Master of Science in Engineering with Specialisation in Data Science

Master Thesis

On Improving Pattern Recognition in Deep Learning Systems through Self-Organization

Incorporating Findings from Neuroscience about Brain Functionality into Modern Deep Learning Architectures

Pascal Sager

September 5, 2022

Zurich University of Applied Sciences

Dedicated to the dedicated.

– Pascal Sager

"Max Planck said, 'Science progresses one funeral at a time.'
The future depends on some graduate student who is deeply
suspicious of everything I have said."

– Geoffrey Hinton, University of Toronto, 2017.

Zusammenfassung

Abstract

Preface

This thesis is the last step stone before I will hold the title “Master of Science”. To me science means the systematic analysis of the real or virtual world through observations and experiments as well as the further development of existing technology. I am lucky enough to be able to apply the knowledge and methodologies I learned during my studies to research projects at the Centre for Artificial Intelligence (CAI) of the Zurich University of Applied Sciences (ZHAW). I was mentored and supported during my studies by Prof. Dr. Thilo Stadelmann. He uses to say (also in accordance with his blog-post) “Great methodology delivers great theses”. It is always desirable to have an excellent outcome such as a system that can execute a task and thereby achieves or even overcomes state-of-the-art performance. However, in my opinion, it is equally or even more important to reason why and how something works, to justify choices, and to show limitations. I wrote my Thesis with these thoughts in mind and hope that the readers are able to follow my reasoning.

In the Introduction section, the fundamentals of Deep Learning and its limitations is described. Afterwards, it is motivated why methodologies inspired by neuroscience could overcome these limitations. This thesis aims at a target audience with a background in Deep Learning. Consequently, the concepts of Deep Learning are only roughly described. Since neurocomputing may be rather unknown to the target audience, a more extensive overview about this field is given.

I would like to thank a couple of colleagues and friends. First I think of my mentor Prof. Dr. Thilo Stadelmann who got me excited about AI years ago and later introduced me to research. He always encouraged creative ideas and helped to link different topics to address problems with methodologies from other fields. Thanks to his support, help, and guidance, I have grown personally as well as professionally. Further thanks go to Dr. Jan Deriu. Especially at the beginning of my thesis, he steered my thoughts in one direction. His unconventional thinking has led to the questioning of many methods that have stood the test of time for decades (this was also the inspiration for Geoffrey Hinton’s quote on the page before, although I wouldn’t presume to say that this thesis will change the future). To Prof. Dr. Christoph von der Malsburg for his seemingly endless patience in introducing me to neuroscience. He could build the bridge between the two diverging fields of Deep Learning and Neuroscience, serving as an inspiration for various new ideas.

The biggest thanks, however, goes to my family, who made this journey possible for me. My parents, who supported and encouraged me in every way. My younger brother who inspired me to study. My wife and son, who have been understanding and supportive and have always been the perfect counterbalance to the daily routine of studying. Without the support of my family, I would never have been able to embark on this academic path.

Contents

List of Figures

List of Tables

Introduction 1

Mankind has always tried to simplify its life through technological progress. Thousands of years before Christ the wheel was invented, later the pulley, the printing press, and the steam locomotive. At the beginning of the 19th century, the first generators were built to produce electricity, and in 1835, the first light bulb was invented by James Bowman Lindsay[1]. In the course of time more and more complex electrical circuits were produced and finally in 1941 the first digital calculating machine, the computer was developed. This calculating machine is able to perform various arithmetic operations on the basis of commands. Through the development of transistors, more and more complex computers could be produced, which are able to execute various tasks. The definition of these tasks are captured in software. The computer has been so successful in fact that it has spawned its own scientific discipline, Computer Science. Nowadays it is impossible to imagine life without computers: almost every household and company owns computers. Computers facilitate various tasks, from communication to the acquisition of knowledge. For all these tasks, software developers have created appropriate tools. Software development is the process of writing a script with a programming language to specify how the computer should behave for a given input. Simply put: A software program tells the computer what to do if the user enters a command. This works very well if the tasks is clearly defined and can be described precisely. However, there exists tasks which are almost impossible to program. For example, writing a script that detects cats in images is almost impossible because we cannot describe how a cat looks like[2].

1: and not, as so often falsely claimed, by Thomas Alva Edision

2: at least not on a pixel-level basis so that we can simply compare a given image with our description

Therefore, computer scientists came up with the idea to not just program such tasks but to let the computer learn them instead. Machine learning (ML) are algorithms that are able to learn and adapt without following explicit instructions such as program code. Instead, they use statistical models to analyse data, to find patterns in the data and to draw inferences out of it. Machine learning has become an indispensable part of our everyday lives. For example, we use it for machine translation, transport and logistics organization, product recommendations, fraud detection, self-driving cars, unlocking smartphones, improving video games, speech recognition, and much more. A sub-branch of Machine Learning, namely Deep Learning, has made waves in the last decade. Breakthroughs are being made with this technology are made almost monthly and allows us to execute more and more tasks.

However, such DL system are usually good at one ore a few closely related tasks. In fact, they have they have some crucial flaws by definition (c.f. Section Section 2.2) which cannot be resolved for sure in the current DL framework. One of the Godfathers of Deep Learning is the Turing Award winner Geoffrey Hinton. Especially his contribution to back propagation (c.f. Section Section 2.1) has shaped the field. Back propagation is *the* core learning algorithm of DL systems and was developed in 1986. More than

30 years later, in 2017, Hinton says that he is "deeply suspicious" about back propagation and in his view we have to "throw it all away and start again" to improve current systems fundamentally [1]. Considering what DL systems have achieved, this seems a bit extreme. However, it also shows that the current learning algorithm of such systems has serious flaws.

[1]: Inc. (2017)

Mankind is often inspired by nature when it comes to developing novel systems. The development of artificial intelligence (AI) and Deep Learning has been strongly influenced by, according to our perception, one of (if not the) most intelligent systems on planet Earth, the human brain. The brain is studied by the scientific field of neuroscience[3]. From neuroscience or related fields come various insights on how the human nervous system and especially the brain works. Much of this knowledge has been gained through observations and experiments on humans or other living creatures. Often these findings are only what can be measured from the outside, the real core, the mechanism that causes intelligence, remains unknown.

3: there exist many additional (sub-)fields studying the brain such as cognitive science, cognitive psychology, neurology, and neuropsychology

While Deep Learning is clearly inspired by the insights of Neuroscience, the two fields have little in common in their present form. The implementation of neuroscientific findings has emerged as a subfield in its own right, called Neurocomputing (c.f. Section Section 2.4). In general, neurocomputing is closer related to the functionality of the human brain than Deep Learning. Neurocomputing has produced many promising algorithms that can overcome some weaknesses faced by deep learning systems. Although neurocomputing has also achieved significant breakthroughs, most systems used in everyday life are still based on the principle of Deep Learning.

TODO: deep learning or Deep Learning -> make uniform

1.0.1 Motivation

Despite the fact that Deep Learning has achieved incredible performance on a variety of tasks it is still questionable whether the current methodology is enough to achieve real (or at least more advanced) intelligence (c.f. Section 2.2). Many of the current limitations are tackled by approaches from the field of neurocomputing (c.f. Section 2.4). However, even tough algorithms from the field of neurocomputing have interesting properties, it usually does not perform as good as Deep Learning and often works on specific or small data only.

Neuroscience provides a source of inspiration for AI algorithms, independent of mathematical models. So far, neuroscientists have studied the building blocks of the brain, their responsibilities in the overall process as well as how they interconnect and communicate. The "algorithm" that makes the brain intelligent remains unknown. Furthermore, it is not known which observed functionality in the brain is really part of this algorithm and which are just consequences of it. For example, it is hard to tell if the dynamics in the human brain are necessary to achieve intelligence or if this is just nature's way of implementing it.

Renowned scientists [2] put forward the hypothesis that the core of natural intelligence lies within the self-organization of the brain. Self-organization is the process by which individual units organize their global behavior by local interactions amongst themselves. There is no central control unit that orchestrates the units. In the context of the brain, self-organization primarily affects how neurons are interconnected. The brains of living beings already have certain structures from birth, which are specified by the DNA. In the course of time, these structures change. For example, neurons or connections between neurons may be added or removed. Thus, the brain adapts to its environment over time.

[2]: Malsburg et al. (2022)

Such a mechanism has not been successfully implemented in deep learning nor neurocomputing systems. In deep learning system, the architecture is usually predefined and only the parameters (i.e. weight and biases, c.f. 2.1) are updated. In neurocomputing, on the other hand, the theory behind self-organization is often associated with Hebbian Learning (c.f. Section Section 2.4.1). this dynamic learning of connections is often associated with Hebbian Learning (c.f. Section Section 2.4.1). The theory behind this learning rule can be summarized as "cells that fire together wire together"[3]. However, Hebbian learning is usually rather used to update the weight between connections instead of to add or to remove connections. Connections are therefore not removed or added but can be "turned off" and "turned on" if the activation function of a neuron uses a threshold below which signals are no longer transmitted[4]. In this case, the architecture of the network is to a large part still predefined and self-organization can for example not be used to add additional layers.

[3]: Löwel et al. (1992)

4: if the weights get small enough, the threshold limit cannot be reached and the neurons output is 0

This thesis aims to combine the strengths of deep learning and neurocomputing with recent findings of neuroscience. Especially the hypothesis that self-organization is key for natural intelligence is incorporated in the learning process of modern artificial neural networks. While in classical Deep Learning the architecture is predefined, in this thesis we attempt to learn not only the parameters but also the architecture itself. The architecture is built by self-organization. his means that local communication between the building blocks defined by an initial architecture takes place and that new connections can be created or existing connections can be removed. The challenge is how this re-wiring of the connections can be be learned. It seems certain that the existing learning algorithm back-propagation must either be extended or replaced. However, proposing a concrete algorithm based on the rather abstract findings from Neuroscience is part of this thesis.

1.0.2 Organization of Thesis

The remainder of the thesis is organized as follows: In chapter 2 we present the fundamentals necessary to understand this thesis. We provide an overview about how deep learning works and what the most common research areas in neurocomputing are. Chapter 3 presents the work related to this thesis.

TODO....

Fundamentals 2

TODO: Input Thilo -> mache den Bezug von diesem Teil auf die schlussendliche Arbeit viel deutlicher -> was wird wie gebraucht?

Machine Learning uses mathematical functions to map an input to an output. These functions usually extract patterns from the input data to build a relationship between input and output. The term Machine Learning stems from the fact that we use *machines* to correlate the input and the output to a function (i.e. to *learn* a function) during a training period. A sub-branch of Machine Learning is Deep Learning (DL). DL algorithms are able to learn hidden patterns within data to make predictions. They benefit from the accelerated computing power and big data made available in the last decade. Deep Learning is considered state-of-the-art for many learning tasks especially for high-dimensional data. Typical high-dimensional data are texts, audio recordings, 2D as well as 3D images, and videos. Deep Learning models use artificial neural networks to learn the mapping function between given input and output data. In the following, the fundamentals of Deep Learning is explained. Only those aspects that are relevant for the understanding of the rest of the thesis are discussed.

2.1 Artificial Neural Networks

The idea for artificial neural networks (ANN) stems from biology and aims to capture the interaction of brain cells (neurons) with a mathematical model. A first model for a neuron was proposed by MuCulloch and Pitts in 1943 [4]. Similar to how a"neuron of the human brain transmits electrical impulses through the nervous system, the artificial neuron of MuCulloch and Pitts receives multiple input signals and transforms them into a output signal. A neuron takes an input vector $\boldsymbol{x} = (x_1, ..., x_n)$ where $x_i \in \{0, 1\}$ and maps it to an output $\hat{y} \in \{0, 1\}$. The mapping from the input to the output is done by using an aggregation function g that sums up the input vector $\boldsymbol{x}$ and a activation function f that outputs 1 if the output of g is greater than a threshold θ and 0 otherwise.

[4]: McCulloch et al. (1943)

$$g(\boldsymbol{x}) = g(x_1, ..., x_n) = \sum_{i=1}^{n} x_i \tag{2.1}$$

$$\hat{y} = f(g(\boldsymbol{x})) = \begin{cases} 1, & \text{if } g(x) \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

In 1958, Rosenblatt Rosenblatt [5] developed the Perceptron which works with real numbers as input. The input vector $\boldsymbol{x} = (x_1, ..., x_n)$ where $x_i \in \mathbb{R}^n$ is multiplied with a weight vector $\boldsymbol{w} = (w_1, ..., w_n)$ where $w_i \in \mathbb{R}^n$ with the same length.

[5]: Rosenblatt (1958)

$$g(\boldsymbol{x}) = g(x_1, ..., x_n) = \sum_{i=1}^{n} w_i \cdot x_i \tag{2.3}$$

The output $\hat{y} \in \{0, 1\}$ is similar to the MuCulloch and Pitts neuron 1 if the aggregated value is greater than a threshold θ and 0 otherwise as described in equation **??**. The equations (2.3) and (2.2) can be rewritten as

$$\hat{y} = f(g(\boldsymbol{x})) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i \cdot x_i - \theta \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

Later, the step-function f was replaced with other functions so that the output could also be a real number $\hat{y} \in \mathbb{R}$. Often used functions are

$$\begin{aligned} \text{Sigmoid: } \sigma(z) &= \frac{1}{1 + \mathrm{e}^{-z}} \\ \text{Rectified linear unit (ReLU): } (z)^+ &= \max 0, z \\ \text{Hyperbolic tangent (tanh): } \tanh(z) &= \frac{\mathrm{e}^{z} - \mathrm{e}^{-z}}{\mathrm{e}^{z} + \mathrm{e}^{-z}} \end{aligned} \tag{2.5}$$

By convention, instead of using the threshold $-\theta$ often a bias b is used which leads to:

$$\begin{aligned} z = g(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} &= \sum_{i=1}^{n} w_i \cdot x_i + b \\ \hat{y} &= f(z) \end{aligned} \tag{2.6}$$

However, the brain consists of multiple neurons which are connected through synapses. Therefore, ANNs consist not only of one neuron but combine multiple neurons in a network. These neurons are organized into layers. A shallow neural network consists of one hidden layer in which the input $\boldsymbol{x}$ is fed to calculate the output $\hat{\boldsymbol{y}}$[1]. The universal approximation theorem of Cybenko [6] proves that a shallow network with enough neurons can approximate any mapping function between inputs and outputs. However, very complex mapping functions may need too many hidden neurons. The neurons of the hidden layer extract features in the input space. Since only one layer is used, the features cannot be hierarchically organized and become complex enough.

1: $\hat{\boldsymbol{y}}$ has become a vector since multiple neurons produce multiple output values

[6] Cybenko (1989)

A multi-layer perceptron (MLP), on the other hand, consists of multiple layers. The different layers extract increasingly complex features. In a MLP. the input $\boldsymbol{x}$ is fed into the first layer, each subsequent layer l gets the output of the previous layer $l-1$ as input. In the following, we describe the mathematical model for fully connected layers, where all neurons of a layer are connected with the subsequent layer[2]. For a MLP with m layers, we define the output of the aggregation g as $\boldsymbol{z}^{[l]}$ and the output the activation function as $\boldsymbol{a}^{[l]}$ for layer l. Furthermore, we use $\boldsymbol{w}^{[l]}$ for the weight vector and $b^{[l]}$ for the bias of layer l. Thus, the mathematical model of a MLP is defined as

2: many modern network architectures are not fully connected and can have either missing or recurrent connections

$$\begin{aligned} \boldsymbol{z}^{[l]} &= \boldsymbol{w}^{[l]}\boldsymbol{a}^{[l-1]} + b^{[l]} \\ \boldsymbol{a}^{[l]} &= f(z^{[l]}) \end{aligned} \tag{2.7}$$

Since the input is fed into the first layer and the output is the result from the last layer $\boldsymbol{x} = a^{[0]}$ and $\hat{\boldsymbol{y}} = a^{[m]}$ holds true.

So far, only the forward-pass which is used to calculate the output $\hat{\boldsymbol{y}}$ was discussed. However, the model output $\hat{\boldsymbol{y}}$ will only be close to the target output $\boldsymbol{y}$ if the weights $\boldsymbol{w}^{[l]}$ and biases $b^{[l]}$ are properly defined. These parameters are learned during a training period. The training can take place in a supervised, semi-supervised, self-supervised, unsupervised, or reinforcement learning based manner. In supervised learning, the output of the model $\hat{\boldsymbol{y}}$ for a given input $\boldsymbol{x}$ is compared to manually created target outputs $\boldsymbol{y}$. Unsupervised learning, on the other hand, tries to find patterns in the input $\boldsymbol{x}$ and to cluster the samples into meaningful groups without using target labels. Semi-supervised learning is a hybrid approach of the the aforementioned principles that combines a small amount of labelled data with a large amount of unlabelled data. In self-supervised learning, the target outputs $\boldsymbol{y}$ are directly derived from the input data $\boldsymbol{x}$ (e.g. predict a masked part of the input $\boldsymbol{x}$). Lastly, reinforcement learning algorithms aim to maximize a reward that they become from an environment based on some action they executed.

These learning principle have in common that a loss function $\mathcal{L}$ can calculate a loss value based on the model output $\hat{\boldsymbol{y}}$. For example, the mean square error (MSE) can be used for regression problems or the negative log-likelihood for classification problems. The chosen loss function is minimized interatively with stochastic gradient descent (SGD)[3] until the network converges. The idea behind stochastic gradient descent is to make use of the fact that the negative gradient of the loss value points to the direction of the steepest descent (i.e. in the direction where the loss gets smaller). SGD therefore updates the network parameters by taking a step of size η in the direction of their negative gradient

$$\begin{aligned} \Delta\boldsymbol{w}^{[l]} &= -\eta\nabla_{\boldsymbol{w}^{[l]}}\mathcal{L} \\ \Delta b^{[l]} &= -\eta\nabla_{b^{[l]}}\mathcal{L} \end{aligned} \tag{2.8}$$

The gradients of the weights $\boldsymbol{w}^{[l]}$ and biases $b^{[l]}$ can efficiently be calculated with an algorithm called backpropagation [8], which is just a smart implementation of the chain rule[4].

TODO: Describe CNN, Transformer, ... etc.???

3: There exist also other optimizer methods such as SGD with momentum, RMSprop, or Adam [7]

[8]: Rumelhart et al. (1986)

4: While a detailed discussion on backpropagation is out of scope for this thesis, we refer interested reader to the Deep Learning course by Andrew Ng [9]

2.2 Limitations of Deep Learning

The rise of Deep Learning over the past decade has only been possible because of major technological advances in hardware. Without the computational resources and the storage capacity of the systems created in the last decades no system could run today's algorithm. Moreover, much of the progress of recent years was possible due to the improved hardware. Moore's law [10] states that the number of transistors in a

dense integrated circuit doubles about every two years and is the only known physical process following an exponential curve. An analysis by OpenAI shows that since 2012 the amount of compute has even increasing exponentially with a doubling time of 3.4 months [11]. However, the exponential increase seems to come to an end since the size of transistors hit physical limitations. It is assumed that Moore's law will end by around 2025 [12]. Besides the progress in the field and the development of new technology, Deep Learning models also became better because the number of parameters and the size of datasets grew exponentially. Even the growth in the last five years is astonishing. While the state-of-the-art language model from 2018 [13] had around 94M parameters, the state-of-the-art in 2020 [14] already had 175B parameters. Training such a model on a single V100 GPU would take about 355 years and cost about 4.6M dollars [15]. A recent language model from Microsoft and Nvidia [16] even has 530B parameters. Only a few institutions with massive resources are able to train such big models. In general, inference on low-budget hardware such as smartphones or embedded hardware becomes prohibitive with the growing size of deep networks. Even tough there exist techniques to shrink the model size after training such as quantization [17], model pruning [18], or model distillation [19] it is questionable if making models bigger is the best way to develop intelligent systems.

[12]: Kumar (2015)

[13]: Peters et al. (2018)

[14]: Brown et al. (2020)

[16]: Shoeybi et al. (2020)

Another major issue of Deep Learning systems is that they suffer from catastrophic forgetting. If a model is trained on a specific task and afterwards trained (or fine-tuned) on another task, the model suffers a "catastrophic" drop in performance over the first task. The reason for this effect is that the model during training on the second task adjusts the parameters learned during the first task and therefore "forgets" the learned mapping functions. Just mixing all datasets or to learn all tasks in parallel in a current multi-task setup [20] doesn't seem feasible to achieve some kind of general intelligence as this involves too many different unrelated tasks. Catastrophic forgetting is also caused by the fact that learning is mostly done offline[5]. Online learning [21] and lifelong learning [22] are currently hot research topics. However, these methods have not yet been established.

5: Offline in this context means that the model parameters are not adapted after training during inference time

Furthermore, there exists problems which may cannot be solved with the current principles used for Deep Learning. First of all, it is questionable if Deep Learning models can achieve *real* generalization[6]. With enough data, can achieve generalization in the sense that the model can interpolate within the data distribution. However, deep learning models fail to extrapolate. For example, convolutional neural networks (CNNs) do not generalize to different viewpoints unless they are added to the training data [23].

6: Generalization refers to the ability of the model to adapt properly to previously unseen data from the same distribution

[23]: Madan et al. (2022)

Second, Deep Learning is not able to learn abstract relationships in a few trials but requires many samples of it and is thus data hungry.[7] Marcus Gary [24] argues that if he tells that a "schmister" is a sister over the age of 10 but under the age of 21, humans can immediately infer whether they or their best friends have any "schmister". However, modern DL systems lacks a mechanism for learning abstractions through explicit, verbal definition and require thousands or even more training samples.

7: Delme!!!

[24]: Marcus (2018)

Third, no DL model has been able to demonstrate causal reasoning in

a generic way. Deep Learning models find correlations between the inputs and the outputs, but not the causation. Other AI approaches such as hierarchical Bayesian computing or probabilistic graphical models are better at causal reasoning but cannot be well combined with Deep Learning models.

Lastly, Deep Learning models are to some extend too isolated since they have no embodiment and cannot interact with the world. For example, the human body provides needs, goals, emotions, and gut feeling[8]. In current Deep Learning systems are emotions totally absent and the goals are set externally. Deep Reinforcement Learning can be considered as a first step in the direction of dissolving this isolation, as they interact with a virtual environment. AI systems that interact with the real world do not work well so far. Moravec's paradox [25] states that "it is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility".

8: one could argue that the body is therefore even a co-processor of the brain

[25]: Moravec (1995)

2.3 Biological Learning

The human brain comprises many interconnected areas processing everything in parallel. For example, Figure Figure 2.1 illustrates the connections between different organizational units in the cerebral cortex which are responsible for vision. It can be seen that these areas are connected in a rather complex structure. Deep Learning architectures, on the other hand, are mostly unidirectional and the signal flows unidirectional from layer to layer[9]. However, the choice of the architecture influences the how the model can learn the mapping function from input to output. It could be that the complex structure of our brain comprises an inductive bias which was learned over time through evolution.

9: Except for recurrent connections, skip-connections, or residual-connections

A learning system requires a mechanism that tells the system if something goes well or wrong so that it can learn from it. This is called the *credit assignment problem*. Backpropagation (c.f. Section **??**) solves this problem by propagating the error backwards through the network. However, information flows in the brain only in one direction from the presynaptic neurons to the postsynaptic neurons. Therefore, backpropagation is not biologically plausible. Lillicrap et al. [27] shows that an additional set of random feedback weights is able to transmit useful gradients. Their work has reopend questions how the brain could process error signals and has dispel some long-held assumptions about algorithmic constraints on learning.

[27]: Lillicrap et al. (2016)

Not only the structure of the network and the way how the feedback is calculated is different between biological learning and Deep Learning. Also the neurons themselves are different. While the artificial neuron doesn't have any dynamics (c.f. Equation (2.6)), biological neurons are highly dynamic: Biological neurons adapt their firing rate to constant inputs, they may continue firing after an input disappears, and can even fire when no input is active.

TODO: Add reference to reservoir computing

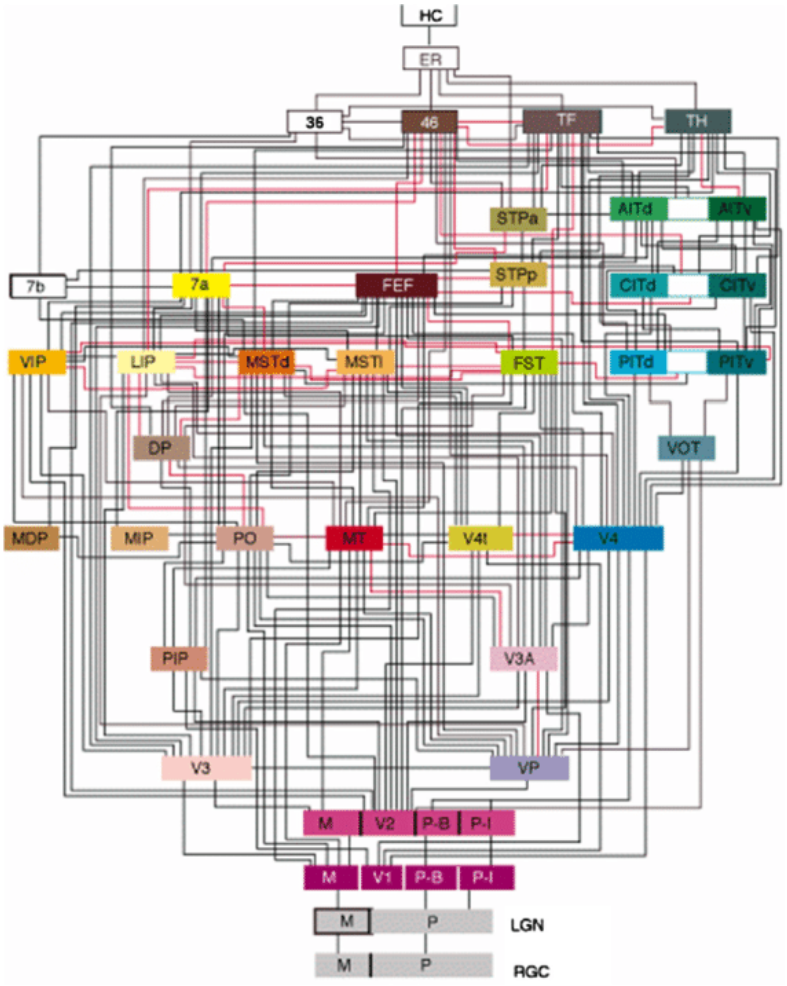

Figure 2.1: The organization of the visual system in the cerebral cortex. The image is from Felleman et al. [26].

Lastly, the neurons in the brain are self-organizing. This means that a group of elementary units such as neurons or a group of neurons perform similar rule of behavior on a sub-set of the available information. Such a system doesn't have a central supervision that orchestrates these units. Each unit applies similar deterministic functions to the information received. Two important principles of such systems are (i) localized learning which means that each unit adapt their behavior to the information they receive; and (ii) emergence which means that there is no explicit loss function that tells the system what to do.

2.4 Neurocomputing

2.4.1 Hebbian Learning

[28]: Hebb (1949)

Donald Hebb [28] describes how the connections between cells in the nervous system adapt as: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased". This statement

is often simplified to the well-known phrase "Neurons that fire together wire together".

Hebbian learning is based on this principle. The weight w_{ij} from neuron i to neuron j changes based on the pre-synaptic activity r_i of neuron i and post-synaptic activity r_j of neuron j

$$\Delta w_{ij} = \eta r_i r_j \tag{2.9}$$

where η is the learning rate. Thus, the weights between frequently co-activated neurons becomes strong which is called Hebbian plasticity.

In its original form, Hebbian learning had the problem that the connections could only become stronger but not weaker. Therefore, it is often extended based on the covariance of the activity between neurons. The covariance is positive if two neurons fire often together and negative if they do not often fire together. The following equation changes the weight relative to the covariance:

$$\Delta w_{ij} = \eta(r_i - \psi_i) \cdot (r_j - \psi_j) \tag{2.10}$$

where ψ_i and ψ_j are estimates of the expected pre- and post-synaptic activity[10]. The formulation above lacks of boundaries, i.e. the weights could grow to infinite. A simple solution is to enforce hard boundaries $w_{min} \leq w_{ij} \leq w_{max}$.

10: the expected activity can for example be estimated through a moving average function

Another solution to weaken the connections is given by the Bienenstock-Cooper-Monroe (BCM) learning rule which was introduced by Bienenstock et al. [29] and extended by Intrator and Cooper [30]. They propose a sliding threshold for long-term potentiation (LTP) or long-term depression (LTD) induction. When a pre-synaptic neuron fires and the post-synaptic neuron is in a lower activity state than the sliding threshold, it tends to undergo a LTD (i.e. the connection is weaken).

[29]: Bienenstock et al. (1982)

Around the same time, Oja [31] improved the learning rule of Equation (2.9) with an normalization term:

[31]: Oja (1982)

$$\Delta w_{ij} = \eta(r_i r_j - \alpha r_j^2 w_{ij}) \tag{2.11}$$

The parameter α is a constant value that determines the size of the norm of the weight vector. This update rule is also known as the Oja learning rule. Furthermore, he has found that a layer of multiple linear neurons converges to the first principle component of the input data. As all neurons only learn the first principle component, a network of multiple neurons in this setting seem not very useful. Differentiation between neurons can be achieved with several different methods. Two well known approaches are the winner-take-all competition (i.e. only the neuron with the most similar activity is selected for learning)[11] and a recurrent circuit that provides a competitive signal (i.e. the neurons compete with their neighbours to become active to learn).

11: in practice is k-winner-take-all often preferred where k instead of one neuron learns

It is known that independent neurons can encode more information and work better than dependent neurons [32]. Anti-Hebbian learning is a

[32]: Simoncelli et al. (2001)

method that adds a penalty for similarly active neurons and thus minimizes the linear dependency between neurons. Vogels et al. implemented this by switching the sign of the weight change [33].

[33]: Vogels et al. (2011)

There exists many further improvements for Hebbian learning which are not summarized in this thesis. For example, Joshi and Triesch [34] as well as Teichmann and Hamker [35] adapt the activation function of the neurons to enforce a certain activity distribution and to stabilize Hebbian learning even in multilayer neural networks.

[34]: Joshi et al. (2009)
[35]: Teichmann et al. (2015)

Similar to large parts of the brain, Hebbian learning is unsupervised and learns based on local information (i.e. neurons in close proximity). However, the brain is also largely recurrent and could guide neighbouring or preceding units. This assumption inspired supervised Hebbian learning. In supervised Hebbian learning, a subset of inputs which should evoke post-synaptic activity can be selected. Supervised Hebbian learning can be extended to top-down and bottom-up learning [36] which leads to a combination of supervised and unsupervised Hebbian learning.

[36]: Grossberg (1988)

2.4.2 Hopfield Networks

[37]: Hopfield (1982)

Hopfield networks [37] were introduced 1982 by J. Hopfield. They serve as associative (i.e. content-adressable) memory systems. Such systems are particularly useful to retrieve representations based on degraded or partial inputs. Auto-associative memories return for an input the most similar previously seen sample. A classical implementation of an auto-associative memory is the nearest neighbour algorithm [38]. This algorithm compares a given samples with the previously seen training data with a distance metric and returns the most similar sample[12]. Memory networks [39] implement an auto-associative memory within the Deep Learning framework. Such networks convert an input $\boldsymbol{x}$ to a internal feature representation $I(\boldsymbol{x})$, update memories given the new input $m = G(m, I(\boldsymbol{x}))$, and compute the output features $o = O(m, I(\boldsymbol{x}))$. This process is applied during the training and inference phase. The only difference is that the parameters for the functions I, G, and O are only updated during training.

[38]: Fix et al. (1989)

12: or the k most similar samples in the case of the k-nearest neighbour (k-NN) algorithm

[39]: Weston et al. (2015)

In a Hopfield network are all neurons are connected, but there are no self-connections: $w_{ii} = 0$ where w_{ij} is the weight between neuron i and neuron j. Furthermore, the weights are symmetrical $w_{ij} = w_{ji}$. A Hopfield network in its original form works only with binary units. For consistency, this networks are called binary Hopfield networks in the following. The output of a neuron in a binary Hopfield network depends on the output of the other neurons within the network:

$$x_i = \sum_{i \neq j} w_{ij} y_j + b \tag{2.12}$$

$$y_i = \begin{cases} 1, & \text{if} x > 0 \\ -1, & \text{otherwise} \end{cases} \tag{2.13}$$

TODO in equation: check where to use y and where to use y with hat

Hopfield networks have their own dynamics and the output evolves over time. If the initial value y_i of a binary Hopfield network has a different sign than $\sum_{i \neq j} w_{ij} y_j + b$ the output will flip (i.e. change its sign). This will in turn influence all other neurons which may also flip. The term $y_i(\sum_{i \neq j} w_{ij} y_j + b)$ is negative if y_i is not equal to $\sum_{i \neq j} w_{ij} y_j + b$, otherwise it is positive. Since the neuron flips if the term $y_i(\sum_{i \neq j} w_{ij} y_j + b)$ is negative or stays the same if this term is positive, the change of this term can only be positive:

$$\Delta[y_i(\sum_{i \neq j} w_{ij} y_j + b)] \geq 0 \tag{2.14}$$

The negative sum of the term $y_i(\sum_{i \neq j} w_{ij} y_j + b)$ for the entire network is called the energy E of the network:

$$E(\boldsymbol{y}) = -\sum_i y_i \left(\sum_{j>i} w_{ji} y_j + b \right) \tag{2.15}$$

As we have shown in (2.14), the energy function $E\boldsymbol{y})$ of the network can only decrease. Moreover, the energy function has a lower bound and thus the network reaches after a finite number of iterations a stable state. A stable pattern of the network (i.e. no neurons flip their sign) is a local minima of the energy function and is called a point attractor. A pattern is attracted by the closest stable pattern. Thus, the network can be used as associative memory because an input pattern can be matched to the closest stable pattern.

McElice [40] has shown that a binary Hopfield network with N neurons has a capacity of $C = 0.138N$ (i.e. the number of patterns that can be stored. Hebbian learning (c.f. Section 2.4.1) can be used to store pattern in a Hopfield network. To store a pattern, the weights $\boldsymbol{w}$ must be choosen in a way so that the desired local patterns $(\boldsymbol{y}^1, ..., \boldsymbol{y}^P)$ are local minima of the energy function. By combining Hebbian learning with some smart mathematical transformations it can be shown that the weights can be directly learned with only one iteration over the training patterns[13]:

[40]: McEliece et al. (1987)

13: for the derivation of this equation refer to [37]

$$\boldsymbol{w} = \frac{1}{p} \sum_{k=1}^{P} \boldsymbol{y}^k \times (\boldsymbol{y}^k)^T - \boldsymbol{I} \tag{2.16}$$

where $\boldsymbol{I}$ is the identity matrix. Later, Hopfield et al. [41] extended the binary Hopfield network so that it can either learn pattern during an awake cycle or forget patterns during a sleep cycle.

[41]: Hopfield et al. (1983)

One of the limiting factors of binary Hopfield networks is the capacity of $C = 0.138N$. The problems comes from the fact that the energy function is a quadratic function. More than three decades after the introduction of the binary Hopfield networks, Krotov and Hopfield [42] reformulated the energy function as a polynomial function to get polynomial capacity $C \approx N^{a-1}$ where a is the order of the function. Later, the energy function was reformulated as exponential function [43] and thus modern Hopfield networks have an exponential capacity of $C \approx 2^{\frac{N}{2}}$.

[42]: Krotov et al. (2016)

[43]: Demircigil et al. (2017)

The second limiting factor of binary Hopfield networks is that only binary patterns can be stored. Ramsauer et al. [44] extended the binary Hopfield network to continous patterns by reformulating the energy function and the corresponding update rule. Continous Hopfield networks can retrieve continous patterns or even combination of several similar continous patterns. The authors claim that a continous Hopfield networks can replace fully-connected layers, attention layers, LSTM layers, support vector machines (SVM), and k-NN.

[44]: Ramsauer et al. (2021)

TODO: References for LSTM, attention, SVM, ...???

2.4.3 Spiking Neural Networks

Biological neurons emit spikes. To transmit information, especially the firing rate (i.e. the number of spikes per second in Hz) and precise timing of the spikes are relevant. The amplitude and duration of the spike does not matter much. This behaviour has also been implemented in ANNs. So called Spiking neural networks (SNNs) incorporate the concept of time into their model. SNN do not transmit information in each forward-pass but rather transmit a signal when the membrane potential reaches a threshold value[14]. The neuron fires as soon as the threshold is reached and thereby influences the potential of other neurons. The most prominent model of a spiking neuron is the leaky integrate-and-fire (LIF) neuron [45]. The LIF neuron models the membrane potential with a differential equation. Incoming spikes can either increase or decrease the membrane potential. The membrane potential either decays over time or is reset to a lower value if the threshold value is reached and the neuron has fired. There exists different integrate-and-fire (IF) neurons models such as the Izhikevich quadratic IF [46] or the adaptive exponential IF [47]. While each of these model gas different mathematical properties, the concept of a membrane potential that is increased or decreased through spikes from other neurons and decays over time or by emitting a spike is similar to the LIF.

14: the membrane potential is related to the electrical charge of the membrane of a biological neuron

[45]: Abbott (1999)

[46]: Izhikevich (2003)

[47]: Brette et al. (2005)

Biological neurons have different dynamics. Some neurons fire regularly if the receive an input current, others slow down the firing rate over time or emits bursts of spikes. Modern models of spiking neurons can recreate this behaviour of biological neurons [48].

[48]: Paugam-Moisy (2006)

The synaptic plasticity can be modeled with Hebbian learning (c.f. Section 2.4.1). The spike-timing dependent (STDP) plasticity rule [49] distinguishes the tiring behaviour of pre-synaptic and post-synaptic neurons. If the pre-synaptic neurons fires before the post-synaptic neuron, the connection is strengthen, otherwise it is weaken.

[49]: Bi et al. (2001)

For a long time, SNN only worked for very shallow networks. In 2018, Kheradpisheh et al. [50] has proposed a SNN based on the idea of CNNs called a deep spiking convolutional network. This network uses convolutional and pooling layers with IF neurons instead of classical artificial neurones and is trained with STDP. First, the image is fed into DoG cells. This cells apply the difference of Gaussians (DoG) feature enhancement algorithm. This algorithm subtracts a Gaussian blurred version of an image from the original image. By doing so, positive or negative contrast is detected in the input image. The higher the contrast is, the stronger is a cell activated and the earlier it emits a spike. Thus, the

[50]: Kheradpisheh et al. (2018)

order of the spikes depends on the order of the contrast. These spikes are forwarded to a convolutional layer. Deep spiking convolutional networks use two types of LIF neurons: On-center neurons fire when a bright area is surrounded by a darker area, off-center neurons do the opposite. Convolutional neurons emit a spike as soon as they detect their preferred visual feature[15]. Neurons that fire early perform the STDP update with a winner-takes-all mechanism. This means that the neurons within a layer compete with each other and those which fire earlier learn the input pattern. This prevents other neurons from firing and guarantees a sparse connection. Later convolutional layers detect more complex features by integrating input spikes from the previous layer. The features from the last convolutional layer are flatten and a Support Vector Machine is used to classify the features.

15: the location of the feature is not relevant as convolution layers are translation invariant

2.4.4 Reservoir Computing

As described in Section Section 2.3, biological neurons are highly dynamical while artificial neurons are not. Reservoir computing introduces such dynamics into an artificial network. Reservoir computing is an umbrella term for networks based on the concepts of Echo State Networks (ESN) [51] and Liquid State Machines (LSM) [52]. A reservoir is a fixed non-linear system that maps a input vector $\boldsymbol{x}$ to a higher dimensional computation space. After the input vector is mapped into computation space, a simple readout mechanism is trained to return the desired output based on the reservoir state. In principle, the system should be capable of any computation if it has a high enough complexity [53]. However, not every system is suited as reservoir. A good reservoir system distributes different inputs into different regions of the computation space [53].

[51]: Jaeger (2001)

[52]: Maass et al. (2002)

[53]: Konkoli (2018)

A ESN is a set of sparsely connected recurrent neurons as visualized in Figure 2.2.

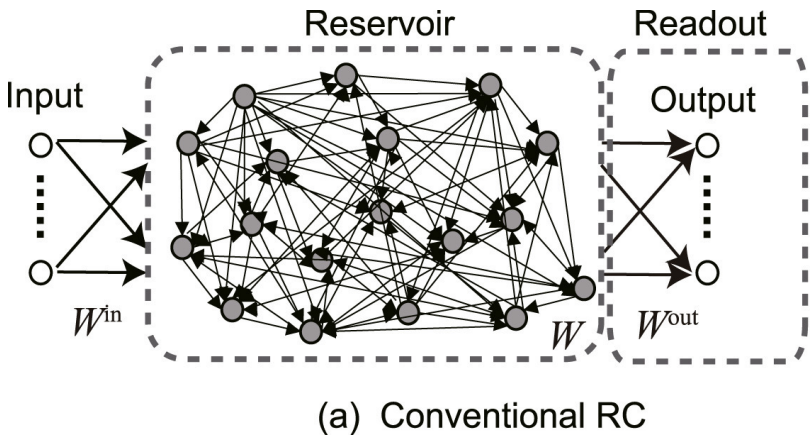

Figure 2.2: Structure of a Echo State Network. The image is from Tanaka et al. [54].

The reservoir consists of N nodes which are connected according a Erdős–Rényi graph model[16] [55]. This graph model is represented by an adjacency matrix $\boldsymbol{w}$[17] of size $N \times N$. The time varying input signal $\boldsymbol{x(t)}$ is mapped to a sub-set of N/M graph nodes by multiplying it with $\boldsymbol{w}_{\text{in}} \in \mathbb{R}^{N \times M}$ and the output by multiplying the reservoir state with $\boldsymbol{w}_{\text{out}} \in \mathbb{R}^{M \times N}$. We refer interested reads to [56] to read more about the mathematical properties and how network is updated in detail.

[55]: Erdős et al. (1959)

16: The Erdős–Rényi model is a model for generating random graphs where all graphs on a fixed set of vertices and edges is equally likely

17: Figure 2.2 uses upper case letter $\boldsymbol{W}$

[56]: Lukoševičius (2012)

In the original form of ESN, only the readout weights are learned, the rest is chosen randomly. The input $\boldsymbol{x(t)}$ brings the recurrent units in a initial state. The recurrent connections inside the reservoir create different dynamics in the network. The readout neurons linearly transform the recurrent dynamics into temporal outputs. The readout weights $\boldsymbol{w}_{\text{out}}$ are trained to reproduce a target function $\boldsymbol{y(t)}$.

Liquid State machines use a spiking neural network instead of a graph of recurrent units as reservoir. The nodes of the spiking neural network are randomly connected together. Thus, every node receives time varying inputs from the inputs as well as from other nodes. The recurrent connections turns the varying input into a spatio-temporal pattern. Similar to ESN, the spatio-temporal patterns of activation are read out by a linear layer.

In general, reservoirs are universal approximators and can approximate any non-linear function given there are enough neurons in the reservoir. They generalize better and faster than equivalent MLP. The main drawback of current systems is that cannot deal well with high-dimensional inputs such as images.

3 Related Work

3.1 Natural Intelligence

This thesis is inspired by the work "A Theory of Natural Intelligence' from von der Malsburg et al. [2]. Therefore, we dedicate this section to summarize their work in detail.

[2]: Malsburg et al. (2022)

According [2], the process of learning is influenced by "nature", "nurture", and "emergence"[1]. They point out that human genome (as of nature) only contain 1GB of information [57] and humans only absorb a few GB into permanent memory over a lifetime (as of nurture) but it requires about 1PB to describe the connectivity in human brain. Therefore, it is important to distinguish the amount of information to describe a structure from the amount of information needed to generate it. Similar, nature and nurture only require a few GB to construct, respectively instruct the entire human brain. Therefore, they argue that the human brain must be highly structured (i.e. nature and nurture "generate" the human brain by selecting from a set pre-structured patterns). The authors call the process of generating the highly structured network in the human brain the "Kolmogorov [58] Algorithm of the Brain"[2]. Network self-organization is the only mechanism that has not yet been disproved by experiments as the brains Kolmogorov algorithm [59, 60]. This mechanism loops between activity and connectivity, with activity acting back on connectivity through synaptic plasticity until a steady state, called an attractor network, is reached. The consistency property of an attractor network means that a network has many alternative signal pathways between pairs of neurons [61]. Thus, the brain develops as an overlay of attractor networks called net-fragments [62]. Net-fragments consist of small sets of neurons, whereby each neuron can be part of several net fragments. The network self-organization has to start from an already established coarse global structure which is improved in a coarse-to-fine manner to avoid being caught in a local optima.

1: nature refers to the influence of genes and evolution, nurture to the influence of experience and education

[57]: Mellerson et al. (2001)

2: as the Kolmogorov complexity describes the number of bits required by the shortest algorithm that can generate the structure

[58]: Kolmogorov (1998)
[59]: Willshaw et al. (1976)
[60]: Willshaw et al. (1979)

[61]: Malsburg et al. (1987)
[62]: Malsburg (2018)

Also, von der Malsburg et al. [2] discuss scene representation (i.e. how a scene is represented in the brain) even tough they point out that this is a contested concept [63]. Scene representation is a organization framework to put abstract interpretation of scene layouts, elements, potential actions, and emotional responses in relation. The details are not rendered as in photographic images but the framework supports the detailed reconstructions of narrow sectors of the scene. The basic goal if learning is to integrate a behavioral schema into the flow of scene representations. They propose the hypothesis that the network structure resulting from self-organization together with the neural activation in the framework of scene representation are the inductive bias that tunes the brain to the natural environment.

[63]: Freeman III et al. (1990)

Finally, they discuss how net fragments can be used to implement such structures and processes using vision as an example. They point out that

a neuron is grouped in one or multiple net fragments through network self-organization. The net fragments can be considered as filters that detect previously seen patterns in the visual input signal. An object is represented by multiple net fragments, where each fragment responds to the surface of that object and has shared neurons and connections with other net fragments representing that object. Thus, net fragments render the topological structure of the surfaces that dominate the environment. Von der Malsburg et al. [2] propose that net fragments represent shape primitives which can adapt to the shape of actual objects[3]. Shifter circuits are one possible implementation of networks that enable invariant responses to the position- and shape-variant representations [64, 65]. They are composed of net-fragments that can be formed by network self-organization [66]. Ref. [2] also argue that net fragments are the compositional data structure used by the brain. A hierarchy of features may be represented by nested net fragments of different size. Complex objects, such as mental constructs, can thus be seen as larger net fragments composed as mergers of pre-existing smaller net fragments.

3: adapt in spite of metric deformations, depth rotation, and position

[64]: Arathorn (2002)
[65]: Olshausen et al. (1995)

[66]: Fernandes et al. (2015)

3.2 Self-Organization

[67]: Kelso (1995)

The human brain is self-organizing [67]. Self-organization is the process by which systems consisting of many units spontaneously acquire their structure or function without interference from a external agent or system. The absence of a central control unit allow self-organizing systems to quickly adjust to new environmental conditions. Additionally, such systems have in-built redundancy with a high degree of robustness as they are made of many simpler individual units. These individual units can even fail without the overall system breaking down. Dresp [68] describes seven clearly identified properties of self-organization in the human brain: (i) modular connectivity, (ii) unsupervised learning, (iii) adaptive ability, (iv) functional resiliency, (v) functional plasticity, (vi) from-local-to-global functional organization, and (vii) dynamic system growth.

[68]: Dresp (2020)

Before summarizing the literature specific to self-organization of neural networks, the general literature on self-organization with a focus on deep learning is described in the following. Many of these fundamental algorithms for self-organization serve as inspiration for how ANNs can be designed to be self-organizing.

In nature, groups of millions units that solve complex tasks by using only local interactions can be observed. For example, ants can navigate difficult terrain with a local pheromone-based communication and thus form a collective type of intelligence. Such observations inspired researchers to build algorithms which are based on local communication and self-organization, for example ant colony optimization algorithms [69]. DeepSwarm [70] is a neural architecture search method that uses this algorithm to search for the best neural architecture. This methods achieves competitive performance on rather small datasets such as MNIST, Fashion-MNIST, and CIFAR-10.

[69]: Dorigo et al. (1997)
[70]: Byla et al. (2019)

Cellular Automata mimic developmental processes in multi-cell organisms. They contain a grid of similar cells with an internal state which is

updated periodically. The transition from a given state to a subsequent state is defined by some update rules. Such automata can be used to study biological pattern formations [71] or physical systems [72]. Neural Cellular Automata [73] use neural networks to learn these update rules.

[71]: Wolfram (1984)
[73]: Wulff et al. (1992)

Appendix

TODO A

TODO

Bibliography

Here is the list of references in citation order.

[1] Axios Media Inc. *Artificial intelligence pioneer says we need to start over*. 2017. URL: https://www.axios.com/2017/12/15/artificial-intelligence-pioneer-says-we-need-to-start-over-1513305524 (visited on 09/04/2022) (cited on page 2).

[2] Christoph von der Malsburg, Thilo Stadelmann, and Benjamin F. Grewe. 'A Theory of Natural Intelligence'. In: arXiv:2205.00002 (Apr. 2022). arXiv:2205.00002 [cs, q-bio] (cited on pages 3, 17, 18).

[3] Siegrid Löwel and Wolf Singer. 'Selection of Intrinsic Horizontal Connections in the Visual Cortex by Correlated Neuronal Activity'. In: *Science* 255.5041 (1992), pp. 209–212. DOI: 10.1126/science.1372754 (cited on page 3).

[4] Warren S. McCulloch and Walter Pitts. 'A logical calculus of the ideas immanent in nervous activity'. en. In: *The Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943), pp. 115–133. DOI: 10.1007/BF02478259 (cited on page 5).

[5] F. Rosenblatt. 'The perceptron: A probabilistic model for information storage and organization in the brain.' en. In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519 (cited on page 5).

[6] G. Cybenko. 'Approximation by superpositions of a sigmoidal function'. en. In: *Mathematics of Control, Signals, and Systems* 2.4 (Dec. 1989), pp. 303–314. DOI: 10.1007/BF02551274 (cited on page 6).

[7] Diederik P. Kingma and Jimmy Ba. 'Adam: A Method for Stochastic Optimization'. In: *CoRR* abs/1412.6980 (2015) (cited on page 7).

[8] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 'Learning representations by back-propagating errors'. en. In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. DOI: 10.1038/323533a0 (cited on page 7).

[9] Coursera Inc. *Deep Learning Specialization*. 2022. URL: https://www.coursera.org/specializations/deep-learning (visited on 08/19/2022) (cited on page 7).

[10] Gordon E. Moore. 'Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.' In: *IEEE Solid-State Circuits Society Newsletter* 11.3 (Sept. 2006), pp. 33–35. DOI: 10.1109/N-SSC.2006.4785860 (cited on page 7).

[11] Open AI. *AI and Compute*. 2018. URL: https://openai.com/blog/ai-and-compute/ (visited on 08/19/2022) (cited on page 8).

[12] Suhas Kumar. 'Fundamental Limits to Moore's Law'. In: arXiv:1511.05956 (Nov. 2015). arXiv:1511.05956 [cond-mat] (cited on page 8).

[13] Matthew Peters et al. 'Deep Contextualized Word Representations'. en. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202 (cited on page 8).

[14] Tom Brown et al. 'Language Models are Few-Shot Learners'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901 (cited on page 8).

[15] Lambda. *OpenAI's GPT-3 Language Model: A Technical Overview*. 2021. URL: https://lambdalabs.com/blog/demystifying-gpt-3/ (visited on 08/19/2022) (cited on page 8).

[16] Mohammad Shoeybi et al. 'Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism'. In: arXiv:1909.08053 (Mar. 2020). arXiv:1909.08053 [cs] (cited on page 8).

[17] Hao Wu et al. 'Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation'. In: arXiv:2004.09602 (Apr. 2020). arXiv:2004.09602 [cs, stat] (cited on page 8).

[18] Tejalal Choudhary et al. 'A comprehensive survey on model compression and acceleration'. en. In: *Artificial Intelligence Review* 53.7 (Oct. 2020), pp. 5113–5155. DOI: 10.1007/s10462-020-09816-7 (cited on page 8).

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 'Distilling the Knowledge in a Neural Network'. In: arXiv:1503.02531 (Mar. 2015). arXiv:1503.02531 [cs, stat] (cited on page 8).

[20] Yu Zhang and Qiang Yang. 'A Survey on Multi-Task Learning'. In: arXiv:1707.08114 (Mar. 2021). arXiv:1707.08114 [cs] (cited on page 8).

[21] Doyen Sahoo et al. 'Online Deep Learning: Learning Deep Neural Networks on the Fly'. In: arXiv:1711.03705 (Nov. 2017). arXiv:1711.03705 [cs] (cited on page 8).

[22] German I. Parisi et al. 'Continual lifelong learning with neural networks: A review'. en. In: *Neural Networks* 113 (May 2019), pp. 54–71. DOI: 10.1016/j.neunet.2019.01.012 (cited on page 8).

[23] Spandan Madan et al. 'When and how convolutional neural networks generalize to out-of-distribution category–viewpoint combinations'. en. In: *Nature Machine Intelligence* 4.2 (Feb. 2022), pp. 146–153. DOI: 10.1038/s42256-021-00437-5 (cited on page 8).

[24] Gary Marcus. 'Deep Learning: A Critical Appraisal'. In: arXiv:1801.00631 (Jan. 2018). arXiv:1801.00631 [cs, stat] (cited on page 8).

[25] Hans Moravec. *Mind children: the future of robot and human intelligence*. eng. 4. print. Cambridge: Harvard Univ. Press, 1995 (cited on page 9).

[26] D. J. Felleman and D. C. Van Essen. 'Distributed Hierarchical Processing in the Primate Cerebral Cortex'. en. In: *Cerebral Cortex* 1.1 (Jan. 1991), pp. 1–47. DOI: 10.1093/cercor/1.1.1 (cited on page 10).

[27] Timothy P. Lillicrap et al. 'Random synaptic feedback weights support error backpropagation for deep learning'. en. In: *Nature Communications* 7.1 (Dec. 2016), p. 13276. DOI: 10.1038/ncomms13276 (cited on page 9).

[28] D. O. Hebb. *The organization of behavior; a neuropsychological theory.* The organization of behavior; a neuropsychological theory. Oxford, England: Wiley, 1949, pp. xix, 335 (cited on page 10).

[29] El Bienenstock, Ln Cooper, and Pw Munro. 'Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex'. en. In: *The Journal of Neuroscience* 2.1 (Jan. 1982), pp. 32–48. DOI: 10.1523/JNEUROSCI.02-01-00032.1982 (cited on page 11).

[30] Nathan Intrator and Leon N Cooper. 'Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions'. en. In: *Neural Networks* 5.1 (Jan. 1992), pp. 3–17. DOI: 10.1016/S0893-6080(05)80003-6 (cited on page 11).

[31] Erkki Oja. 'Simplified neuron model as a principal component analyzer'. en. In: *Journal of Mathematical Biology* 15.3 (Nov. 1982), pp. 267–273. DOI: 10.1007/BF00275687 (cited on page 11).

[32] Eero P Simoncelli and Bruno A Olshausen. 'Natural Image Statistics and Neural Representation'. en. In: *Annual Review of Neuroscience* 24.1 (Mar. 2001), pp. 1193–1216. DOI: 10.1146/annurev.neuro.24.1.1193 (cited on page 11).

[33] T. P. Vogels et al. 'Inhibitory Plasticity Balances Excitation and Inhibition in Sensory Pathways and Memory Networks'. en. In: *Science* 334.6062 (Dec. 2011), pp. 1569–1573. DOI: 10.1126/science.1211095 (cited on page 12).

[34] Prashant Joshi and Jochen Triesch. 'Rules for information maximization in spiking neurons using intrinsic plasticity'. In: *2009 International Joint Conference on Neural Networks*. 2009, pp. 1456–1461. DOI: 10.1109/IJCNN.2009.5178625 (cited on page 12).

[35] Michael Teichmann and Fred Hamker. 'Intrinsic plasticity: A simple mechanism to stabilize Hebbian learning in multilayer neural networks.' In: Mar. 2015 (cited on page 12).

[36] Stephen Grossberg. 'Nonlinear neural networks: Principles, mechanisms, and architectures'. en. In: *Neural Networks* 1.1 (Jan. 1988), pp. 17–61. DOI: 10.1016/0893-6080(88)90021-4 (cited on page 12).

[37] J J Hopfield. 'Neural networks and physical systems with emergent collective computational abilities.' en. In: *Proceedings of the National Academy of Sciences* 79.8 (Apr. 1982), pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554 (cited on pages 12, 13).

[38] Evelyn Fix and J. L. Hodges. 'Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties'. In: *International Statistical Review / Revue Internationale de Statistique* 57.3 (Dec. 1989), p. 238. DOI: 10.2307/1403797 (cited on page 12).

[39] Jason Weston, Sumit Chopra, and Antoine Bordes. 'Memory Networks'. In: arXiv:1410.3916 (Nov. 2015). arXiv:1410.3916 [cs, stat] (cited on page 12).

[40] R. McEliece et al. 'The capacity of the Hopfield associative memory'. In: *IEEE Transactions on Information Theory* 33.4 (1987), pp. 461–482. DOI: 10.1109/TIT.1987.1057328 (cited on page 13).

[41] J. J. Hopfield, D. I. Feinstein, and R. G. Palmer. ''Unlearning' has a stabilizing effect in collective memories'. In: *Nature* 304.5922 (July 1983), pp. 158–159. DOI: 10.1038/304158a0 (cited on page 13).

[42] Dmitry Krotov and John J. Hopfield. 'Dense Associative Memory for Pattern Recognition'. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, pp. 1180–1188 (cited on page 13).

[43] Mete Demircigil et al. 'On a Model of Associative Memory with Huge Storage Capacity'. en. In: *Journal of Statistical Physics* 168.2 (July 2017), pp. 288–299. DOI: 10.1007/s10955-017-1806-y (cited on page 13).

[44] Hubert Ramsauer et al. 'Hopfield Networks is All You Need'. In: arXiv:2008.02217 (Apr. 2021). arXiv:2008.02217 [cs, stat] (cited on page 14).

[45] L. F. Abbott. 'Lapicque's introduction of the integrate-and-fire model neuron (1907)'. In: *Brain Research Bulletin* 50 (1999), pp. 303–304 (cited on page 14).

[46] E.M. Izhikevich. 'Simple model of spiking neurons'. en. In: *IEEE Transactions on Neural Networks* 14.6 (Nov. 2003), pp. 1569–1572. DOI: 10.1109/TNN.2003.820440 (cited on page 14).

[47] Romain Brette and Wulfram Gerstner. 'Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity'. en. In: *Journal of Neurophysiology* 94.5 (Nov. 2005), pp. 3637–3642. DOI: 10.1152/jn.00686.2005 (cited on page 14).

[48] Hélène Paugam-Moisy. 'Spiking Neuron Networks A survey'. In: (2006) (cited on page 14).

[49] Guo-qiang Bi and Mu-ming Poo. 'Synaptic Modification by Correlated Activity: Hebb's Postulate Revisited'. en. In: *Annual Review of Neuroscience* 24.1 (Mar. 2001), pp. 139–166. DOI: 10.1146/annurev.neuro.24.1.139 (cited on page 14).

[50] Saeed Reza Kheradpisheh et al. 'STDP-based spiking deep convolutional neural networks for object recognition'. In: *Neural Networks* 99 (2018), pp. 56–67. DOI: https://doi.org/10.1016/j.neunet.2017.12.005 (cited on page 14).

[51] Herbert Jaeger. 'The" echo state" approach to analysing and training recurrent neural networks-with an erratum note''. In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148 (Jan. 2001) (cited on page 15).

[52] Wolfgang Maass, Thomas Natschläger, and Henry Markram. 'Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations'. en. In: *Neural Computation* 14.11 (Nov. 2002), pp. 2531–2560. DOI: 10.1162/089976602760407955 (cited on page 15).

[53] Zoran Konkoli. 'Reservoir Computing'. en. In: *Unconventional Computing*. Ed. by Andrew Adamatzky. New York, NY: Springer US, 2018, pp. 619–629. DOI: 10.1007/978-1-4939-6883-1_683 (cited on page 15).

[54] Gouhei Tanaka et al. 'Recent advances in physical reservoir computing: A review'. en. In: *Neural Networks* 115 (July 2019), pp. 100–123. DOI: 10.1016/j.neunet.2019.03.005 (cited on page 15).

[55] P. Erdös and A. Rényi. 'On Random Graphs I'. In: *Publicationes Mathematicae Debrecen* 6 (1959), p. 290 (cited on page 15).

[56] Mantas Lukoševičius. 'A Practical Guide to Applying Echo State Networks'. en. In: *Neural Networks: Tricks of the Trade*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Vol. 7700. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 659–686. DOI: 10.1007/978-3-642-35289-8_36 (cited on page 15).

[57] John D. McPherson et al. 'A physical map of the human genome'. In: *Nature* 409.6822 (Feb. 2001), pp. 934–941. DOI: 10.1038/35057157 (cited on page 17).

[58] A.N. Kolmogorov. 'On tables of random numbers'. en. In: *Theoretical Computer Science* 207.2 (Nov. 1998), pp. 387–395. DOI: 10.1016/S0304-3975(98)00075-9 (cited on page 17).

[59] D. J. Willshaw and Christoph Von Der Malsburg. 'How patterned neural connections can be set up by self-organization'. en. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 194.1117 (Nov. 1976), pp. 431–445. DOI: 10.1098/rspb.1976.0087 (cited on page 17).

[60] D. J. Willshaw and Christoph Von Der Malsburg. 'A marker induction mechanism for the establishment of ordered neural mappings: its application to the retinotectal problem'. en. In: *Philosophical Transactions of the Royal Society of London. B, Biological Sciences* 287.1021 (Nov. 1979), pp. 203–243. DOI: 10.1098/rstb.1979.0056 (cited on page 17).

[61] C. von der Malsburg and E Bienenstock. 'A Neural Network for the Retrieval of Superimposed Connection Patterns'. In: *Europhysics Letters (EPL)* 3.11 (June 1987), pp. 1243–1249. DOI: 10.1209/0295-5075/3/11/015 (cited on page 17).

[62] C. von der Malsburg. 'Concerning the Neuronal Code'. In: *Journal of Cognitive Science* 19.4 (Dec. 2018), pp. 511–550. DOI: 10.17791/JCS.2018.19.4.511 (cited on page 17).

[63] Walter J Freeman III and Christine A Skarda. 'Representations: Who needs them?' In: (1990) (cited on page 17).

[64] D. W. Arathorn. *Map-seeking circuits in visual cognition: a computational mechanism for biological and machine vision*. Stanford, Calif: Stanford University Press, 2002 (cited on page 18).

[65] Bruno A. Olshausen, Charles H. Anderson, and David C. Van Essen. 'A multiscale dynamic routing circuit for forming size- and position-invariant object representations'. In: *Journal of Computational Neuroscience* 2.1 (Mar. 1995), pp. 45–62. DOI: 10.1007/BF00962707 (cited on page 18).

[66] Tomas Fernandes and Christoph von der Malsburg. 'Self-Organization of Control Circuits for Invariant Fiber Projections'. en. In: *Neural Computation* 27.5 (May 2015), pp. 1005–1032. DOI: 10.1162/NECO_a_00725 (cited on page 18).

[67] JA Scott Kelso. *Dynamic patterns: The self-organization of brain and behavior*. MIT press, 1995 (cited on page 18).

[68] Birgitta Dresp. 'Seven Properties of Self-Organization in the Human Brain'. In: *Big Data Cogn. Comput.* 4 (2020), p. 10 (cited on page 18).

[69] Marco Dorigo and Luca Maria Gambardella. 'Ant colony system: a cooperative learning approach to the traveling salesman problem'. In: *IEEE Transactions on evolutionary computation* 1.1 (1997), pp. 53–66 (cited on page 18).

[70] Edvinas Byla and Wei Pang. 'DeepSwarm: Optimising Convolutional Neural Networks Using Swarm Intelligence'. en. In: *Advances in Computational Intelligence Systems*. Ed. by Zhaojie Ju et al. Vol. 1043. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2020, pp. 119–130. DOI: 10.1007/978-3-030-29933-0_10 (cited on page 18).

[71] Stephen Wolfram. 'Cellular automata as models of complexity'. In: *Nature* 311.5985 (Oct. 1984), pp. 419–424. DOI: 10.1038/311419a0 (cited on page 19).

[72] Gérard Y. Vichniac. 'Simulating physics with cellular automata'. In: *Physica D: Nonlinear Phenomena* 10.1 (1984), pp. 96–116. DOI: https://doi.org/10.1016/0167-2789(84)90253-7 (cited on page 19).

[73] N. H. Wulff and John A. Hertz. 'Learning Cellular Automation Dynamics with Neural Networks'. In: *NIPS*. 1992 (cited on page 19).