Neuroscience **357** (2017) 172.

[199] U. Rokni et al., *Motor Learning with Unstable Neural Representations*, Neuron **54** (2007) 653.

[200] F. Zenke, E. Agnes, and W. Gerstner, *Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks*, Nature Communications **6** (2015) 6922.

[201] A. Litwin-Kumar and B. Doiron, *Formation and maintenance of neuronal assemblies through synaptic plasticity.*, Nat. Commun. **5** (2014) 5319.

[202] N. R. Tannenbaum and Y. Burak, *Shaping Neural Circuits by High Order Synaptic Interactions*, PLOS Comp. Biol. **12** (2016) e1005056.

[203] L. Montangie, C. Miehl, and J. Gjorgjieva, *Autonomous emergence of connectivity assemblies via spike triplet interactions*, PLOS Computational Biology **16** (2020) 1.

[204] M. Stimberg, D. F. M. Goodman, and R. Brette, *Brian 2, an intuitive and efficient neural simulator*, eLife **8** (2019).

[205] https://github.com/fkalle/drifting-assemblies.

[206] V. D. Blondel et al., *Fast unfolding of communities in large networks*, J. Stat. Mech.: Theory Exp. **2008** (2008) P10008.

[207] R. LaPlante et al., *bctpy v0.5.2: Brain Connectivity Toolbox for Python*, URL: `https://github.com/aestrivex/bctpy`.

[208] G. K. Ocker and B. Doiron, *Training and Spontaneous Reinforcement of Neuronal Assemblies by Spike Timing Plasticity*, Cereb. Cortex **29** (2018) 937.

[209] M. J. Fauth and M. C. van Rossum, *Self-organized reactivation maintains and reinforces memories despite synaptic turnover*, eLife **8** (2019).

[210] E. Bienenstock, L. Cooper, and P. Munro, *Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex*, The Journal of Neuroscience **2** (1982) 32.

[211] I. R. Fiete et al., *Spike-time-dependent plasticity and heterosynaptic competition organize networks to produce long scale-free sequences of neural activity.*, Neuron **65** (2010) 563.

[212] D. Acker, S. Paradis, and P. Miller, *Stable memory and computation in randomly rewiring neural networks*, J. Neurophysiol. **122** (2019) 66.

[213] J. Humble et al., *Intrinsic Spine Dynamics Are Critical for Recurrent Network Learning in Models With and Without Autism Spectrum Disorder*, Front. Comput. Neurosci. **13** (2019).

[214] R. Ajemian et al., *A theory for how sensorimotor skills are learned and retained in noisy and nonstationary neural circuits*, PNAS **110** (2013) E5078.

[215] D. Kappel et al., *A Dynamic Connectome Supports the Emergence of Stable Computational Function of Neural Circuits through Reward-Based Learning*, eneuro **5** (2018) 0301.

[216] G. Mongillo, S. Rumpel, and Y. Loewenstein, *Inhibitory connectivity defines the realm of excitatory plasticity*, Nat. Neurosci. **21** (2018) 1463.

[217] L. Susman, N. Brenner, and O. Barak, *Stable memory with unstable synapses*, Nat. Commun. **10** (2019).

[218] M. Gillett, U. Pereira, and N. Brunel, *Characteristics of sequential activity in networks with temporally asymmetric Hebbian learning*, PNAS **117** (2020) 29948.

[219] M. A. Triplett, L. Avitan, and G. J. Goodhill, *Emergence of spontaneous assembly activity in developing neural networks without afferent input*, PLOS Comp. Biol. **14** (2018) e1006421.

[220] N. Hiratani and T. Fukai, *Interplay between Short- and Long-Term Plasticity in Cell-Assembly Formation*, PLOS ONE **9** (2014) 1.

[221] M. E. Rule et al., *Stable task information from an unstable neural population*, eLife **9** (2020) e51121.

[222] T. Pietri et al., *The Emergence of the Spatial Structure of Tectal Spontaneous Activity Is Independent of Visual Inputs*, Cell Reports **19** (2017) 939.

[223] T. Freund and G. Buzsáki, *Interneurons of the hippocampus*, Hippocampus **6** (1996) 347.

[224] R. Miles, *Synaptic excitation of inhibitory cells by single CA3 hippocampal pyramidal cells of the guinea-pig in vitro.*, The Journal of Physiology **428** (1990) 61.

[225] C. W. Ang, G. C. Carlson, and D. A. Coulter, *Hippocampal CA1 Circuitry Dynamically Gates Direct Cortical Inputs Preferentially at Theta Frequencies*, Journal of Neuroscience **25** (2005) 9567.

[226] F. Pouille and M. Scanziani, *Routing of spike series by dynamic circuits in the hippocampus*, Nature **429** (2004) 717.

[227] L. Pothmann et al., *Function of Inhibitory Micronetworks Is Spared by Na+ Channel-Acting Anticonvulsant Drugs*, Journal of Neuroscience **34** (2014) 9720.

[228] N. Spruston, P. Jonas, and B. Sakmann, *Dendritic glutamate receptor channels in rat hippocampal CA3 and CA1 pyramidal neurons.*, The Journal of Physiology **482** (1995) 325.

[229] M. A. Smith, G. C. R. Ellis-Davies, and J. C. Magee, *Mechanism of the distance-dependent scaling of Schaffer collateral synapses in rat CA1 pyramidal neurons*, The Journal of Physiology **548** (2003) 245.

[230] A. Ylinen et al., *Intracellular correlates of hippocampal theta rhythm in identified pyramidal cells, granule cells, and basket cells*, Hippocampus **5** (1995) 78.

[231] A. Oliva et al., *Origin of Gamma Frequency Power during Hippocampal Sharp-Wave Ripples*, Cell Reports **25** (2018) 1693.

[232] C. Müller et al.,
Inhibitory Control of Linear and Supralinear Dendritic Excitation in CA1 Pyramidal Neurons, Neuron **75** (2012) 851.

[233] G. Girardeau et al., *Selective suppression of hippocampal ripples impairs spatial memory*, Nature Neuroscience **12** (2009) 1222.

[234] L. Wittner et al., *Surviving CA1 pyramidal cells receive intact perisomatic inhibitory input in the human epileptic hippocampus*, Brain **128** (2004) 138.

[235] S. Williams, P. Vachon, and J.-C. Lacaille,
Monosynaptic GABA-mediated inhibitory postsynaptic potentials in ca1 pyramidal cells of hyperexcitable hippocampal slices from kainic acid-treated rats, Neuroscience **52** (1993) 541.

[236] M. Esclapez et al., *Operative GABAergic inhibition in hippocampal CA1 pyramidal neurons in experimental epilepsy*, PNAS **94** (1997) 12151.

[237] R. Cossart et al.,
Dendritic but not somatic GABAergic inhibition is decreased in experimental epilepsy, Nature Neuroscience **4** (2001) 52.

[238] H. Su et al., *Upregulation of a T-Type Ca2+ Channel Causes a Long-Lasting Modification of Neuronal Firing Mode after Status Epilepticus*, Journal of Neuroscience **22** (2002) 3645.

[239] Y. Yaari, C. Yue, and H. Su, *Recruitment of apical dendritic T-type Ca2+ channels by backpropagating spikes underlies de novo intrinsic bursting in hippocampal epileptogenesis*, The Journal of Physiology **580** (2007) 435.

[240] A. J. Becker et al., *Transcriptional Upregulation of Cav3.2 Mediates Epileptogenesis in the Pilocarpine Model of Epilepsy*, Journal of Neuroscience **28** (2008) 13341.

[241] M. Beiran et al.,
Parametric control of flexible timing through low-dimensional neural manifolds, 2021, bioRxiv: 2021.11.08.467806.

[242] S. Xu et al., *Activity recall in a visual cortical ensemble*, Nature Neuroscience **15** (2012) 449.

[243] C. Klos, D. Miner, and J. Triesch, *Bridging structure and function: A model of sequence learning and prediction in primary visual cortex*, PLOS Computational Biology **14** (2018) 1.

Acknowledgments

First and foremost I would like to thank my advisor Raoul-Martin Memmesheimer for the opportunity to do research in such an interesting field of study. I am grateful for his guidance and support and for always having time to provide feedback on my research. In addition, he gave me the opportunity to attend various conferences and a summer school, which have helped me to broaden my knowledge on theoretical neuroscience. Further, I would like to thank Ulf-G. Meißner for agreeing to be the second referee of my thesis.

Also, I am grateful to Heinz Beck, Oliver Braganza and Leonie Pothmann for the collaborative work and the accompanying discussions that led to Chapter 6.

I would like to thank all current and former members of the Neural Network Dynamics and Computation group for the fruitful collaboration, the pleasant working atmosphere and the common activities unrelated to science. In particular, I am grateful to Yaroslav Felipe Kalle Kossio for the productive collaboration on the projects that led to Chapters 3 and 5. For the collaboration on the same projects and for proofreading parts of this thesis, I would like to thank Sven Goedeke. For the cooperation on the project that led to Chapter 4 and for proofreading Chapter 4, I want to thank Paul Züge. Further, I would like to thank Aditya Gilra for his help on the work that led to Chapter 3 and Paul Manz for being an ejoyable office mate and for many valuable discussions. Additionally, I am grateful to Simon Altrogge and Fabian Pallasdies for taking over my teaching duties when I had no time and to Valèria Ribelles Pérez, whom I had the pleasure to supervise during her Bachelor's project. I am especially thankful to Wilhelm Braun, for the shared activities outside the university, in particular the successful pub quizzes and the regular runs along the Rhine, which provided a welcome balance to the daily research work.

Furthermore, I would like to thank the members of the Institute of Genetics for the welcoming atmosphere, when our group, including myself, moved to Bonn.

For their enduring friendship despite diverging life paths, I would like to thank Alexander Brandt, Yannik Haber and Patrick Kramer, whom I have known since high-school or before.

Finally, I would like to thank my family. I am grateful to my brother and his wife, my niece and my nephew and, above all, my parents for their constant support and encouragement in all matters.