Draft:
Deep Learning in Neural Networks: An Overview

Technical Report IDSIA-03-14

Jürgen Schmidhuber
The Swiss AI Lab IDSIA, Galleria 2, 6928 Manno-Lugano
University of Lugano & SUPSI, Switzerland

14 April 2014

Abstract

In recent years, deep neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarises relevant work, much of it from the previous millennium. Shallow and deep learners are distinguished by the depth of their *credit assignment paths*, which are chains of possibly learnable, causal links between actions and effects. I review deep supervised learning (also recapitulating the history of backpropagation), unsupervised learning, reinforcement learning & evolutionary computation, and indirect search for short programs encoding deep and large networks.

Preface

This is the draft of an invited *Deep Learning* (DL) overview. One of its goals is to assign credit to those who contributed to the present state of the art. I acknowledge the limitations of attempting to achieve this goal. The DL research community itself may be viewed as a continually evolving, deep network of scientists who have influenced each other in complex ways. Starting from recent DL results, I tried to trace back the origins of relevant ideas through the past half century and beyond, sometimes using "local search" to follow citations of citations backwards in time. Since not all DL publications properly acknowledge earlier relevant work, additional global search strategies were employed, aided by consulting numerous neural network experts. As a result, the present draft mostly consists of references (600 entries so far). Nevertheless, through an expert selection bias I may have missed important work. A related bias was surely introduced by my special familiarity with the work of my own DL research group in the past quarter-century. For these reasons, the present draft should be viewed as merely a snapshot of an ongoing credit assignment process. To help improve it, please do not hesitate to send corrections and suggestions to *juergen@idsia.ch*.

Contents

1 Introduction **3**

2 Event-Oriented Notation for Activation Spreading in FNNs/RNNs **3**

3 Depth of Credit Assignment Paths (CAPs) and of Problems **4**

4 Recurring Themes of Deep Learning (DL) **5**
4.1 Dynamic Programming (DP) for DL . . . 5
4.2 Unsupervised Learning (UL) Facilitating Supervised Learning (SL) and RL . . . 5
4.3 Occam's Razor: Compression and Minimal Description Length (MDL) . . . 6
4.4 Learning Hierarchical Representations in Deep SL, UL, RL . . . 6
4.5 Fast GPU Hardware for DL NN . . . 6

5 Supervised NNs, Some Helped by Unsupervised NNs **6**
5.1 1940s and Earlier . . . 7
5.2 Around 1960: More Neurobiological Inspiration for DL . . . 7
5.3 1979: Convolution + Weight Replication + Winner-Take-All (WTA) . . . 7
5.4 1960-1981 and Beyond: Development of Backpropagation (BP) for NNs . . . 8
5.4.1 BP for Weight-Sharing FNNs and RNNs . . . 8
5.5 1989: BP for CNNs . . . 9
5.6 Late 1980s-2000: Improvements of NNs . . . 9
5.6.1 Ideas for Dealing with Long Time Lags and Deep CAPs . . . 9
5.6.2 Better BP Through Advanced Gradient Descent . . . 9
5.6.3 Discovering Low-Complexity, Problem-Solving NNs . . . 10
5.6.4 Potential Benefits of UL for SL . . . 10
5.7 1987: UL Through Autoencoder (AE) Hierarchies . . . 11
5.8 1991: Fundamental Deep Learning Problem of Gradient Descent . . . 12
5.9 1991-: Deep Hierarchy of Recurrent NNs . . . 12
5.10 1994: Contest-Winning Not So Deep NNs . . . 13
5.11 1995-: Supervised Deep Recurrent Learner (LSTM RNN) . . . 13
5.12 1999: Max-Pooling (MP) . . . 15
5.13 2003: More Contest-Winning/Record-Setting, Often Not So Deep NNs . . . 15
5.14 2006: Deep Belief Networks / Improved CNNs / GPU-CNNs . . . 15
5.15 2009: First Official Competitions Won by Deep Learning RNNs . . . 16
5.16 2010: Plain Backprop (+ Distortions) on GPU Yields Excellent Results . . . 16
5.17 2011: Max-Pooling (MP) CNNs on GPU / Superhuman Vision Performance . . . 16
5.18 2011: Hessian-Free Optimization for RNNs . . . 17
5.19 2012: First Contests Won on Object Detection / Segmentation / ImageNet . . . 17
5.20 2013: More Contests and Benchmark Records . . . 18
5.20.1 Currently Successful Supervised Techniques: LSTM RNNs / GPU-MPCNNs . . . 19
5.21 Recent Tricks for Improving SL Deep NNs (Compare Sec. 5.6.2, 5.6.3) . . . 19
5.22 Consequences for Neuroscience . . . 19

6 DL in FNNs and RNNs for Reinforcement Learning (RL) **20**
6.1 RL Through NN World Models Yields RNNs With Deep CAPs . . . 20
6.2 Deep FNNs for Traditional RL and Markov Decision Processes (MDPs) . . . 21
6.3 Deep RL RNNs for Partially Observable MDPs (POMDPs) . . . 21
6.4 RL Facilitated by UL . . . 22
6.5 Deep Hierarchical RL (HRL) and Subgoal Learning . . . 22
6.6 Deep RL by Direct NN Search / Policy Gradients / Evolution . . . 22
6.7 Deep RL by Compressed NN Search . . . 23
6.8 Universal RL . . . 23

7 Conclusion **24**

1 Introduction

Which modifiable components of a learning system are responsible for its success or failure? What changes to them improve performance? This has been called the *fundamental credit assignment problem* (Minsky, 1963). There are general credit assignment methods for *universal problem solvers* that are time-optimal in various theoretical senses (Sec. 6.8). The present survey, however, will focus on the narrower, but now commercially important, subfield of *Deep Learning (DL) in Artificial Neural Networks* (NNs). We are interested in accurate credit assignment across possibly *many*, often nonlinear, computational stages of NNs.

Shallow NN-like models have been around for many decades if not centuries (Sec. 5.1). NNs with several successive non-linear layers of neurons date back at least to the late 1970s (Sec. 5.4). An efficient gradient descent method for teacher-based *Supervised Learning* (SL) in discrete, differentiable networks of arbitrary depth called *backpropagation (BP)* was developed in the 1960s and 1970s, and applied to NNs in 1981 (Sec. 5.4). BP-based training of *deep* NNs with *many* layers, however, had been found to be difficult in practice by the late 1980s (Sec. 5.6), and become an explicit research subject by the early 1990s (Sec. 5.8). DL became practically feasible to some extent through the help of *Unsupervised Learning* (UL), e.g., (Sec. 5.9, 5.14). The 1990s and 2000s also saw many improvements of purely supervised DL (Sec. 5). In the new millennium, deep NNs have finally attracted wide-spread attention, mainly by outperforming alternative machine learning methods such as kernel machines (Vapnik, 1995; Schölkopf et al., 1998) in numerous important applications. In fact, supervised deep NNs have won numerous recent official international pattern recognition competitions (e.g., Sec. 5.15, 5.17, 5.19, 5.20), achieving the first superhuman visual pattern recognition results in limited domains (Sec. 5.17). Deep NNs also have become relevant for the more general field of *Reinforcement Learning* (RL) where there is no supervising teacher (Sec. 6).

Both *feedforward* (acyclic) NNs (FNNs) and *recurrent* (cyclic) NNs (RNNs) have won contests (Sec. 5.10, 5.13, 5.15, 5.17, 5.19, 5.20). RNNs are general computers more powerful than FNNs, and can in principle encode memories of arbitrary sequences of input patterns, e.g., (Siegelmann and Sontag, 1991; Schmidhuber, 1990a). The rest of this paper is structured as follows. Sec. 2 introduces a compact, event-oriented notation that is simple yet general enough to accommodate both FNNs and RNNs. Sec. 3 introduces the concept of *Credit Assignment Paths* (CAPs) to measure whether learning in a given NN application is of the *deep* or *shallow* type. Sec. 4 lists recurring themes of DL in SL, UL, and RL. Sec. 5 focuses on SL and UL, and on how UL can facilitate SL, although pure SL has become dominant in recent competitions (Sec. 5.15-5.20). Sec. 5 is arranged in a historical timeline format with subsections on important inspirations and technical contributions. Sec. 6 on deep RL discusses not only traditional *Dynamic Programming* (DP)-based RL combined with gradient-based search techniques for SL or UL in deep NNs, but also general methods for direct and indirect search in the weight space of deep FNNs and RNNs, including successful policy gradient and evolutionary methods.

2 Event-Oriented Notation for Activation Spreading in FNNs/RNNs

Throughout this paper, let i, j, k, t, p, q, r denote positive integer variables assuming ranges implicit in the given contexts. Let n, m, T denote positive integer constants.

An NN's topology may change over time, e.g., (Fahlman, 1991; Fritzke, 1994). At any given moment, it can be described as a finite subset of units (or nodes or neurons) $N = \{u_1, u_2, \ldots, \}$ and a finite set $H \subseteq N \times N$ of directed edges or connections between nodes. FNNs are acyclic graphs, RNNs cyclic. The first (input) layer is the set of input units, a subset of N. In FNNs, the k-th layer ($k > 1$) is the set of all nodes $u \in N$ such that there is an edge path of length $k - 1$ (but no longer path) between some input unit and u. (There may be shortcut connections between distant layers.)

The NN's behavior or program is determined by a set of real-valued, possibly modifiable, parameters or weights w_i ($i = 1, \ldots, n$). We now focus on a single finite *episode* or *epoch* of information processing and activation spreading, without learning through weight changes. The following slightly unconventional notation is designed to compactly describe what is happening *during the runtime* of the system.

During an episode, there is a *partially causal sequence* $x_t (t = 1, \ldots, T)$ of real values that I call

events. Each x_t is either an input set by the environment, or the activation of a unit that may directly depend on other $x_k(k < t)$ through a current NN topology-dependent set in_t of indices k representing incoming causal connections or links. Let the function v encode topology information and map such event index pairs (k, t) to weight indices. For example, in the non-input case we may have $x_t = f_t(net_t)$ with real-valued $net_t = \sum_{k \in in_t} x_k w_{v(k,t)}$ (additive case) or $net_t = \prod_{k \in in_t} x_k w_{v(k,t)}$ (multiplicative case), where f_t is a typically nonlinear real-valued function such as $tanh$. In many recent competition-winning NNs (Sec. 5.17, 5.19, 5.20) there also are events of the type $x_t = max_{k \in in_t}(x_k)$. x_t may directly affect certain $x_k(k > t)$ through outgoing connections or links represented through a current set out_t of indices k with $t \in in_k$. Some non-input events are called *output events*.

Note that many of the x_t may refer to different, time-varying activations of the *same* unit in sequence-processing RNNs, e.g., (Williams, 1989)(*"unfolding in time"*), or also in FNNs sequentially exposed to time-varying input patterns of a large training set encoded as input events. During an episode, the same weight may get reused over and over again in topology-dependent ways, e.g., in RNNs, or in convolutional NNs (Sec. 5.3, 5.5). I call this weight sharing *across space and/or time*. Weight sharing may greatly reduce the NN's descriptive complexity, which is the number of bits of information required to describe the NN.

In *Supervised Learning* (SL), certain NN output events x_t may be associated with teacher-given, real-valued labels or targets d_t yielding errors e_t, e.g., $e_t = 1/2(x_t - d_t)^2$. A typical goal of supervised NN training is to find weights that yield episodes with small total error E, the sum of all such e_t. The hope is that the NN will generalize well in later episodes, causing only small errors on previously unseen sequences of input events. Many alternative error functions for SL and UL are possible.

SL assumes that input events are independent of earlier output events (which may affect the environment through actions causing subsequent perceptions). This simplifying assumption does not hold in the broader fields of *Sequential Decision Making* and *Reinforcement Learning* (RL) (Kaelbling et al., 1996; Sutton and Barto, 1998; Hutter, 2005) (Sec. 6). In RL, some of the input events may encode real-valued reward signals given by the environment, and a typical goal is to find weights that yield episodes with a high sum of reward signals, through sequences of appropriate output actions.

Sec. 5.4 will use the notation above to compactly describe a central algorithm of DL, namely, back-propagation (BP) for supervised weight-sharing FNNs and RNNs. (FNNs may be viewed as RNNs with certain fixed zero weights.) Sec. 6 will address the more general RL case.

3 Depth of Credit Assignment Paths (CAPs) and of Problems

To measure whether credit assignment in a given NN application is of the *deep* or *shallow* type, I introduce the concept of *Credit Assignment Paths* or CAPs, which are chains of possibly causal links between events.

Let us first focus on SL. Consider two events x_p and x_q $(1 \leq p < q \leq T)$. Depending on the application, they may have a *Potential Direct Causal Connection* (PDCC) expressed by the Boolean predicate $pdcc(p, q)$, which is true if and only if $p \in in_q$. Then the 2-element list (p, q) is a (minimal) CAP from p to q. A learning algorithm may be allowed to change $w_{v(p,q)}$ to improve performance in future episodes.

More general, possibly indirect, *Potential Causal Connections* (PCC) are expressed by the recursively defined Boolean predicate $pcc(p, q)$, which in the SL case is true only if $pdcc(p, q)$, or if $pcc(p, k)$ for some k and $pdcc(k, q)$. In the latter case, appending q to any CAP from p to k yields a CAP from p to q (this is a recursive definition, too). The set of such CAPs may be large but is finite. Note that the same weight may affect many different PDCCs between successive events listed by a given CAP, e.g., in the case of RNNs, or weight-sharing FNNs.

Suppose a CAP has the form $(\ldots, k, t, \ldots, q)$, where k and t (possibly $t = q$) are the first successive elements with *modifiable* $w_{v(k,t)}$. Then the length of the suffix list $(t, \ldots, q)$ is called the CAP's *depth* (which is 0 if there are no modifiable links at all). This depth limits how far backwards credit assignment can move down the causal chain to find a modifiable weight.[1]

Suppose an episode and its event sequence $x_1, \ldots, x_T$ satisfy a computable criterion used to decide whether a given problem has been solved (e.g., total error E below some threshold). Then the set of used weights is called a *solution* to the problem, and the depth of the deepest CAP within the sequence is

[1] An alternative would be to count only *modifiable* links when measuring depth. In many typical NN applications this would not make a difference, but in some it would, e.g., Sec. 6.1.

called the *solution's depth*. There may be other solutions (yielding different event sequences) with different depths. Given some fixed FNN or RNN topology, the smallest depth of any solution is called the *problem's depth*. By definition, problems of depth > 10 require *Very Deep Learning*.

Sometimes we also speak of the *depth* of an architecture: SL FNNs with fixed topology imply a problem-independent maximal problem depth, typically the number of non-input layers. Similar for certain SL RNNs (Maass et al., 2002; Jaeger, 2001, 2004) with fixed weights for all connections except those to output units—their maximal problem depth is 1, because only the final links in the corresponding CAPs are modifiable.

Note that the definitions above are solely based on the depths of causal chains, and agnostic of the temporal distance between events. For example, *shallow* FNNs perceiving large "time windows" of input events may correctly classify *long* input sequences through appropriate output events, and thus solve *shallow* problems involving *long* time lags between relevant events.

The *difficulty* of a problem may have little to do with its depth. Some NNs can quickly learn to solve certain deep problems, e.g., through random weight guessing (Sec. 5.8) or other types of direct search (Sec. 6.6) or indirect search (Sec. 6.7) in weight space, or through collapsing sequences of (non)linear operations into a single (non)linear operation, or through training an NN first on shallow problems whose solutions may then generalize to deep problems,

Above we have focused on SL. In the more general case of RL in unknown environments, $pcc(p, q)$ is also true if x_p is an output event and x_q any later input event—any action may affect the environment and thus any later perception. (In the real world, the environment may even influence *non-input* events computed on a physical hardware entangled with the entire universe, but this is ignored here.) It is possible to model and replace such unmodifiable *environmental* PCCs through a part of the NN that has already learned to predict (through some of its units) input events from former input events and actions (Sec. 6.1). Its weights are frozen, but can help to assign credit to other, still modifiable weights used to compute actions (Sec. 6.1). This approach may lead to very deep CAPs though.

Some DL research is about automatically rephrasing problems such that their depth is reduced (Sec. 4). In particular, sometimes UL is used to make SL problems less deep, e.g., Sec. 5.9. Often *Dynamic Programming* (Sec. 4.1) is used to facilitate certain traditional RL problems, e.g., Sec. 6.2. Sec. 5 focuses on CAPs for SL, Sec. 6 on the more complex case of RL.

4 Recurring Themes of Deep Learning (DL)

4.1 Dynamic Programming (DP) for DL

One recurring theme of DL is *Dynamic Programming* (DP) (Bellman, 1957), which can help to facilitate credit assignment under certain assumptions. For example, in SL NNs, backpropagation itself can be viewed as a DP-derived method (Sec. 5.4). In traditional RL based on strong Markovian assumptions, DP-derived methods can help to greatly reduce problem depth (Sec. 6.2). DP algorithms are also essential for systems that combine graphical models (Dempster et al., 1977) such as *Hidden Markov Models* (HMMs) and NNs, e.g., (Bottou, 1991; Bengio, 1991; Bourlard and Morgan, 1994; Jordan and Sejnowski, 2001; Bishop, 2006; Dahl et al., 2012; Hinton et al., 2012a).

4.2 Unsupervised Learning (UL) Facilitating Supervised Learning (SL) and RL

Another recurring theme is how UL can facilitate both SL (Sec. 5) and RL (Sec. 6). UL (Sec. 5.6.4) is normally used to encode raw incoming data such as video or speech streams in a form that is more convenient for subsequent goal-directed learning. In particular, codes that describe the original data in a less redundant or more compact way can be fed into SL (Sec. 5.9, 5.14) or RL machines (Sec. 6.4), whose search spaces may thus become smaller (and whose CAPs shallower) than those necessary for dealing with the raw data. UL is closely connected to the topics of *regularization* and compression (Sec. 4.3).

4.3 Occam's Razor: Compression and Minimal Description Length (MDL)

Occam's razor favors simple solutions over complex ones. Given some programming language, the principle of *Minimal Description Length* (MDL) can be used to measure the complexity of a solution candidate by the length of the shortest program that computes it, e.g., (Solomonoff, 1964; Kolmogorov, 1965b; Chaitin, 1966; Wallace and Boulton, 1968; Levin, 1973a; Rissanen, 1986; Li and Vitányi, 1997). Some methods explicitly take into account program runtime (Allender, 1992; Watanabe, 1992; Schmidhuber, 2002, 1995); many consider only programs with constant runtime, e.g., (Rissanen, 1986; Hinton and van Camp, 1993; Li and Vitányi, 1997).

In the NN case, the MDL principle suggests that low NN weight complexity corresponds to high NN probability in the Bayesian view, e.g., (MacKay, 1992; Buntine and Weigend, 1991), and to high generalization performance. Many methods have been proposed for *regularizing* NNs, that is, searching for solution-computing, low-complexity SL NNs (Sec. 5.6.3) and RL NNs (Sec. 6.7). This is closely related to certain UL methods (Sec. 4.2, 5.6.4).

4.4 Learning Hierarchical Representations in Deep SL, UL, RL

Many methods of *Good Old-Fashioned Artificial Intelligence* (GOFAI) (Nilsson, 1980) and more recent approaches to AI (Russell and Norvig, 1994) learn hierarchies of more and more abstract data representations. For example, certain methods of syntactic pattern recognition (Fu, 1977) such as *grammar induction* discover hierarchies of formal rules to model observations. The partially (un)supervised *Automated Mathematician / EURISKO* (Lenat, 1983; Lenat and Brown, 1984) continually learns concepts by combining previously learnt concepts. Such hierarchical representation learning (Ring, 1994; Bengio et al., 2013; Deng and Yu, 2014) is also a recurring theme of DL NNs for SL (Sec. 5), UL-aided SL (Sec. 5.7, 5.9, 5.14), and hierarchical RL (Sec. 6.5). Often, abstract hierarchical representations are natural by-products of data compression (Sec. 4.3), e.g., (Sec. 5.9).

4.5 Fast GPU Hardware for DL NN

While the previous millennium saw several attempts at creating fast NN-specific hardware, e.g., (Jackel et al., 1990; Faggin, 1992; Ramacher et al., 1993; Widrow et al., 1994; Heemskerk, 1995; Korkin et al., 1997; Urlbe, 1999), the new millennium enabled a DL NN breakthrough in form of cheap, multi-processor graphics cards or GPUs. GPUs are widely used for video games, a huge and competitive market that drove down hardware prices. GPUs excel at fast matrix and vector multiplications required not only for convincing virtual realities but also for NN training, where they can speed up learning by a factor of 50 and more. Some of the GPU-based FNN implementations (Sec. 5.14 - 5.17) have greatly contributed to recent successes in contests for pattern recognition (Sec. 5.17 - 5.20), image segmentation (Sec. 5.19), and object detection (Sec. 5.19 - 5.20).

5 Supervised NNs, Some Helped by Unsupervised NNs

The main focus of current practical applications is on *Supervised Learning* (SL), which has dominated recent pattern recognition contests (Sec. 5.15-5.20). Several methods, however, use additional *Unsupervised Learning* (UL) to facilitate SL (Sec. 5.7, 5.9, 5.14). It does make sense to treat SL and UL in the same section: often gradient-based methods, such as BP (Sec. 5.4.1), are used to optimize objective functions of both UL and SL, and the boundary between SL and UL may blur, for example, when it comes to time series prediction and sequence classification, e.g., Sec. 5.9, 5.10.

A historical timeline format will help to arrange subsections on important inspirations and technical contributions (although such a subsection may span a time interval of many years). Sec. 5.1 briefly mentions early, shallow NN models since the 1940s, Sec. 5.2 additional early neurobiological inspiration relevant for modern Deep Learning (DL). Sec. 5.3 is about the relatively deep *Neocognitron* NN (1979) which is similar to certain modern deep FNN architectures, as it combines convolutional NNs (CNNs), weight pattern replication, and winner-take-all (WTA) mechanisms. Sec. 5.4 uses the notation of Sec. 2 to compactly describe a central algorithm of DL, namely, *backpropagation* (BP) for supervised weight-sharing

FNNs and RNNs. It also summarizes the history of BP 1960-1981 and beyond. Sec. 5.5 is about applying BP to CNNs, important in today's DL applications. Sec. 5.6 describes problems encountered in the late 1980s with BP for deep NNs, and mentions several ideas from the previous millennium to overcome them; Sec. 5.7 mentions a first hierarchical stack of coupled UL-based Auto-Encoders (AEs). Sec. 5.8 explains BP's *Fundamental DL Problem* (of vanishing/exploding gradients) discovered in 1991. Sec. 5.9 explains how a deep RNN stack of 1991 (the *History Compressor*) pre-trained by UL helped to solve previously unlearnable DL benchmarks requiring *Credit Assignment Paths* (CAPs, Sec. 3) of depth 1000 and more. Sec. 5.10 mentions a first important contest won by SL NNs in 1994. Sec. 5.11 describes a purely supervised DL RNN (*Long Short-Term Memory*, LSTM) for problems of depth 1000 and more. Sec. 5.12 mentions a particular WTA method called *Max-Pooling* (MP) important in today's DL FNNs. Sec. 5.13 mentions an early contest of 2003 won by an ensemble of shallow NNs, as well as good pattern recognition results with CNNs (2003) and LSTM RNNs. Sec. 5.14 is mostly about *Deep Belief Networks* (DBNs, 2006) and related stacks of *Autoencoders* (AEs, Sec. 5.7) pre-trained by UL to facilitate SL. Sec. 5.15-5.20 focus on official competitions with secret test sets won by DL NNs since 2009, in sequence recognition, image classification, image segmentation, and object detection. Many RNN results depended on LSTM (Sec. 5.11); many FNN results on GPU-based FNN code developed since 2004 (Sec. 5.14, 5.15, 5.16, 5.17), in particular, GPU-MPCNNs (Sec. 5.17).

5.1 1940s and Earlier

NN research started in the 1940s, e.g., (McCulloch and Pitts, 1943; Hebb, 1949); compare also later work (Rosenblatt, 1958, 1962; Widrow and Hoff, 1962; Grossberg, 1969; Kohonen, 1972; von der Malsburg, 1973; Narendra and Thathatchar, 1974; Willshaw and von der Malsburg, 1976; Palm, 1980; Hopfield, 1982). In a sense NNs have been around even longer, since early supervised NNs were essentially variants of linear regression methods going back at least to the early 1800s, e.g., (Gauss, 1821). Early NNs had a maximal CAP depth of 1 (Sec. 3).

5.2 Around 1960: More Neurobiological Inspiration for DL

Simple cells and complex cells were found in the visual cortex, e.g., (Hubel and Wiesel, 1962; Wiesel and Hubel, 1959). These cells fire in response to certain properties of visual sensory inputs, such as the orientation of edges. Complex cells exhibit more spatial invariance than simple cells. This inspired later deep NN architectures (Sec. 5.3) used in certain modern award-winning Deep Learners (Sec. 5.17-5.20).

5.3 1979: Convolution + Weight Replication + Winner-Take-All (WTA)

The *Neocognitron* (Fukushima, 1979, 1980, 2013a) was perhaps the first artificial NN that deserved the attribute *deep*, and the first to incorporate the neurophysiological insights of Sec. 5.2. It introduced *convolutional NNs* (today often called CNNs or convnets), where the (typically rectangular) receptive field of a *convolutional unit* with given weight vector is shifted step by step across a 2-dimensional array of input values, such as the pixels of an image. The resulting 2D array of subsequent activation events of this unit can then provide inputs to higher-level units, and so on. Due to massive *weight replication* (Sec. 2), relatively few parameters may be necessary to describe the behavior of such a *convolutional layer*.

Competition layers have WTA subsets whose maximally active units are the only ones to adopt non-zero activation values. They essentially "down-sample" the competition layer's input. This helps to create units whose responses are insensitive to small image shifts (compare Sec. 5.2).

The Neocognitron is very similar to the architecture of modern, contest-winning, purely *supervised*, feedforward, gradient-based Deep Learners (Sec. 5.17-5.20). Fukushima, however, did not set the weights by supervised backpropagation (Sec. 5.4, 5.5), but by local *un*supervised learning rules, e.g., (Fukushima, 2013b), or by pre-wiring. In that sense he did not care for the DL problem (Sec. 5.8), although his architecture was comparatively deep indeed. He also used spatial averaging (Fukushima, 1980, 2011) instead of max-pooling (MP, Sec. 5.12), today a particularly convenient and popular WTA mechanism. Today's CNN-based DL machines profit a lot from later CNN work, e.g., (LeCun et al., 1989; Ranzato et al., 2007) (Sec. 5.5, 5.14, 5.17).

5.4 1960-1981 and Beyond: Development of Backpropagation (BP) for NNs

The minimisation of errors through gradient descent (Hadamard, 1908) in the parameter space of complex, nonlinear, multi-stage, differentiable, NN-related systems has been discussed at least since the early 1960s, e.g., (Kelley, 1960; Bryson, 1961; Bryson and Denham, 1961; Pontryagin et al., 1961; Dreyfus, 1962; Wilkinson, 1965; Amari, 1967; Bryson and Ho, 1969; Director and Rohrer, 1969; Griewank, 2012), initially within the framework of Euler-LaGrange equations in the *Calculus of Variations*, e.g., (Euler, 1744). Steepest descent in such systems can be performed (Bryson, 1961; Kelley, 1960; Bryson and Ho, 1969) by iterating the ancient chain rule (Leibniz, 1676; L'Hôpital, 1696) in *Dynamic Programming* (DP) style (Bellman, 1957). A simplified derivation uses the chain rule only (Dreyfus, 1962). The systems of the 1960s were already efficient in the DP sense. However, they backpropagated derivative information through standard Jacobian matrix calculations from one "layer" to the previous one, explicitly addressing neither direct links across several layers nor potential additional efficiency gains due to network sparsity (but perhaps such enhancements seemed obvious to the authors).

Explicit, efficient error backpropagation (BP) in arbitrary, discrete, possibly sparsely connected, NN-like networks apparently was first described in 1970 (Linnainmaa, 1970, 1976), albeit without reference to NNs. BP is also known as the reverse mode of automatic differentiation (Griewank, 2012), where the costs of forward activation spreading essentially equal the costs of backward derivative calculation. See early FORTRAN code (Linnainmaa, 1970); compare (Ostrovskii et al., 1971). Efficient BP was soon explicitly used to minimize cost functions by adapting control parameters (weights) (Dreyfus, 1973). Compare some NN-specific discussion (Werbos, 1974, section 5.5.1), a method for multilayer threshold NNs (Bobrowski, 1978), and a computer program for automatically deriving and implementing BP for given differentiable systems (Speelpenning, 1980).

To my knowledge, the first NN-specific application of efficient BP was described in 1981 (Werbos, 1981, 2006). Compare (LeCun, 1985; Parker, 1985; LeCun, 1988). A paper of 1986 significantly contributed to the popularisation of BP (Rumelhart et al., 1986). See generalisations for sequence-processing recurrent NNs, e.g., (Williams, 1989; Robinson and Fallside, 1987; Werbos, 1988; Williams and Zipser, 1988, 1989b,a; Rohwer, 1989; Pearlmutter, 1989; Gherrity, 1989; Williams and Peng, 1990; Schmidhuber, 1992a; Pearlmutter, 1995), also for equilibrium RNNs (Almeida, 1987; Pineda, 1987) with stationary inputs. See also natural gradients (Amari, 1998).

5.4.1 BP for Weight-Sharing FNNs and RNNs

Using the notation of Sec. 2 for weight-sharing FNNs or RNNs, after an episode of activation spreading through differentiable f_t, a single iteration of gradient descent through BP computes changes of all w_i in proportion to $\frac{\partial E}{\partial w_i} = \sum_t \frac{\partial E}{\partial net_t} \frac{\partial net_t}{\partial w_i}$ as in Algorithm 5.4.1 (for the additive case), where each weight w_i is associated with a real-valued variable $\triangle_i$ initialized by 0.

Alg. 5.4.1: One iteration of BP for weight-sharing FNNs or RNNs

for $t = T, \ldots, 1$ **do**
to compute $\frac{\partial E}{\partial net_t}$, inititalize real-valued error signal variable δ_t by 0;
if x_t is an input event then continue with next iteration;
if there is an error e_t then $\delta_t := x_t - d_t$;
add to δ_t the value $\sum_{k \in out_t} w_{v(t,k)} \delta_k$; *(this is the elegant and efficient recursive chain rule application collecting impacts of net_t on future events)*
multiply δ_t by $f'_t(net_t)$;
for all $k \in in_t$ add to $\triangle_{w_{v(k,t)}}$ the value $x_k \delta_t$
end for

Finally, to finish one iteration of steepest descent, change all w_i in proportion to $\triangle_i$ and a small learning rate. The computational costs of the backward (BP) pass are essentially those of the forward pass (Sec. 2). Forward and backward passes are re-iterated until sufficient performance is reached.

As of 2013, this simple BP method is still the central learning algorithm for FNNs and RNNs. Notably, most contest-winning NNs up to 2013 (Sec. 5.10, 5.13, 5.15, 5.17, 5.19, 5.20) did *not* augment supervised

BP by some sort of *un*supervised learning as discussed in Sec. 5.7, 5.9, 5.14.

5.5 1989: BP for CNNs

In 1989, backpropagation (Sec. 5.4) was applied (LeCun et al., 1989, 1990a, 1998) to weight-sharing convolutional neural layers with adaptive connections (compare Sec. 5.3). This combination, augmented by max-pooling (Sec. 5.12), and sped up on graphics cards (Sec. 5.17), has become an essential ingredient of many modern, competition-winning, feedforward, visual Deep Learners (Sec. 5.17-5.19). This work also introduced the MNIST data set of handwritten digits (LeCun et al., 1989), which over time has become perhaps the most famous benchmark of Machine Learning. A CNN of depth 5 achieved good performance on MNIST (LeCun et al., 1990a); similar CNNs were used commercially in the 1990s.

5.6 Late 1980s-2000: Improvements of NNs

By the late 1980s it seemed clear that BP by itself (Sec. 5.4) was no panacea. Most FNN applications focused on FNNs with few hidden layers. Many practitioners found solace in a theorem (Kolmogorov, 1965a) stating that an NN with a single layer of enough hidden units can approximate any multivariate continous function with arbitrary accuracy. Additional hidden layers often did not seem to offer empirical benefits.

Likewise, most RNN applications did not require backpropagating errors far. Many researchers helped their RNNs by first training them on shallow problems (Sec. 3) whose solutions then generalized to deeper problems. In fact, some popular RNN algorithms restricted credit assignment to a single step backwards (Elman, 1988; Jordan, 1986), also in more recent studies (Jaeger, 2002; Maass et al., 2002; Jaeger, 2004).

Generally speaking, although BP allows for deep problems in principle, it seemed to work only for *shallow* problems. The late 1980s and early 1990s saw a few ideas with a potential to overcome this problem, which was fully understood only in 1991 (Sec. 5.8).

5.6.1 Ideas for Dealing with Long Time Lags and Deep CAPs

To deal with long time lags between relevant events, several sequence processing methods were proposed, including *focused BP* for RNNs (Mozer, 1989, 1992), *Time-Delay Neural Networks* (Lang et al., 1990) and their adaptive extension (Bodenhausen and Waibel, 1991), *NARX RNNs* (Lin et al., 1995, 1996), certain hierarchical RNNs (Hihi and Bengio, 1996), RL economies in RNNs with WTA units (Schmidhuber, 1989b), and other methods, e.g., (Ring, 1993, 1994; Plate, 1993; de Vries and Principe, 1991; Sun et al., 1993a; Bengio et al., 1994). However, these algorithms either worked for shallow CAPs only, could not generalize to unseen CAP depths, had problems with greatly varying time lags between relevant events, needed external fine tuning of delay constants, or suffered from other problems. In fact, it turned out that certain simple but deep benchmark problems used to evaluate such methods are more quickly solved by randomly guessing RNN weights until a solution is found (Schmidhuber and Hochreiter, 1996; Schmidhuber et al., 2001).

The *Neural Heat Exchanger* (Schmidhuber, 1990c) consists of two parallel deep FNNs with opposite flow directions. Input patterns enter the first FNN and are propagated "up". Desired outputs (targets) enter the "opposite" FNN and are propagated "down". Using a local learning rule, each layer in each net tries to be similar (in information content) to the preceding layer and to the adjacent layer of the other net. The input entering the first net slowly "heats up" to become the target. The target entering the opposite net slowly "cools down" to become the input. The *Helmholtz Machine* (Dayan et al., 1995; Dayan and Hinton, 1996) may be viewed as an unsupervised (Sec. 5.6.4) variant thereof (Peter Dayan, personal communication, 1994).

5.6.2 Better BP Through Advanced Gradient Descent

Numerous improvements of steepest descent through BP (Sec. 5.4) have been proposed. Least-squares methods (Gauss-Newton, Levenberg-Marquardt) (Levenberg, 1944; Marquardt, 1963) and quasi-Newton

methods (Broyden-Fletcher-Goldfarb-Shanno, BFGS) (Broyden et al., 1965; Fletcher and Powell, 1963; Goldfarb, 1970; Shanno, 1970) are computationally too expensive for large NNs. Partial BFGS (Battiti, 1992; Saito and Nakano, 1997) and conjugate gradient (Hestenes and Stiefel, 1952; Møller, 1993) as well as other methods (Solla, 1988; Schmidhuber, 1989a; Cauwenberghs, 1993) provide sometimes useful fast alternatives. To speed up BP, *momentum* was introduced (Rumelhart et al., 1986), ad-hoc constants were added to the slope of the linearized activation function (Fahlman, 1988), or the nonlinearity of the slope was exaggerated (West and Saad, 1995). Only the signs of the error derivatives are taken into account by the successful and widely used BP variant *R-prop* (Riedmiller and Braun, 1993). The local gradient can be normalized based on the NN architecture (Schraudolph and Sejnowski, 1996), through a diagonalized Hessian approach (Becker and Le Cun, 1989), or related efficient methods (Schraudolph, 2002). BP can be treated as a linear least-squares problem (Biegler-König and Bärmann, 1993), where second-order gradient information is passed back to preceding layers. Some algorithms for controlling BP step size adapt a global learning rate (Lapedes and Farber, 1986; Vogl et al., 1988; Battiti, 1989; LeCun et al., 1993; Yu et al., 1995), others individual learning rates for each weight (Jacobs, 1988; Silva and Almeida, 1990). In online learning, where BP is applied after each pattern presentation, the vario-η algorithm (Neuneier and Zimmermann, 1996) sets each weight's learning rate inversely proportional to the empirical standard deviation of its local gradient, thus normalizing the stochastic weight fluctuations. Compare a local online step size adaptation method for nonlinear NNs (Almeida et al., 1997). Many additional tricks for improving NNs have been described, e.g., (Orr and Müller, 1998; Montavon et al., 2012). Compare Sec. 5.6.3 and recent developments mentioned in Sec. 5.21.

5.6.3 Discovering Low-Complexity, Problem-Solving NNs

Many researchers used BP-like methods to search for low-complexity NNs (Sec. 4.3) with high generalization capability. Most approaches are based on strong prior assumptions. For example, weight decay (Hanson and Pratt, 1989; Weigend et al., 1991; Krogh and Hertz, 1992) prefers small weights and can be derived from Gaussian or Laplace weight priors (Hinton and van Camp, 1993); compare (Murray and Edwards, 1993). Some assume that a distribution of networks with many similar weights generated by Gaussian mixtures is "better" *a priori* (Nowlan and Hinton, 1992). Other weight priors are implicit in additional penalty terms (MacKay, 1992) or in methods based on validation sets (Mosteller and Tukey, 1968; Stone, 1974; Eubank, 1988; Hastie and Tibshirani, 1990; Craven and Wahba, 1979; Golub et al., 1979), "final prediction error" (Akaike, 1970), "generalized prediction error" (Moody and Utans, 1994; Moody, 1992); see also (Holden, 1994; Wang et al., 1994; Amari and Murata, 1993; Wang et al., 1994; Guyon et al., 1992; Vapnik, 1992; Wolpert, 1994). Similar for constructive and pruning algorithms, e.g., "sequential network construction" (Fahlman, 1991; Ash, 1989; Moody, 1989), input pruning (Moody, 1992; Nicholas Refenes et al., 1994), unit pruning (White, 1989; Mozer and Smolensky, 1989; Levin et al., 1994), weight pruning, e.g. "optimal brain damage" (LeCun et al., 1990b), "optimal brain surgeon" (Hassibi and Stork, 1993). Compare (Geman et al., 1992). A very general but not always practical approach for discovering low-complexity SL NNs or RL NNs searches among weight matrix-computing programs written in a universal programming language (Schmidhuber, 1995, 1997) (Sec. 6.7). *Flat Minimum Search* (FMS) (Hochreiter and Schmidhuber, 1997a, 1999) searches for a "flat" minimum of the error function: a large connected region in weight space where error is low and remains approximately constant, that is, few bits of information are required to describe low-precision weights with high variance. Compare perturbation tolerance conditions (Minai and Williams, 1994; Murray and Edwards, 1993; Neti et al., 1992; Matsuoka, 1992; Bishop, 1993; Kerlirzin and Vallet, 1993; Carter et al., 1990). An MDL-based, Bayesian argument suggests that flat minima correspond to "simple" NNs and low expected overfitting. Compare Sec. 5.6.4 and more recent developments mentioned in Sec. 5.21.

5.6.4 Potential Benefits of UL for SL

The SL notation of Sec. 2 focused on teacher-given labels d_t. Many papers of the previous millennium, however, were about *unsupervised learning (UL) without a teacher*, e.g., (Hebb, 1949; von der Malsburg, 1973; Kohonen, 1972, 1982, 1988; Willshaw and von der Malsburg, 1976; Grossberg, 1976a,b; Watanabe, 1985; Pearlmutter and Hinton, 1986; Barrow, 1987; Field, 1987; Oja, 1989; Barlow et al., 1989; Baldi and

Hornik, 1989; Rubner and Tavan, 1989; Sanger, 1989; Ritter and Kohonen, 1989; Rubner and Schulten, 1990; Földiák, 1990; Martinetz et al., 1990; Mozer, 1991; Palm, 1992; Atick et al., 1992; Schmidhuber and Prelinger, 1993; Miller, 1994; Saund, 1994; Földiák and Young, 1995; Deco and Parra, 1997); see also post-2000 work, e.g., (Klapper-Rybicka et al., 2001; Wiskott and Sejnowski, 2002; Franzius et al., 2007). Many UL methods are designed to maximize information-theoretic objectives, e.g., (Linsker, 1988; Barlow et al., 1989; MacKay and Miller, 1990; Becker, 1990; Plumbley, 1991; Schmidhuber, 1992b,c; Schraudolph and Sejnowski, 1993; Redlich, 1993; Zemel, 1993; Zemel and Hinton, 1994; Field, 1994; Hinton et al., 1995; Dayan and Zemel, 1995; Amari et al., 1996; Deco and Parra, 1997), and to uncover and disentangle hidden underlying sources of signals, e.g., (Jutten and Herault, 1991; Schuster, 1992; Andrade et al., 1993; Molgedey and Schuster, 1994; Comon, 1994; Cardoso, 1994; Bell and Sejnowski, 1995; Belouchrani et al., 1997; Hyvärinen and Oja, 2000). Many UL methods automatically and robustly generate distributed, sparse representations of input patterns (Földiák, 1990; Olshausen and Field, 1996; Hinton and Ghahramani, 1997; Lewicki and Olshausen, 1998; Hyvarinen et al., 1999; Hochreiter and Schmidhuber, 1999) through well-known feature detectors, such as orientation sensitive edge detectors and off-center-on-surround-like structures, e.g., (Olshausen and Field, 1996; Schmidhuber et al., 1996), thus extracting simple features related to those considered useful for image pre-processing and compression, and also to those observed in early visual pre-processing stages of biological systems.

UL can help to encode input data in a form advantageous for further processing. In the context of DL, one important goal of UL is redundancy reduction. Ideally, given an ensemble of input patterns, redundancy reduction through a deep NN will create a *factorial code* (a code with statistically independent components) of the ensemble (Barlow et al., 1989; Barlow, 1989), to disentangle the unknown factors of variation, e.g., (Bengio et al., 2013). Such codes may be sparse and can be advantageous for (1) data compression, (2) speeding up subsequent BP (Becker, 1991), (3) trivialising the task of subsequent naive yet optimal Bayes classifiers (Schmidhuber et al., 1996).

Most early UL FNNs had a single layer. Methods for deeper FNNs include nonlinear *Autoencoders* (AEs) with more than 3 (e.g., 5) layers (Kramer, 1991; Oja, 1991; DeMers and Cottrell, 1993). Such an AE NN (Rumelhart et al., 1986) can be trained to map input patterns to themselves, for example, by compactly encoding them through activations of units of a narrow bottleneck hidden layer. Other non-linear methods include *Predictability Minimization* (PM) (Schmidhuber, 1992c), where non-linear feature detectors fight non-linear predictors, trying to become both informative and as unpredictable as possible, and LOCOCODE (Hochreiter and Schmidhuber, 1999), where FMS (Sec. 5.6.3) is applied to find low-complexity AEs with low-precision weights describable by few bits of information, often yielding sparse or factorial codes.

5.7 1987: UL Through Autoencoder (AE) Hierarchies

Perhaps the first work to study potential benefits of UL-based pre-training was published in 1987. It proposed unsupervised AE hierarchies (Ballard, 1987), closely related to certain post-2000 feedforward Deep Learners based on UL (Sec. 5.14). The lowest-level AE NN with a single hidden layer is trained to map input patterns to themselves. Its hidden layer codes are then fed into a higher-level AE of the same type, and so on. The hope is that the codes in the hidden AE layers have properties that facilitate subsequent learning. In one experiment, a particular AE-specific learning algorithm (different from traditional BP of Sec. 5.4.1) was used to learn a mapping in an AE stack pre-trained by this type of UL (Ballard, 1987). This was faster than learning an equivalent mapping by BP through a single deeper AE without pre-training. On the other hand, the task did not really require a deep AE, that is, the benefits of UL were not that obvious from this experiment. Compare an early survey (Hinton, 1989) and the somewhat related *Recursive Auto-Associative Memory* (RAAM) (Pollack, 1988, 1990; Melnik et al., 2000), originally used to encode linguistic structures, but later also as an unsupervised pre-processor to facilitate deep credit assignment for RL (Gisslen et al., 2011) (Sec. 6.4).

In principle, many UL methods (Sec. 5.6.4) could be stacked like the AEs above, the RNNs of Sec. 5.9, or the DBNs of Sec. 5.9, to facilitate subsequent SL. Compare *Stacked Generalization* (Wolpert, 1992).

5.8 1991: Fundamental Deep Learning Problem of Gradient Descent

A diploma thesis (Hochreiter, 1991) represented a milestone of explicit DL research. As mentioned in Sec. 5.6, by the late 1980s, experiments had indicated that traditional deep feedforward or recurrent networks are hard to train by backpropagation (BP) (Sec. 5.4). Hochreiter's work formally identified a major reason: Typical deep NNs suffer from the now famous problem of vanishing or exploding gradients. With standard activation functions (Sec. 1), cumulative backpropagated error signals (Sec. 5.4.1) either shrink rapidly, or grow out of bounds. In fact, they decay exponentially in the number of layers or CAP depth (Sec. 3), or they explode. This is also known as the *long time lag problem*. Much subsequent DL research of the 1990s and 2000s was motivated by this insight. Compare (Bengio et al., 1994), who also studied basins of attraction and their stability under noise from a dynamical systems point of view: either the dynamics are not robust to noise, or the gradients vanish; see also (Hochreiter et al., 2001a; Tiňo and Hammer, 2004). Over the years, several ways of partially overcoming the *Fundamental Deep Learning Problem* were explored:

I A Very Deep Learner of 1991 (Sec. 5.9) alleviates the problem through unsupervised pre-training for a hierarchy of RNNs. This greatly facilitates subsequent supervised credit assignment through BP (Sec. 5.4). Compare conceptually related AE stacks (Sec. 5.7) and Deep Belief Nets (Sec. 5.14) for the FNN case.

II LSTM-like networks (Sec. 5.11, 5.15, 5.20) alleviate the problem through a special architecture unaffected by it.

III Today's GPU-based computers have a million times the computational power of desktop machines of the early 1990s. This allows for propagating errors a few layers further down within reasonable time, even in traditional NNs (Sec. 5.16). That is basically what is winning many of the image recognition competitions now (Sec. 5.17, 5.19, 5.20). (Although this does not really overcome the problem in a fundamental way.)

IV Hessian-free optimization (Sec. 5.6.2) can alleviate the problem for FNNs (Møller, 1993; Pearlmutter, 1994; Schraudolph, 2002; Martens, 2010) (Sec. 5.6.2) and RNNs (Martens and Sutskever, 2011) (Sec. 5.18).

V The space of NN weight matrices can also be searched without relying on error gradients, thus avoiding the *Fundamental Deep Learning Problem* altogether. Recall from Sec. 5.7 that random weight guessing sometimes works better than more sophisticated methods. Certain more complex problems are better solved by using *Universal Search* (Levin, 1973b) for weight matrix-computing programs written in a universal programming language (Schmidhuber, 1995, 1997). Some are better solved by using linear methods to obtain optimal weights for connections to output events, and evolving weights of connections to other events—this is called *Evolino* (Schmidhuber et al., 2007). Direct search methods are relevant not only for SL but also for more general RL, and are discussed in more detail in Sec. 6.6.

5.9 1991-: Deep Hierarchy of Recurrent NNs

A working *Very Deep Learner* (Sec. 3) of 1991 (Schmidhuber, 1992b, 2013a) could perform credit assignment across hundreds of nonlinear operators or neural layers, by using *unsupervised pre-training* for a stack of RNNs.

The basic idea is still relevant today. Each RNN is trained for a while in unsupervised fashion to predict its next input; e.g., (Connor et al., 1994). From then on, only unexpected inputs (errors) convey new information and get fed to the next higher RNN which thus ticks on a slower, self-organising time scale. It can easily be shown that no information gets lost. It just gets compressed (much of machine learning is essentially about compression, e.g., Sec. 4.3, 5.6.3, 6.7). For each individual input sequence, we get a series of less and less redundant encodings in deeper and deeper levels of this *History Compressor*, which can compress data in both space (like feedforward NNs) and time. There also is a continuous variant (Schmidhuber et al., 1993).

The RNN stack is essentially a deep generative model of the data, which can be reconstructed from its compressed form. Adding another RNN to the stack improves a bound on the data's description length—equivalent to the negative logarithm of its probability (Huffman, 1952; Shannon, 1948)—as long as there is remaining local learnable predictability in the data representation on the corresponding level of the hierarchy.

The system was able to learn many previously unlearnable DL tasks. One ancient illustrative DL experiment (Schmidhuber, 1993b) required *Credit Assignment Paths* (CAPs, Sec. 3) of depth 1200. The top level code of the initially unsupervised RNN stack, however, got so compact that (previously infeasible) sequence classification through additional SL became possible. Essentially the system used UL to greatly reduce CAP depth.

There is a way of compressing higher levels down into lower levels, thus fully or partially collapsing the RNN stack. The trick is to retrain a lower-level RNN to continually imitate (predict) the hidden units of an already trained, slower, higher-level RNN (the "conscious" chunker), through additional predictive output neurons (Schmidhuber, 1992b). This helps the lower RNN (the "automatizer") to develop appropriate, rarely changing memories that may bridge very long time lags. Again, this procedure can greatly reduce the required depth of the BP process.

The 1991 system was a working Deep Learner in the modern post-2000 sense, and also a first *Neural Hierarchical Temporal Memory* (HTM). It is conceptually similar to AE hierarchies (Sec. 5.7) and later *Deep Belief Networks* (DBNs) (Sec. 5.14), but more general in the sense that it uses sequence-processing RNNs instead of FNNs with unchanging inputs. More recently, well-known entrepreneurs (Hawkins and George, 2006; Kurzweil, 2012) also got interested in HTMs. Stacks of RNNs were used in later work with great success, e.g., Sec. 5.11, 5.15, 5.20. Clockwork RNNs (Koutník et al., 2014) also consist of interacting RNN modules with different clock rates, but do not require UL to set those rates.

5.10 1994: Contest-Winning Not So Deep NNs

Back in the 1990s, certain NNs already won certain controlled pattern recognition contests with secret test sets. Notably, an NN with internal delay lines won the Santa Fe time-series competition on chaotic intensity pulsations of an NH3 laser (Wan, 1994; Weigend and Gershenfeld, 1993). No very deep CAPs (Sec. 3) were needed though.

5.11 1995-: Supervised Deep Recurrent Learner (LSTM RNN)

Supervised *Long Short-Term Memory* (LSTM) RNN (Hochreiter and Schmidhuber, 1997b; Gers et al., 2000; Pérez-Ortiz et al., 2003) could eventually perform similar feats as the deep RNN hierarchy of 1991 (Sec. 5.9), overcoming the *Fundamental Deep Learning Problem* (Sec. 5.8) without any unsupervised pre-training. LSTM could also learn DL tasks *without* local sequence predictability (and thus *un*learnable by the partially unsupervised 1991 History Compressor, Sec. 5.9), dealing with *Credit Assignment Paths* (CAPs, Sec. 3) of depth 1000 and more, e.g., (Gers and Schmidhuber, 2001; Gers et al., 2002).

The basic LSTM idea is very simple. Some of the units are called constant error carrousels (CECs). Each CEC uses as an activation function f, the identity function, and has a connection to itself with fixed weight of 1.0. Due to f's constant derivative of 1.0, errors backpropagated through a CEC cannot vanish or explode but stay as they are (unless they "flow out" of the CEC to other units). This is the main reason why LSTM nets can learn to discover the importance of (and memorize) events that happened thousands of discrete time steps ago, while previous RNNs already failed in case of minimal time lags of 10 steps. Some LSTM variants also use *modifiable* self-connections of CECs (Gers and Schmidhuber, 2001).

A CEC is connected to several *nonlinear* adaptive units (often including multiplicative gates) needed for learning nonlinear behavior (often using error signals propagated far back in time through a CEC). Many different LSTM variants and topologies are allowed. It is possible to evolve good problem-specific topologies (Bayer et al., 2009). Compare also more recent RNN algorithms able to deal with long time lags (Schäfer et al., 2006; Zimmermann et al., 2012; Koutník et al., 2014).

To a certain extent, LSTM is biologically plausible (O'Reilly, 2003). LSTM learned to solve many previously unlearnable DL tasks involving: Recognition of the temporal order of widely separated events in

noisy input streams; Robust storage of high-precision real numbers across extended time intervals; Arithmetic operations on continuous input streams; Extraction of information conveyed by the temporal distance between events; Recognition of temporally extended patterns in noisy input sequences (Hochreiter and Schmidhuber, 1997b; Gers et al., 2000); Stable generation of precisely timed rhythms (Gers and Schmidhuber, 2000), smooth and non-smooth periodic trajectories. LSTM clearly outperformed previous RNNs on tasks that require learning the rules of regular languages (RLs) describable by deterministic *Finite State Automata* (FSA) (Casey, 1996; Siegelmann, 1992; Blair and Pollack, 1997; Kalinke and Lehmann, 1998; Zeng et al., 1994; Manolios and Fanelli, 1994; Omlin and Giles, 1996; Vahed and Omlin, 2004), both in terms of reliability and speed. LSTM also worked better on tasks involving context free languages (CFLs) that cannot be represented by HMMs or similar FSAs discussed in the RNN literature (Sun et al., 1993b; Wiles and Elman, 1995; Steijvers and Grunwald, 1996; Tonkes and Wiles, 1997; Rodriguez et al., 1999; Rodriguez and Wiles, 1998). CFL recognition (Lee, 1996) requires the functional equivalent of a stack. Some previous RNNs failed to learn small CFL training sets (Rodriguez and Wiles, 1998). Those that did not (Rodriguez et al., 1999; Bodén and Wiles, 2000) failed to extract the general rules, and did not generalize well on substantially larger test sets. Similar for context-sensitive languages (CFLs), e.g., (Chalup and Blair, 2003). LSTM generalized well though, requiring only the 30 shortest exemplars ($n \leq 10$) of the CSL $a^n b^n c^n$ to correctly predict the possible continuations of sequence prefixes for n up to 1000 and more. A combination of LSTM and the decoupled extended Kalman filter (Pérez-Ortiz et al., 2003) learned to deal correctly with values of n up to 10 million and more. That is, after training the network was able to read sequences of 30,000,000 symbols and more, one symbol at a time, and finally detect the subtle differences between *legal* strings such as $a^{10,000,000} b^{10,000,000} c^{10,000,000}$ and very similar but *illegal* strings such as $a^{10,000,000} b^{9,999,999} c^{10,000,000}$.

Bi-directional RNNs (Schuster and Paliwal, 1997; Schuster, 1999) are designed for input sequences whose starts and ends are known in advance, such as spoken sentences to be labeled by their phonemes; compare (Fukada et al., 1999; Baldi et al., 2001). To take both past and future context of each sequence element into account, one RNN processes the sequence from start to end, the other backwards from end to start. At each time step their combined outputs predict the corresponding label (if there is any). Bi-directional RNNs unfold their full potential when combined with the LSTM concept (Graves and Schmidhuber, 2005; Graves et al., 2009; Graves and Schmidhuber, 2009). Compare DAG-RNNs (Baldi and Pollastri, 2003).

Particularly successful in recent competitions are stacks (Sec. 5.9) of LSTM RNNs (Fernandez et al., 2007; Graves and Schmidhuber, 2009) trained by *Connectionist Temporal Classification* (CTC) (Graves et al., 2006), a gradient-based method for finding RNN weights that maximize the probability of teacher-given label sequences, given (typically much longer and more high-dimensional) streams of real-valued input vectors. CTC-LSTM performs simultaneous segmentation (alignment) and recognition (Sec. 5.20).

In the early 2000s, speech recognition was still dominated by HMMs combined with FNNs, e.g., (Bourlard and Morgan, 1994). Nevertheless, when trained from scratch on utterances from the TIDIGITS speech database, in 2003 LSTM already obtained results comparable to those of HMM-based systems (Graves et al., 2003; Beringer et al., 2005; Graves et al., 2006). In 2013, LSTM achieved best known results on the famous TIMIT phoneme recognition benchmark (Graves et al., 2013) (Sec. 5.20). Besides speech recognition and keyword spotting (Fernández et al., 2007), important applications of LSTM include protein analysis (Hochreiter and Obermayer, 2005), robot localization (Förster et al., 2007) and robot control (Mayer et al., 2008), handwriting recognition (Graves et al., 2008, 2009; Graves and Schmidhuber, 2009; Bluche et al., 2014), optical character recognition (Breuel et al., 2013), and others. RNNs can also be used for metalearning (Schmidhuber, 1987; Schaul and Schmidhuber, 2010) because they can in principle learn to run their own weight change algorithm (Schmidhuber, 1993a). A successful metalearner (Hochreiter et al., 2001b) used an LSTM RNN to quickly *learn a full-fledged learning algorithm* for quadratic functions (compare Sec. 6.8).

More recently, LSTM RNNs won several international pattern recognition competitions and set benchmark records on large and complex data sets, e.g., Sec. 5.15, 5.19, 5.20. LSTM is no panacea though—other RNNs sometimes outperformed LSTM at least on certain tasks (Jaeger, 2004; Martens and Sutskever, 2011; Pascanu et al., 2013b; Koutník et al., 2014) (compare Sec. 5.18).

5.12 1999: Max-Pooling (MP)

The feedforward HMAX model (Riesenhuber and Poggio, 1999) is similar to the Neocognitron (Sec. 5.3), but uses max-pooling layers instead of alternative local WTA methods, e.g., (Fukushima, 1980; Schmidhuber, 1989b; Maass, 2000; Fukushima, 2013a). Here a 2-dimensional layer or array of unit activations is partitioned into smaller rectangular arrays. Each is replaced in a down-sampling layer by the activation of its maximally active unit.

This is noteworthy because max-pooling (MP) CNNs or convnets (Sec. 5.3, 5.5) have become an essential ingredient of many modern, competition-winning, feedforward, visual Deep Learners (Sec. 5.17-5.19). Unlike HMAX, however, these are trained by BP (Sec. 5.4), which was applied to such MPCNNs only much later (Ranzato et al., 2007). Advantages of doing this were pointed out subsequently (Scherer et al., 2010).

5.13 2003: More Contest-Winning/Record-Setting, Often Not So Deep NNs

In the decade around 2000, many practical and commercial pattern recognition applications were dominated by non-neural machine learning methods such as *Support Vector Machines* (SVMs) (Vapnik, 1995; Schölkopf et al., 1998). Nevertheless, at least in certain domains, NNs outperformed other techniques.

A *Bayes NN* based on an ensemble (Breiman, 1996) of NNs won the *NIPS 2003 Feature Selection Challenge* with secret test set (Neal and Zhang, 2006; Neal, 2006). The NN was not very deep though—it had two hidden layers and thus rather shallow CAPs (Sec. 3) of depth 3.

Important for many present competition-winning pattern recognisers (Sec. 5.17, 5.19, 5.20) were developments in the CNN department. A BP-trained (LeCun et al., 1989) CNN (Sec. 5.3, Sec. 5.5) set a new MNIST record of 0.4% (Simard et al., 2003), using training pattern deformations (Baird, 1990) but no unsupervised pre-training (Sec. 5.9, 5.14). A standard BP net achieved 0.7% (Simard et al., 2003). Again, the corresponding CAP depth was low. Compare further improvements in Sec. 5.14, 5.16, 5.17.

Good image interpretation results (Behnke, 2003) were achieved with rather deep NNs trained by the BP variant *R-prop* (Riedmiller and Braun, 1993) (Sec. 5.6.2).

Deep LSTM RNNs started to obtain certain first speech recognition results comparable to those of HMM-based systems (Graves et al., 2003); compare Sec. 5.11, 5.19, 5.20.

5.14 2006: Deep Belief Networks / Improved CNNs / GPU-CNNs

While explicit DL research results have been published at least since 1991 (Sec. 5.8, 5.9), the expression *Deep Learning* (DL) was actually coined relatively late, around 2006, in the context of unsupervised pre-training of NN (Hinton and Salakhutdinov, 2006; Hinton et al., 2006) for accelerating subsequent supervised learning through BP.

The *Deep Belief Network* (DBN) is a stack of *Restricted Boltzmann Machines* (RBMs) (Smolensky, 1986), which in turn are *Boltzmann Machines* (Hinton and Sejnowski, 1986) with a single layer of feature-detecting units. Each RBM perceives pattern representations from the level below and learns to encode them in unsupervised fashion. At least in theory under certain assumptions, adding more layers improves a bound on the data's negative log probability (equivalent to its description length—compare Sec. 5.9).

Without any training pattern deformations (Sec. 5.13), such a system (re-trained by supervised BP) achieved 1.2% error rate (Hinton and Salakhutdinov, 2006) on the MNIST handwritten digits (Sec. 5.5, 5.13). This result helped to arouse interest in deep NNs. DBNs also achieved good results on phoneme recognition (Mohamed et al., 2009; Mohamed and Hinton, 2010; Hinton et al., 2012a; Deng and Yu, 2014). Compare Sec. 5.20.

Auto-Encoder (AE) stacks (Ballard, 1987) (Sec. 5.7) became a popular alternative way of pre-training deep FNNs in unsupervised fashion before fine-tuning them through BP (Sec. 5.4) (Bengio et al., 2007; Erhan et al., 2010). Denoising AEs (Vincent et al., 2008) discourage hidden unit perturbations in response to input perturbations, similar to how FMS (Sec. 5.6.3) for LOCOCODE AEs (Sec. 5.6.4) discourages output perturbations in response to weight perturbations. Sparse coding (Sec. 5.6.4) was formulated as a combination of convex optimization problems (Lee et al., 2007a). There is an extensive survey of stacked RBM and AE methods, focusing on post-2006 developments (Bengio, 2009).

Unsupervised DBNs and AE stacks are conceptually similar to, but in a sense less general than, the unsupervised RNN stack-based History Compressor of 1991 (Sec. 5.9), which can process not only stationary input patterns, but entire pattern sequences.

Also in 2006, a BP-trained (LeCun et al., 1989) CNN (Sec. 5.3, Sec. 5.5) set a new MNIST record of 0.39% (Ranzato et al., 2006), using training pattern deformations (Sec. 5.13) but no unsupervised pre-training. Compare further improvements in Sec. 5.16, 5.17. Similar CNNs were used for off-road obstacle avoidance (LeCun et al., 2006).

2006 also saw an early GPU-based CNN implementation (Chellapilla et al., 2006); compare earlier GPU-FNNs with a reported speed-up factor of 20 (Oh and Jung, 2004). GPUs or graphics cards have become more and more important for DL in subsequent years (Sec. 5.16-5.19).

5.15 2009: First Official Competitions Won by Deep Learning RNNs

Stacks (Sec. 5.9) of LSTM RNNs (Fernandez et al., 2007; Graves and Schmidhuber, 2009) trained by *Connectionist Temporal Classification* (CTC) (Graves et al., 2006) (Sec. 5.11) became the first RNNs to win official international pattern recognition contests (with secret test sets known only to the organisers). More precisely, three connected handwriting competitions at ICDAR 2009 in three different languages (French, Arab, Farsi) were won by deep LSTM RNNs without any *a priori* linguistic knowledge (Graves and Schmidhuber, 2009; Schmidhuber et al., 2011), performing simultaneous segmentation and recognition. Compare (Graves and Schmidhuber, 2005; Graves et al., 2006, 2009; Graves and Schmidhuber, 2009; Schmidhuber et al., 2011; Graves et al., 2013) (Sec. 5.20).

2009 also saw an impressive GPU-DBN implementation (Raina et al., 2009) orders of magnitudes faster than previous CPU-DBNs (see Sec. 5.14); see also (Coates et al., 2013).

5.16 2010: Plain Backprop (+ Distortions) on GPU Yields Excellent Results

In 2010, a new MNIST record of 0.35% error rate was set by good old BP (Sec. 5.4) in deep but otherwise standard NNs (Ciresan et al., 2010), using neither unsupervised pre-training (e.g., Sec. 5.7, 5.9, 5.14) nor convolution (e.g., Sec. 5.3, 5.5, 5.13). However, training pattern deformations (e.g., Sec. 5.13) were important to generate a big training set and avoid overfitting. This success was made possible mainly through a GPU implementation of BP that was up to 50 times faster than standard CPU versions. A good value of 0.95% was obtained without distortions except for small saccadic eye movement-like translations—compare Sec. 5.14.

Since BP was 3-5 decades old by then (Sec. 5.4), and pattern deformations 2 decades (Baird, 1990) (Sec. 5.13), these results seemed to suggest that advances in exploiting modern computing hardware were more important than advances in algorithms.

5.17 2011: Max-Pooling (MP) CNNs on GPU / Superhuman Vision Performance

In 2011, the first *GPU-implementation* (Ciresan et al., 2011a) of *Max-Pooling CNNs/Convnets* (GPU-MPCNNs) was described, building on earlier (MP)CNN work (Sec. 5.3, 5.5, 5.12), and on early GPU-based CNNs without MP (Chellapilla et al., 2006) (Sec. 5.14); compare GPU-DBNs (Raina et al., 2009) (Sec. 5.14) and early GPU-NNs (Oh and Jung, 2004). MPCNNs have alternating convolutional layers (Sec. 5.3) and max-pooling layers (MP, Sec. 5.12) topped by standard fully connected layers. All weights are trained by BP (Sec. 5.4, 5.5) (LeCun et al., 1989; Ranzato et al., 2007; Scherer et al., 2010). GPU-MPCNNs have become essential for many contest-winning FNNs (Sec. 5.19, Sec. 5.20).

Multi-Column (MC) GPU-MPCNNs (Ciresan et al., 2011b) are committees (Breiman, 1996; Schapire, 1990; Wolpert, 1992; Hashem and Schmeiser, 1992; Ueda, 2000; Dietterich, 2000a) of GPU-MPCNNs with simple democratic output averaging. Several MPCNNs see the same input; their output vectors are used to assign probabilities to the various possible classes. The class with the on average highest probability is chosen as the system's classification of the present input. Compare earlier, more sophisticated ensemble methods (Schapire, 1990), and the contest-winning ensemble Bayes-NN (Neal, 2006) of Sec. 5.13.

An MC-GPU-MPCNN was the first system to achieve superhuman visual pattern recognition (Ciresan et al., 2012b, 2011b) in a controlled competition, namely, the IJCNN 2011 traffic sign recognition contest

in San Jose (CA) (Stallkamp et al., 2011). This is of interest for fully autonomous, self-driving cars in traffic, e.g., (Dickmanns et al., 1994). The MC-GPU-MPCNN obtained 0.56% error rate and was twice better than human test subjects, three times better than the closest artificial NN competitor (Sermanet and LeCun, 2011), and six times better than the best non-neural method.

A few months earlier, the qualifying round was won in a 1st stage online competition, albeit by a much smaller margin: 1.02% (Ciresan et al., 2011b) vs 1.03% for second place (Sermanet and LeCun, 2011). After the deadline, the organisers revealed that human performance on the test set was 1.19%. That is, the best methods already seemed human-competitive. However, during the qualifying it was possible to incrementally gain information about the test set by probing it through repeated submissions. This is illustrated by better and better results obtained by various teams over time (Stallkamp et al., 2011) (the organisers eventually imposed a limit of 10 resubmissions). In the final competition this was not possible.

This illustrates a general problem with benchmarks whose test sets are public, or at least can be probed to some extent: competing teams tend to overfit on the test set even when it cannot be directly used for training, only for evaluation.

In 1997 many thought it a big deal that human chess world champion Kasparov was beaten by an IBM computer. But back then computers could not at all compete with little kids in visual pattern recognition, which seems much harder than chess from a computational perspective. Of course, the traffic sign domain is highly restricted, and kids are still much better general pattern recognisers. Nevertheless, by 2011, deep NNs could already learn to rival them in important limited visual domains.

An MC-GPU-MPCNN also was the first method to achieve human-competitive performance (around 0.2%) on MNIST (Ciresan et al., 2012c).

Given all the prior work on (MP)CNNs (Sec. 5.3, 5.5, 5.12) and GPU-CNNs (Sec. 5.14), GPU-MPCNNs are not a breakthrough in the scientific sense. But they are a commercially relevant breakthrough in efficient coding that has made a difference in several contests since 2011. Today, GPU-MPCNNs are used by most if not all *feedforward* competition-winning deep NNs (Sec. 5.19, Sec. 5.20).

5.18 2011: Hessian-Free Optimization for RNNs

Also in 2011 it was shown (Martens and Sutskever, 2011) that Hessian-free optimization, e.g., (Møller, 1993; Pearlmutter, 1994; Schraudolph, 2002) (Sec. 5.6.2), can alleviate the *Fundamental Deep Learning Problem* (Sec. 5.8) in RNNs, outperforming standard LSTM RNNs (Sec. 5.11) on several tasks. Compare other RNN (Jaeger, 2004; Pascanu et al., 2013b; Koutník et al., 2014) that also at least sometimes yield better results than LSTM.

5.19 2012: First Contests Won on Object Detection / Segmentation / ImageNet

In 2012, an ensemble-based (Breiman, 1996; Schapire, 1990; Wolpert, 1992; Dietterich, 2000a) Multi-Column (MC, Sec. 5.17, 5.13) variant of a GPU-MPCNN (Sec. 5.17) also achieved best results (Krizhevsky et al., 2012) on the *ImageNet* classification benchmark, which is popular in the computer vision community. Here relatively large image sizes of 256x256 pixels were necessary, as opposed to only 48x48 pixels for the traffic sign competition (Sec. 5.17).

Also in 2012, the biggest NN so far (10^9 free parameters) was trained in unsupervised mode (Sec. 5.7, 5.14) on unlabeled data (Le et al., 2012), then applied to ImageNet. The codes across its top layer were used to train a simple supervised classifier, which achieved best results so far on 20,000 classes. Instead of relying on efficient GPU programming, this was done by brute force on 1,000 standard machines with 16,000 cores.

So by 2011/2012, excellent results had been achieved by Deep Learners in image *recognition and classification* (Sec. 5.17, 5.19). The computer vision community, however, is especially interested in *object detection* in large images, for applications such as image-based search engines, or for biomedical diagnosis where the goal may be to automatically detect tumors etc in images of human tissue. Object detection presents additional challenges. One natural approach is to train a deep NN classifier on patches of big images, then use it as a feature detector to be shifted across unknown visual scenes, using various rotations and zoom factors. Image parts that yield highly active output units are likely to contain objects similar to those the NN was trained on.

2012 finally saw the first DL system (an MC-GPU-MPCNN, Sec. 5.17) to win a contest on visual *object detection* in large images of several million pixels (ICPR 2012 Contest on Mitosis Detection in Breast Cancer Histological Images., 2012; Roux et al., 2013; Ciresan et al., 2013). Such biomedical applications may turn out to be among the most important applications of DL. The world spends over 10% of GDP on healthcare ($>$ 6 trillion USD per year), much of it on medical diagnosis through expensive experts. Partial automation of this could not only save lots of money, but also make expert diagnostics accessible to many who currently cannot afford it. It is gratifying to observe that today deep NNs may actually help to improve healthcare and perhaps save human lives.

2012 also saw the first pure *image segmentation* contest won by DL, again through an MC-GPU-MPCNN (Segmentation of Neuronal Structures in EM Stacks Challenge, 2012; Ciresan et al., 2012a). (It should be mentioned, however, that LSTM RNNs already performed simultaneous segmentation and recognition when they became the first recurrent Deep Learners to win official international pattern recognition contests—Sec. 5.15.) EM stacks are relevant for the recently approved huge brain projects in Europe and the US. Given electron microscopy images of stacks of thin slices of animal brains, the goal is to build a detailed 3D model of the brain's neurons and dendrites. But human experts need many hours and days and weeks to annotate the images: Which parts depict neuronal membranes? Which parts are irrelevant background? This needs to be automated. Deep MC-GPU-MPCNNs learned to solve this task through experience with many training images, and won the contest on all three evaluation metrics by a large margin, with superhuman performance in terms of pixel error.

Both object detection (Ciresan et al., 2013) and image segmentation (Ciresan et al., 2012a) profit from fast MPCNN-based image scans that avoid redundant computations. Recent methods speed up naive implementations by three orders of magnitude (Masci et al., 2013b,a).

Also in 2012, on the IAM-OnDoDB benchmark, LSTM RNNs (Sec. 5.11) outperformed all other methods (HMMs, SVMs) on online mode detection (Otte et al., 2012; Indermuhle et al., 2012) and keyword spotting (Indermuhle et al., 2011). On the long time lag problem of language modelling, LSTM outperformed all statistical approaches on the IAM-DB benchmark (Frinken et al., 2012).

5.20 2013: More Contests and Benchmark Records

A stack (Fernandez et al., 2007; Graves and Schmidhuber, 2009) (Sec. 5.9) of bi-directional LSTM recurrent NNs (Graves and Schmidhuber, 2005) trained by CTC (Sec. 5.11, 5.15) broke a famous TIMIT speech (phoneme) recognition record, achieving 17.7% test set error rate (Graves et al., 2013), despite thousands of man years previously spent on *Hidden Markov Model* (HMMs)-based speech recognition research. Compare earlier DBN results (Sec. 5.14). CTC-LSTM also helped to score first at NIST's OpenHaRT2013 evaluation (Bluche et al., 2014). For *Optical Character Recognition* (OCR), LSTM RNNs outperformed commercial recognizers of historical data (Breuel et al., 2013).

A new record on the *ICDAR Chinese handwriting recognition benchmark* (over 3700 classes) was set on a desktop machine by an MC-GPU-MPCNN (Sec. 5.17) with almost human performance (Ciresan and Schmidhuber, 2013).

The *MICCAI 2013 Grand Challenge on Mitosis Detection* (Veta et al., 2013) also was won by an object-detecting MC-GPU-MPCNN (Ciresan et al., 2013). Its data set was even larger and more challenging than the one of ICPR 2012 (Sec. 5.19): a real-world dataset including many ambiguous cases and frequently encountered problems such as imperfect slide staining.

Deep GPU-MPCNNs (Sec. 5.17) also helped to achieve new best results on ImageNet classification (Zeiler and Fergus, 2013) and PASCAL object detection (Girshick et al., 2013), previously mentioned important benchmarks of the computer vision community. Compare additional work on object detection (Szegedy et al., 2013) and scene parsing (Farabet et al., 2013).

Additional contests are mentioned in the web pages of the Swiss AI Lab IDSIA, the University of Toronto, NY University, and the University of Montreal. (Unlike in most academic contests, winners of contests listed at the commercial web site *kaggle.com* have to hand their code over to companies.)

5.20.1 Currently Successful Supervised Techniques: LSTM RNNs / GPU-MPCNNs

Most competition-winning or benchmark record-setting Deep Learners actually use one of two *supervised* techniques: (a) recurrent LSTM (1997) trained by CTC (2006) (Sec. 5.11, 5.15, 5.19, 5.20), or (b) feedforward GPU-MPCNNs (2011, Sec. 5.17, 5.19, 5.20) building on earlier work since the 1960s (Sec. 5.4, 5.3, 5.5, 5.12). Exceptions include two 2011 contests specialised on *transfer learning* (Goodfellow et al., 2011; Mesnil et al., 2011; Goodfellow et al., 2012)—but compare SL-based transfer in deep NN (Ciresan et al., 2012d).

Remarkably, in the 1990s a trend went from partially unsupervised RNN stacks (Sec. 5.9) to purely supervised LSTM RNNs (Sec. 5.11), just like in the 2000s a trend went from partially unsupervised FNN stacks (Sec. 5.14) to purely supervised MPCNNs (Sec. 5.17–5.20).

Nevertheless, in many applications it can still be advantageous to combine the best of both worlds - *supervised* learning and *unsupervised* pre-training (Sec. 5.9, 5.14).

5.21 Recent Tricks for Improving SL Deep NNs (Compare Sec. 5.6.2, 5.6.3)

Rectified Linear Units (ReLU) clamp negative activations to zero ($f(x) = x$ if $x > 0$, $f(x) = 0$ otherwise). ReLU NNs are useful for RBMs (Nair and Hinton, 2010; Maas et al., 2013), outperformed sigmoidal activation functions in deep NNs (Glorot et al., 2011), and helped to obtain best results on several benchmark problems across multiple domains, e.g., (Krizhevsky et al., 2012; Dahl et al., 2013).

Maxout NNs (Goodfellow et al., 2013) combine competitive interactions in combination with *dropout* (Hinton et al., 2012b) to achieve excellent results on certain benchmarks. Dropout removes units from NNs during training to improve generalisation. Some view it as an ensemble method that trains multiple data models simultaneously (Baldi and Sadowski, 2013). Under certain circumstances, it could also be viewed as a form of training set augmentation: effectively, more and more informative complex features are removed from the training data. Compare dropout for RNNs (Pham et al., 2013; Pachitariu and Sahani, 2013; Pascanu et al., 2013a). A deterministic approximation coined "fast dropout" (Wang and Manning, 2013) can lead to faster learning and evaluation and was adapted for RNNs (Bayer et al., 2013). Dropout is closely related to older, biologically plausible techniques of adding noise to neurons or synapses during training, e.g., (Murray and Edwards, 1993; Schuster, 1992; Nadal and Parga, 1994; Jim et al., 1995; An, 1996), which in turn are closely related to finding perturbation-resistant low-complexity NNs, e.g., through FMS (Sec. 5.6.3). MDL-based stochastic variational methods (Graves, 2011) are also related to FMS. They are useful for RNNs, where classic regularizers such as weight decay (Sec. 5.6.3) represent a bias towards limited memory capacity, e.g., (Pascanu et al., 2013b).

NNs with competing linear units tend to outperform those with non-competing nonlinear units, and avoid catastrophic forgetting through BP when training sets change over time (Srivastava et al., 2013). In this context, choosing a learning algorithm may be more important than choosing activation functions (Goodfellow et al., 2014). Compare early RNNs with competing units for SL and RL (Schmidhuber, 1989b). To address overfitting, instead of depending on pre-wired regularizers and hyper-parameters (Hertz et al., 1991; Bishop, 2006), self-delimiting RNNs (SLIM NNs) with competing units (Schmidhuber, 2012) can in principle learn to select their own runtime and their own numbers of effective free parameters, thus learning their own computable regularisers (Sec. 4.3, 5.6.3), becoming fast and *slim* when necessary. One may penalize the task-specific total length of connections (Legenstein and Maass, 2002) and communication costs of SLIM NNs implemented on the 3-dimensional brain-like multi-processor hardware to expected in the future.

RmsProp (Tieleman and Hinton, 2012; Schaul et al., 2013) can improve first order gradient descent methods (Sec. 5.4, 5.6.2); compare vario-η (Neuneier and Zimmermann, 1996), *Adagrad* (Duchi et al., 2011) and *Adadelta* (Zeiler, 2012). Many additional, older tricks (Sec. 5.6.2, 5.6.3) should also be applicable to today's deep NNs; compare (Orr and Müller, 1998; Montavon et al., 2012).

5.22 Consequences for Neuroscience

It is ironic that artificial NNs (ANNs) can help to better understand biological NNs (BNNs)—see the ISBI 2012 results mentioned in Sec. 5.19 (Segmentation of Neuronal Structures in EM Stacks Challenge, 2012;

Ciresan et al., 2012a). The feature detectors invented by deep visual ANNs should also be highly predictive of what neuroscientists will find in deep layers of BNNs. While the visual cortex of BNNs may use a quite different learning algorithm, its objective function to be minimised must be similar to the one of visual ANNs. Compare (Lee et al., 2007b; Yamins et al., 2013).

6 DL in FNNs and RNNs for Reinforcement Learning (RL)

So far we have focused on Deep Learning (DL) in supervised or unsupervised NNs. Such NNs learn to perceive / encode / predict / classify patterns or pattern sequences, but they do not learn to act in the more general sense of *Reinforcement Learning* (RL) in unknown environments, e.g., (Kaelbling et al., 1996; Sutton and Barto, 1998). Here we add a discussion of DL FNNs and RNNs for RL. It will be shorter than the discussion of FNNs and RNNs for SL and UL (Sec. 5), reflecting the current size of the various fields.

Without a teacher, solely from occasional real-valued pain and pleasure signals, RL agents must discover how to interact with a dynamic, initially unknown environment to maximize their expected cumulative reward signals (Sec. 2). There may be arbitrary, a priori unknown delays between actions and perceivable consequences. The problem is as hard as any problem of computer science, since any task with a computable description can be formulated in the RL framework, e.g., (Hutter, 2005). For example, an answer to the famous question of whether $P = NP$ (Levin, 1973b; Cook, 1971) would also set limits for what is achievable by general RL.

The following sections mostly focus on certain obvious intersections of DL and RL—they cannot serve as a general RL survey.

6.1 RL Through NN World Models Yields RNNs With Deep CAPs

In the special case of an RL NN controller C interacting with a *deterministic* environment, a separate FNN called M can learn to become C's world model, predicting C's inputs from previous actions and inputs, e.g., (Werbos, 1981, 1987; Munro, 1987; Jordan, 1988; Werbos, 1989b,a; Robinson and Fallside, 1989; Jordan and Rumelhart, 1990; Schmidhuber, 1990d; Narendra and Parthasarathy, 1990; Werbos, 1992; Cochocki and Unbehauen, 1993; Levin and Narendra, 1995; Ljung, 1998). Assume M has learned to produce accurate predictions. We can use M to substitute the environment. Then M and C form an RNN. Now BP for RNNs (Sec. 5.4.1) can be used to achieve desired input events such as high real-valued reward signals: While M's weights remain fixed, gradient information for C's weights is propagated back through M down into C and back through M etc. To a certain extent, the approach is also applicable in probabilistic or uncertain environments, as long as the inner products of M's C-based gradient estimates and M's "true" gradients tend to be positive.

In general, this approach implies deep CAPs for C, unlike in DP-based traditional RL (Sec. 6.2). The approach was used to learn to back up a model truck (Nguyen and Widrow, 1989). An RL active vision system used it to learn sequential shifts of a fovea to detect targets in visual scenes (Schmidhuber and Huber, 1991).

To allow for an M with memories of previous events in *partially observable worlds* (Sec. 6.3), the most general variant of this approach uses an RNN instead of an FNN as world model (Schmidhuber, 1990d, 1991c). This may cause deep CAPs not only for C but also for M. M can also be used to optimise expected reward by *planning* future action sequences (Schmidhuber, 1990d). In fact, the winners of the 2004 RoboCup World Championship in the fast league (Egorova et al., 2004) trained NNs to predict the effects of steering signals on fast robots with 4 motors for 4 different wheels. During play, such NN models were used to achieve desirable subgoals, by optimising action sequences through quickly planning ahead. The approach also was used to create *self-healing* robots able to compensate for faulty motors whose effects do not longer match the predictions of the NN models (Gloye et al., 2005; Schmidhuber, 2007).

Typically M is not given in advance. Then an essential question is: which experiments should C conduct to quickly improve M? The *Formal Theory of Fun and Creativity* (Schmidhuber, 2006a, 2013b) formalizes driving forces and value functions behind such curious and exploratory behavior: A measure of the *learning progress* of M becomes the intrinsic reward of C (Schmidhuber, 1991a); compare (Singh

et al., 2005; Oudeyer et al., 2013). This motivates C to create action sequences (experiments) such that M makes quick progress.

6.2 Deep FNNs for Traditional RL and Markov Decision Processes (MDPs)

The classical approach to RL (Samuel, 1959; Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998) makes the simplifying assumption of *Markov Decision Processes* (MDPs): the current input of the RL agent conveys all information necessary to compute an optimal next output event or decision. This allows for greatly reducing CAP depth in RL NNs (Sec. 3, 6.1) by using the *Dynamic Programming* (DP) trick (Bellman, 1957). The latter is often explained in a probabilistic framework, e.g., (Sutton and Barto, 1998), but its basic idea can already be conveyed in a deterministic setting. For simplicity, using the notation of Sec. 2, let input events x_t encode the entire current state of the environment including a real-valued reward r_t (no need to introduce additional vector-valued notation, since real values can encode arbitrary vectors of real values). The original RL goal (find weights that maximize the sum of all rewards of an episode) is replaced by an equivalent set of alternative goals set by a real-valued value function V defined on input events. Consider any two subsequent input events x_t, x_k. Recursively define $V(x_t) = r_t + V(x_k)$, where $V(x_k) = r_k$ if x_k is the *last* input event. The goal is to find weights that maximize the V of all input events, by causing appropriate output events or actions.

Due to the Markov assumption, an FNN suffices to implement the policy that maps input to output events. Relevant CAPs are not deeper than this FNN. V itself is often modeled by a *separate FNN* (also yielding typically short CAPs) learning to approximate $V(x_t)$ only from *local* information $r_t, V(x_k)$.

Many variants of traditional RL exist, e.g., (Barto et al., 1983; Watkins, 1989; Watkins and Dayan, 1992; Moore and Atkeson, 1993; Schwartz, 1993; Baird, 1994; Rummery and Niranjan, 1994; Singh, 1994; Baird, 1995; Kaelbling et al., 1995; Peng and Williams, 1996; Mahadevan, 1996; Tsitsiklis and van Roy, 1996; Bradtke et al., 1996; Santamaría et al., 1997; Sutton and Barto, 1998; Wiering and Schmidhuber, 1998b; Baird and Moore, 1999; Meuleau et al., 1999; Morimoto and Doya, 2000; Bertsekas, 2001; Brafman and Tennenholtz, 2002; Abounadi et al., 2002; Lagoudakis and Parr, 2003; Sutton et al., 2008; Maei and Sutton, 2010). Most are formulated in a probabilistic framework, and evaluate pairs of input and output (action) events (instead of input events only). To facilitate certain mathematical derivations, some discount delayed rewards (but such distortions of the original RL problem are problematic).

Perhaps the most well-known RL NN is the world-class RL backgammon player Tesauro (1994), which achieved the level of human world champions by playing against itself. Its nonlinear, rather shallow FNN maps a large but finite number of discrete board states to values. More recently, a rather deep GPU-MPCNN (Sec. 5.17) was used in a traditional RL framework to learn to play several Atari 2600 computer games directly from 84x84 pixel video input (Mnih et al., 2013), using experience replay (Lin, 1993), extending previous work on *Neural Fitted Q-Learning* (NFQ) (Riedmiller, 2005). Compare (Grüttner et al., 2010) and an earlier, raw video-based RL NN for computer games (Koutník et al., 2013) trained by *Indirect Policy Search* (Sec. 6.7).

6.3 Deep RL RNNs for Partially Observable MDPs (POMDPs)

The *Markov assumption* (Sec. 6.2) is often unrealistic. We cannot directly perceive what is behind our back, let alone the current state of the entire universe. However, memories of previous events can help to deal with *partially observable Markov decision problems (POMDPs)*, e.g., (Schmidhuber, 1991c; Ring, 1991, 1993, 1994; Williams, 1992; Lin, 1993; Teller, 1994; Schmidhuber, 1995; Kaelbling et al., 1995; Littman et al., 1995; Boutilier and Poole, 1996; Littman, 1996; Jaakkola et al., 1995; McCallum, 1996; Kimura et al., 1997; Wiering and Schmidhuber, 1996, 1998a). A naive way of implementing memories without leaving the MDP framework (Sec. 6.2) would be to simply consider a possibly huge state space, namely, the set of all possible observation histories and their prefixes. A more realistic way is to use function approximators such as RNNs that produce compact state features as a function of the entire history seen so far. Generally speaking, POMDP RL often uses DL RNNs to learn which events to memorize and which to ignore. Three basic alternatives are:

1. Use an RNN as a value function mapping arbitrary event histories to values, e.g., (Schmidhuber,

1990b, 1991c; Lin, 1993; Bakker, 2002). For example, deep LSTM RNNs were used in this way for RL robots (Bakker et al., 2003).

2. Use an RNN controller in conjunction with a second RNN as predictive world model, to obtain a combined RNN with deep CAPs—see Sec. 6.1.

3. Use an RNN for RL by *Direct Search* (Sec. 6.6) or *Indirect Search* (Sec. 6.7) in weight space.

In general, however, POMDPs may imply greatly increased CAP depth.

6.4 RL Facilitated by UL

RL machines may profit from UL for input preprocessing, e.g., (Jodogne and Piater, 2007). In particular, an UL NN can learn to compactly encode environmental inputs such as images or videos, e.g., Sec. 5.7, 5.9, 5.14. The compact codes (instead of the high-dimensional raw data) can be fed into an RL machine, whose job thus may become much easier (Legenstein et al., 2010; Cuccu et al., 2011), just like SL may profit from UL, e.g., Sec. 5.7, 5.9, 5.14. For example, NFQ (Riedmiller, 2005) was applied to real-world control tasks (Lange and Riedmiller, 2010; Riedmiller et al., 2012) where purely visual inputs were compactly encoded by deep autoencoders (Sec. 5.7, 5.14). RL combined with UL based on *Slow Feature Analysis* (Wiskott and Sejnowski, 2002; Kompella et al., 2012) enabled a real humanoid robot to learn skills from raw high-dimensional video streams (Luciw et al., 2013). To deal with POMDPs (Sec. 6.3), a RAAM (Pollack, 1988) (Sec. 5.7) was employed as a deep unsupervised sequence encoder for RL (Gisslen et al., 2011).

6.5 Deep Hierarchical RL (HRL) and Subgoal Learning

Multiple learnable levels of abstraction (Fu, 1977; Lenat and Brown, 1984; Ring, 1994; Bengio et al., 2013; Deng and Yu, 2014) seem as important for RL as for SL. Work on NN-based hierarchical RL (HRL) has been published since the early 1990s. In particular, gradient-based subgoal discovery with FNNs or RNNs decomposes RL tasks into subtasks for RL submodules (Schmidhuber, 1991b; Schmidhuber and Wahnsiedler, 1992). Numerous alternative HRL techniques have been proposed, e.g., (Ring, 1991, 1994; Tenenberg et al., 1993; Moore and Atkeson, 1995; Precup et al., 1998; Dietterich, 2000b; Menache et al., 2002; Doya et al., 2002; Ghavamzadeh and Mahadevan, 2003; Barto and Mahadevan, 2003; Samejima et al., 2003; Bakker and Schmidhuber, 2004; Whiteson et al., 2005; Simsek and Barto, 2008). While HRL frameworks such as Feudal RL (Dayan and Hinton, 1993) and options (Sutton et al., 1999b; Barto et al., 2004; Singh et al., 2005) do not directly address the problem of automatic subgoal discovery, HQ-learning (Wiering and Schmidhuber, 1998a) automatically decomposes POMDPs (Sec. 6.3) into sequences of simpler subtasks that can be solved by memoryless policies learnable by reactive sub-agents. Recent HRL organizes potentially deep NN RL sub-modules into self-organizing, 2-dimensional motor control maps (Ring et al., 2011) inspired by neurophysiological findings (Graziano, 2009).

6.6 Deep RL by Direct NN Search / Policy Gradients / Evolution

Not quite as universal as the methods of Sec. 6.8, yet both practical and more general than most traditional RL algorithms (Sec. 6.2), are methods for *Direct Policy Search* (DS). Without a need for value functions or Markovian assumptions (Sec. 6.2, 6.3), the weights of an FNN or RNN are directly evaluated on the given RL problem. The results of successive trials inform further search for better weights. Unlike with RL supported by BP (Sec. 5.4, 6.3, 6.1), CAP depth (Sec. 3, 5.8) is not a crucial issue. DS may solve the credit assignment problem without backtracking through deep causal chains of modifiable parameters—it neither cares for their existence, nor tries to exploit them.

An important class of DS methods for NNs are policy gradient methods (Williams, 1986, 1988, 1992; Baxter and Bartlett, 1999; Sutton et al., 1999a; Aberdeen, 2003; Ghavamzadeh and Mahadevan, 2003; Kohl and Stone, 2004; Wierstra et al., 2007, 2008; Rückstieß et al., 2008; Peters and Schaal, 2008b,a; Sehnke et al., 2010; Grüttner et al., 2010; Wierstra et al., 2010; Peters, 2010; Bartlett and Baxter, 2011; Grondman

et al., 2012). Gradients of the total reward with respect to policies (NN weights) are estimated (and then exploited) through repeated NN evaluations.

RL NNs can also be evolved through *evolutionary algorithms* (EAs) (Rechenberg, 1971; Schwefel, 1974; Holland, 1975; Fogel et al., 1966; Smith, 1980; Cramer, 1985; Goldberg, 1989) in a series of trials. Here several policies are represented by a population of NNs improved by repeatedly recombining the population's fittest individuals. Compare probability distribution-based EAs (Baluja, 1994; Saravanan and Fogel, 1995; Sałustowicz and Schmidhuber, 1997; Larraanaga and Lozano, 2001) and *Covariance Matrix Estimation Evolution Strategies* (CMA-ES) (Hansen and Ostermeier, 2001; Hansen et al., 2003; Igel, 2003). Many RNN evolvers have been proposed (Miller et al., 1989; Yao, 1993; Nolfi et al., 1994; Sims, 1994; Yamauchi and Beer, 1994; Miglino et al., 1995; Moriarty, 1997; Pasemann et al., 1999). One particularly effective family of methods *coevolves* neurons, combining them into networks, and selecting those neurons for reproduction that participated in the best-performing networks (Moriarty and Miikkulainen, 1996; Gomez, 2003; Gomez and Miikkulainen, 2003). This can help to solve deep POMDPs (Gomez and Schmidhuber, 2005). *Co-Synaptic Neuro-Evolution* (CoSyNE) does something similar on the level of synapses or weights (Gomez et al., 2008); benefits of this were shown on difficult non-linear POMDP benchmarks.

Natural Evolution Strategies (NES) (Wierstra et al., 2008; Glasmachers et al., 2010; Sun et al., 2009, 2013) link policy gradient methods and evolutionary approaches through the concept of natural gradients (Amari, 1998). RNN evolution may also help to improve SL for deep RNNs through *Evolino* (Schmidhuber et al., 2007) (Sec. 5.8).

6.7 Deep RL by Compressed NN Search

Some DS methods (Sec. 6.6) can evolve NNs with hundreds of weights, but not millions. How to search for large and deep NNs? Most SL and RL methods mentioned so far somehow search the space of weights w_i. Some profit from a reduction of the search space through shared w_i that get reused over and over again, e.g., in CNNs (Sec. 5.3, 5.5, 5.19), or in RNNs for SL (Sec. 5.4, 5.11) and RL (Sec. 6.6).

It may be possible, however, to exploit *additional* regularities/compressibilities in the space of solutions, through *indirect search in weight space*. Instead of evolving large NNs directly (Sec. 6.6), one can sometimes greatly reduce the search space by evolving *compact encodings* of NNs, e.g., through *Lindenmeyer Systems* (Lindenmayer, 1968; Jacob et al., 1994), *graph rewriting* (Kitano, 1990), *Cellular Encoding* (Gruau et al., 1996), *NeuroEvolution of Augmenting Topologies* (NEAT) (Stanley and Miikkulainen, 2002), and HyperNEAT (D'Ambrosio and Stanley, 2007; Stanley et al., 2009; Clune et al., 2011).

A general approach (Schmidhuber, 1995) for both SL and RL seeks to compactly encode weights of large NNs (Schmidhuber, 1997) through programs written in a *universal programming language* (Gödel, 1931; Church, 1936; Turing, 1936; Post, 1936). Often it is much more efficient to systematically search the space of such programs with a bias towards short and fast programs (Levin, 1973b; Schmidhuber, 1995, 1997, 2004), instead of directly searching the huge space of possible NN weight matrices. A previous universal language for encoding NNs was assembler-like (Schmidhuber, 1995). More recent work uses more practical languages based on coefficients of popular transforms (Fourier, wavelet, etc). In particular, RNN weight matrices may be compressed like images, by encoding them through the coefficients of a *discrete cosine transform* (DCT) (Koutník et al., 2010, 2013). Compact DCT-based descriptions can be evolved through NES or CoSyNE (Sec. 6.6). An RNN with over a million weights learned (without a teacher) to drive a simulated car in the TORCS driving game (Loiacono et al., 2009, 2011), based on a high-dimensional video-like visual input stream (Koutník et al., 2013). The RNN learned both control and visual processing from scratch, without being aided by UL. (Of course, UL might help to generate more compact image codes to be fed into a smaller RNN, to reduce the overall computational effort.)

6.8 Universal RL

General purpose learning algorithms may improve themselves in open-ended fashion and environment-specific ways in a lifelong learning context (Schmidhuber, 1987; Schmidhuber et al., 1997b,a; Schaul and Schmidhuber, 2010). The most general type of RL is constrained only by the fundamental limitations of computability identified by the founders of theoretical computer science (Gödel, 1931; Church, 1936;

Turing, 1936; Post, 1936). Remarkably, there exist blueprints of *universal problem solvers* or *universal RL machines* for unlimited problem depth that are time-optimal in various theoretical senses (Hutter, 2005, 2002; Schmidhuber, 2002, 2006b). The *Gödel Machine* can be implemented on general computers such as RNNs and may improve any part of its software (including the learning algorithm itself) in a way that is provably time-optimal in a certain sense (Schmidhuber, 2006b). It can be initialized by an *asymptotically optimal* meta-method (Hutter, 2002) (also applicable to RNNs) which will solve any well-defined problem as quickly as the unknown fastest way of solving it, save for an additive constant overhead that becomes negligible as problem size grows. Note that most problems are large; only few are small. AI and DL researchers are still in business because many are interested in problems so small that it is worth trying to reduce the overhead through less general methods (including heuristics). Here I won't further discuss universal RL methods, which go beyond what is usually called DL.

7 Conclusion

Deep Learning (DL) in *Neural Networks* (NNs) is relevant for *Supervised Learning* (SL) (Sec. 5), *Unsupervised Learning* (UL) (Sec. 5), and *Reinforcement Learning* (RL) (Sec. 6). By alleviating problems with deep *Credit Assignment Paths* (CAPs, Sec. 3, 5.8), UL (Sec. 5.6.4) cannot only facilitate SL of sequences (Sec. 5.9) and stationary patterns (Sec. 5.7, 5.14), but also RL (Sec. 6.4, 4.2). *Dynamic Programming* (DP, Sec. 4.1) is important for both deep SL (Sec. 5.4) and traditional RL with deep NNs (Sec. 6.2). A search for solution-computing, perturbation-resistant (Sec. 5.6.3, 5.14, 5.21), low-complexity NNs describable by few bits of information (Sec. 4.3) can improve deep SL & UL (Sec. 5.6.3, 5.6.4) as well as RL (Sec. 6.7), especially in the case of partially observable environments (Sec. 6.3). Deep SL, UL, RL often create hierarchies of more and more abstract, less and less redundant representations of stationary data (Sec. 5.7, 5.14), sequential data (Sec. 5.9), or RL policies (Sec. 6.5). The future may belong to general purpose learning algorithms that improve themselves in provably optimal ways (Sec. 6.8), but these are not yet practical or commercially relevant. While UL can facilitate SL, pure SL for feedforward NNs (FNNs) (Sec. 5.4, 5.5, 5.16) and recurrent NNs (RNNs) (Sec. 5.4, 5.11) did not only win early contests (Sec. 5.10, 5.13), but also most of the recent ones (Sec. 5.15-5.20). DL FNNs profited from GPU implementations (Sec. 5.14, 5.15, 5.16, 5.17). In particular, GPU-based (Sec. 5.17) Max-Pooling Convolutional NNs (Sec. 5.3, 5.5, 5.12) won competitions not only in pattern recognition (Sec. 5.17 - 5.20) but also image segmentation (Sec. 5.19) and object detection (Sec. 5.19 - 5.20).

Acknowledgments

Thanks to numerous NN experts for valuable comments.

References

Aberdeen, D. (2003). *Policy-Gradient Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Australian National University.

Abounadi, J., Bertsekas, D., and Borkar, V. S. (2002). Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3):681–698.

Akaike, H. (1970). Statistical predictor identification. *Ann. Inst. Statist. Math.*, 22:203–217.

Allender, A. (1992). Application of time-bounded Kolmogorov complexity in complexity theory. In Watanabe, O., editor, *Kolmogorov complexity and computational complexity*, pages 6–22. EATCS Monographs on Theoretical Computer Science, Springer.

Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *IEEE 1st International Conference on Neural Networks, San Diego*, volume 2, pages 609–618.

Almeida, L. B., Almeida, L. B., Langlois, T., Amaral, J. D., and Redol, R. A. (1997). On-line step size adaptation. Technical report, INESC, 9 Rua Alves Redol, 1000.

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Trans. EC*, 16(3):299–307.

Amari, S., Cichocki, A., and Yang, H. (1996). A new learning algorithm for blind signal separation. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8. The MIT Press.

Amari, S. and Murata, N. (1993). Statistical theory of learning curves under entropic loss criterion. *Neural Computation*, 5(1):140–153.

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276.

An, G. (1996). The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, 8(3):643–674.

Andrade, M., Chacon, P., Merelo, J., and Moran, F. (1993). Evaluation of secondary structure of proteins from uv circular dichroism spectra using an unsupervised learning neural network. *Protein Engineering*, 6(4):383–390.

Ash, T. (1989). Dynamic node creation in backpropagation neural networks. *Connection Science*, 1(4):365–375.

Atick, J. J., Li, Z., and Redlich, A. N. (1992). Understanding retinal color coding from first principles. *Neural Computation*, 4:559–572.

Baird, H. (1990). Document Image Defect Models. In *Proceddings, IAPR Workshop on Syntactic and Structural Pattern Recognition*, Murray Hill, NJ.

Baird, L. and Moore, A. W. (1999). Gradient descent for general reinforcement learning. In *Advances in neural information processing systems 12 (NIPS)*, pages 968–974. MIT Press.

Baird, L. C. (1994). Reinforcement learning in continuous time: Advantage updating. In *IEEE World Congress on Computational Intelligence*, volume 4, pages 2448–2453. IEEE.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning*, pages 30–37.

Bakker, B. (2002). Reinforcement learning with Long Short-Term Memory. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 1475–1482. MIT Press, Cambridge, MA.

Bakker, B. and Schmidhuber, J. (2004). Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In et al., F. G., editor, *Proc. 8th Conference on Intelligent Autonomous Systems IAS-8*, pages 438–445, Amsterdam, NL. IOS Press.

Bakker, B., Zhumatiy, V., Gruener, G., and Schmidhuber, J. (2003). A robot that reinforcement-learns to identify and memorize important previous observations. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2003*, pages 430–435.

Baldi, P., Brunak, S., Frasconi, P., Pollastri, G., and Soda, G. (2001). Bidirectional dynamics for protein secondary structure prediction. *Lecture Notes in Computer Science*, 1828:80–104.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58.

Baldi, P. and Pollastri, G. (2003). The principled design of large-scale recursive neural network architectures–DAG-RNNs and the protein structure prediction problem. *J. Mach. Learn. Res.*, 4:575–602.

Baldi, P. and Sadowski, P. (2013). Understanding dropout. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 26, pages 2814–2822.

Ballard, D. H. (1987). Modular learning in neural networks. In *Proc. AAAI*, pages 279–284.

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University.

Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, 1(3):295–311.

Barlow, H. B., Kaushal, T. P., and Mitchison, G. J. (1989). Finding minimum entropy codes. *Neural Computation*, 1(3):412–423.

Barrow, H. G. (1987). Learning receptive fields. In *Proceedings of the IEEE 1st Annual Conference on Neural Networks*, volume IV, pages 115–121. IEEE.

Bartlett, P. L. and Baxter, J. (2011). Infinite-horizon policy-gradient estimation. *arXiv preprint arXiv:1106.0665*.

Barto, A. G. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379.

Barto, A. G., Singh, S., and Chentanez, N. (2004). Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of International Conference on Developmental Learning (ICDL)*, pages 112–119. MIT Press, Cambridge, MA.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:834–846.

Battiti, R. (1989). Accelerated backpropagation learning: two optimization methods. *Complex Systems*, 3(4):331–342.

Battiti, T. (1992). First- and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4(2):141–166.

Baxter, J. and Bartlett, P. (1999). Direct Gradient-Based Reinforcement Learning. Technical report, Research School of Information Sciences and Engineering, Australian National University.

Bayer, J., Osendorfer, C., Chen, N., Urban, S., and van der Smagt, P. (2013). On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701*.

Bayer, J., Wierstra, D., Togelius, J., and Schmidhuber, J. (2009). Evolving memory cell structures for sequence learning. In *Proc. ICANN (2)*, pages 755–764.

Becker, S. (1990). Unsupervised learning procedures for neural networks. Technical report, Department of Computer Science, University of Toronto, Ontario.

Becker, S. (1991). Unsupervised learning procedures for neural networks. *International Journal of Neural Systems*, 2(1 & 2):17–33.

Becker, S. and Le Cun, Y. (1989). Improving the convergence of back-propagation learning with second order methods. In Touretzky, D., Hinton, G., and Sejnowski, T., editors, *Proc. 1988 Connectionist Models Summer School*, pages 29–37, Pittsburg 1988. Morgan Kaufmann, San Mateo.

Behnke, S. (2003). *Hierarchical Neural Networks for Image Interpretation*, volume LNCS 2766 of *Lecture Notes in Computer Science*. Springer.

Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1st edition.

Belouchrani, A., Abed-Meraim, K., Cardoso, J.-F., and Moulines, E. (1997). A blind source separation technique using second-order statistics. *Signal Processing, IEEE Transactions on*, 45(2):434–444.

Bengio, Y. (1991). *Artificial Neural Networks and their Application to Sequence Recognition*. PhD thesis, McGill University, (Computer Science), Montreal, Qc., Canada.

Bengio, Y. (2009). *Learning Deep Architectures for AI. Foundations and Trends in Machine Learning, V2(1)*. Now Publishers.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 19 (NIPS)*, pages 153–160. MIT Press.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Beringer, N., Graves, A., Schiel, F., and Schmidhuber, J. (2005). Classifying unprompted speech by retraining LSTM nets. In Duch, W., Kacprzyk, J., Oja, E., and Zadrozny, S., editors, *Artificial Neural Networks: Biological Inspirations - ICANN 2005, LNCS 3696*, pages 575–581. Springer-Verlag Berlin Heidelberg.

Bertsekas, D. P. (2001). *Dynamic Programming and Optimal Control*. Athena Scientific.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic Programming*. Athena Scientific, Belmont, MA.

Biegler-König, F. and Bärmann, F. (1993). A learning algorithm for multilayered neural networks based on linear least squares problems. *Neural Networks*, 6(1):127–131.

Bishop, C. M. (1993). Curvature-driven smoothing: A learning algorithm for feed-forward networks. *IEEE Transactions on Neural Networks*, 4(5):882–884.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blair, A. D. and Pollack, J. B. (1997). Analysis of dynamical recognizers. *Neural Computation*, 9(5):1127–1142.

Bluche, T., Louradour, J., Knibbe, M., Moysset, B., Benzeghiba, F., and Kermorvant., C. (2014). The A2iA Arabic Handwritten Text Recognition System at the OpenHaRT2013 Evaluation. In *International Workshop on Document Analysis Systems*.

Bobrowski, L. (1978). Learning processes in multilayer threshold nets. *Biological Cybernetics*, 31:1–6.

Bodén, M. and Wiles, J. (2000). Context-free and context-sensitive dynamics in recurrent neural networks. *Connection Science*, 12(3-4):197–210.

Bodenhausen, U. and Waibel, A. (1991). The tempo 2 algorithm: Adjusting time-delays by supervised learning. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 155–161. Morgan Kaufmann.

Bottou, L. (1991). *Une approche théorique de l'apprentissage connexioniste; applications à la reconnaissance de la parole*. PhD thesis, Université de Paris XI.

Bourlard, H. and Morgan, N. (1994). *Connnectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers.

Boutilier, C. and Poole, D. (1996). Computing optimal policies for partially observable Markov decision processes using compact representations. In *Proceedings of the AAAI, Portland, OR*.

Bradtke, S. J., Barto, A. G., and Kaelbling, P. (1996). Linear least-squares algorithms for temporal difference learning. In *Machine Learning*, pages 22–33.

Brafman, R. I. and Tennenholtz, M. (2002). R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

Breuel, T. M., Ul-Hasan, A., Al-Azawi, M. A., and Shafait, F. (2013). High-performance OCR for printed English and Fraktur using LSTM networks. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 683–687. IEEE.

Broyden, C. G. et al. (1965). A class of methods for solving nonlinear simultaneous equations. *Math. Comp*, 19(92):577–593.

Bryson, A. and Ho, Y. (1969). *Applied optimal control: optimization, estimation, and control*. Blaisdell Pub. Co.

Bryson, A. E. (1961). A gradient method for optimizing multi-stage allocation processes. In *Proc. Harvard Univ. Symposium on digital computers and their applications*.

Bryson, Jr., A. E. and Denham, W. F. (1961). A steepest-ascent method for solving optimum programming problems. Technical Report BR-1303, Raytheon Company, Missle and Space Division.

Buntine, W. L. and Weigend, A. S. (1991). Bayesian back-propagation. *Complex Systems*, 5:603–643.

Cardoso, J.-F. (1994). On the performance of orthogonal source separation algorithms. In *Proc. EUSIPCO*, pages 776–779.

Carter, M. J., Rudolph, F. J., and Nucci, A. J. (1990). Operational fault tolerance of CMAC networks. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 340–347. San Mateo, CA: Morgan Kaufmann.

Casey, M. P. (1996). The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8(6):1135–1178.

Cauwenberghs, G. (1993). A fast stochastic error-descent algorithm for supervised learning and optimization. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 5*, pages 244–244. Morgan Kaufmann.

Chaitin, G. J. (1966). On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569.

Chalup, S. K. and Blair, A. D. (2003). Incremental training of first order recurrent neural networks to predict a context-sensitive language. *Neural Networks*, 16(7):955–972.

Chellapilla, K., Puri, S., and Simard, P. (2006). High performance convolutional neural networks for document processing. In *International Workshop on Frontiers in Handwriting Recognition*.

Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363.

Ciresan, D. C., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012a). Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems NIPS*, pages 2852–2860.

Ciresan, D. C., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2013). Mitosis detection in breast cancer histology images with deep neural networks. In *MICCAI*, volume 2, pages 411–418.

Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep big simple neural nets for handwritten digit recogntion. *Neural Computation*, 22(12):3207–3220.

Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011a). Flexible, high performance convolutional neural networks for image classification. In *Intl. Joint Conference on Artificial Intelligence IJCAI*, pages 1237–1242.

Ciresan, D. C., Meier, U., Masci, J., and Schmidhuber, J. (2011b). A committee of neural networks for traffic sign classification. In *International Joint Conference on Neural Networks*, pages 1918–1921.

Ciresan, D. C., Meier, U., Masci, J., and Schmidhuber, J. (2012b). Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338.

Ciresan, D. C., Meier, U., and Schmidhuber, J. (2012c). Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2012*. Long preprint arXiv:1202.2745v1 [cs.CV].

Ciresan, D. C., Meier, U., and Schmidhuber, J. (2012d). Transfer learning for Latin and Chinese characters with deep neural networks. In *International Joint Conference on Neural Networks*, pages 1301–1306.

Ciresan, D. C. and Schmidhuber, J. (2013). Multi-column deep neural networks for offline handwritten Chinese character classification. Technical report, IDSIA. arXiv:1309.0261.

Clune, J., Stanley, K. O., Pennock, R. T., and Ofria, C. (2011). On the performance of indirect encoding across the continuum of regularity. *Trans. Evol. Comp*, 15(3):346–367.

Coates, A., Huval, B., Wang, T., Wu, D. J., Ng, A. Y., and Catanzaro, B. (2013). Deep learning with COTS HPC systems. In *Proc. International Conference on Machine learning (ICML'13)*.

Cochocki, A. and Unbehauen, R. (1993). *Neural networks for optimization and signal processing*. John Wiley & Sons, Inc.

Comon, P. (1994). Independent component analysis – a new concept? *Signal Processing*, 36(3):287–314.

Connor, J., Martin, D. R., and Atlas, L. E. (1994). Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254.

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing (STOC'71).*, page 151158. ACM, New York.

Cramer, N. L. (1985). A representation for the adaptive generation of simple sequential programs. In Grefenstette, J., editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications, Carnegie-Mellon University, July 24-26, 1985*, Hillsdale NJ. Lawrence Erlbaum Associates.

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, 31:377–403.

Cuccu, G., Luciw, M., Schmidhuber, J., and Gomez, F. (2011). Intrinsically motivated evolutionary search for vision-based reinforcement learning. In *Proceedings of the 2011 IEEE Conference on Development and Learning and Epigenetic Robotics IEEE-ICDL-EPIROB*, volume 2, pages 1–7. IEEE.

Dahl, G., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42.

Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving Deep Neural Networks for LVCSR using Rectified Linear Units and Dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE.

D'Ambrosio, D. B. and Stanley, K. O. (2007). A novel generative encoding for exploiting neural network sensor and output geometry. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO)*, pages 974–981.

Dayan, P. and Hinton, G. (1993). Feudal reinforcement learning. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 5*, pages 271–278. Morgan Kaufmann.

Dayan, P. and Hinton, G. E. (1996). Varieties of Helmholtz machine. *Neural Networks*, 9(8):1385–1403.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, 7:889–904.

Dayan, P. and Zemel, R. (1995). Competition and multiple cause models. *Neural Computation*, 7:565–579.

de Vries, B. and Principe, J. C. (1991). A theory for neural networks with time delays. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 162–168. Morgan Kaufmann.

Deco, G. and Parra, L. (1997). Non-linear feature extraction by redundancy reduction in an unsupervised stochastic neural network. *Neural Networks*, 10(4):683–691.

DeMers, D. and Cottrell, G. (1993). Non-linear dimensionality reduction. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pages 580–587. Morgan Kaufmann.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39.

Deng, L. and Yu, D. (2014). *Deep Learning: Methods and Applications*. NOW Publishers.

Dickmanns, E. D., Behringer, R., Dickmanns, D., Hildebrandt, T., Maurer, M., Thomanek, F., and Schiehlen, J. (1994). The seeing passenger car 'VaMoRs-P'. In *Proc. Int. Symp. on Intelligent Vehicles '94, Paris*, pages 68–73.

Dietterich, T. G. (2000a). Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer.

Dietterich, T. G. (2000b). Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artif. Intell. Res. (JAIR)*, 13:227–303.

Director, S. W. and Rohrer, R. A. (1969). Automated network design - the frequency-domain case. *IEEE Trans. Circuit Theory*, CT-16:330–337.

Doya, K., Samejima, K., ichi Katagiri, K., and Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Computation*, 14(6):1347–1369.

Dreyfus, S. E. (1962). The numerical solution of variational problems. *Journal of Mathematical Analysis and Applications*, 5(1):30–45.

Dreyfus, S. E. (1973). The computational solution of optimal control problems with time lag. *IEEE Transactions on Automatic Control*, 18(4):383–385.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning*, 12:2121–2159.

Egorova, A., Gloye, A., Göktekin, C., Liers, A., Luft, M., Rojas, R., Simon, M., Tenchio, O., and Wiesel, F. (2004). FU-Fighters Small Size 2004, Team Description. RoboCup 2004 Symposium: Papers and Team Description Papers. CD edition.

Elman, J. L. (1988). Finding structure in time. Technical Report CRL 8801, Center for Research in Language, University of California, San Diego.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660.

Eubank, R. L. (1988). Spline smoothing and nonparametric regression. In Farlow, S., editor, *Self-Organizing Methods in Modeling*. Marcel Dekker, New York.

Euler, L. (1744). *Methodus inveniendi*.

Faggin, F. (1992). Neural network hardware. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 1, page 153.

Fahlman, S. E. (1988). An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, Carnegie-Mellon Univ.

Fahlman, S. E. (1991). The recurrent cascade-correlation learning algorithm. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 190–196. Morgan Kaufmann.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929.

Fernández, S., Graves, A., and Schmidhuber, J. (2007). An application of recurrent neural networks to discriminative keyword spotting. In *Proc. ICANN (2)*, pages 220–229.

Fernandez, S., Graves, A., and Schmidhuber, J. (2007). Sequence labelling in structured domains with hierarchical recurrent neural networks. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

Field, D. J. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America*, 4:2379–2394.

Field, D. J. (1994). What is the goal of sensory coding? *Neural Computation*, 6:559–601.

Fletcher, R. and Powell, M. J. (1963). A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168.

Fogel, L., Owens, A., and Walsh, M. (1966). *Artificial Intelligence through Simulated Evolution*. Willey, New York.

Földiák, P. (1990). Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–170.

Földiák, P. and Young, M. P. (1995). Sparse coding in the primate cortex. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 895–898. The MIT Press.

Förster, A., Graves, A., and Schmidhuber, J. (2007). RNN-based Learning of Compact Maps for Efficient Robot Localization. In *15th European Symposium on Artificial Neural Networks, ESANN*, pages 537–542, Bruges, Belgium.

Franzius, M., Sprekeler, H., and Wiskott, L. (2007). Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):166.

Frinken, V., Zamora-Martinez, F., Espana-Boquera, S., Castro-Bleda, M. J., Fischer, A., and Bunke, H. (2012). Long-short term memory neural networks language modeling for handwriting recognition. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 701–704. IEEE.

Fritzke, B. (1994). A growing neural gas network learns topologies. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *NIPS*, pages 625–632. MIT Press.

Fu, K. S. (1977). *Syntactic Pattern Recognition and Applications*. Berlin, Springer.

Fukada, T., Schuster, M., and Sagisaka, Y. (1999). Phoneme boundary estimation using bidirectional recurrent neural networks and its applications. *Systems and Computers in Japan*, 30(4):20–30.

Fukushima, K. (1979). Neural network model for a mechanism of pattern recognition unaffected by shift in position - Neocognitron. *Trans. IECE*, J62-A(10):658–665.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.

Fukushima, K. (2011). Increasing robustness against background noise: visual pattern recognition by a Neocognitron. *Neural Networks*, 24(7):767–778.

Fukushima, K. (2013a). Artificial vision by multi-layered neural networks: Neocognitron and its advances. *Neural Networks*, 37:103–119.

Fukushima, K. (2013b). Training multi-layered neural network Neocognitron. *Neural Networks*, 40:18–31.

Gauss, C. F. (1821). *Theoria combinationis observationum erroribus minimis obnoxiae (Theory of the combination of observations least subject to error)*.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58.

Gers, F. A. and Schmidhuber, J. (2000). Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, pages 189–194. IEEE.

Gers, F. A. and Schmidhuber, J. (2001). LSTM recurrent networks learn simple context free and context sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Gers, F. A., Schraudolph, N., and Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143.

Ghavamzadeh, M. and Mahadevan, S. (2003). Hierarchical policy gradient algorithms. In *Proceedings of the Twentieth Conference on Machine Learning (ICML-2003)*, pages 226–233.

Gherrity, M. (1989). A learning algorithm for analog fully recurrent neural networks. In *IEEE/INNS International Joint Conference on Neural Networks, San Diego*, volume 1, pages 643–644.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. Technical Report arxiv.org/abs/1311.2524, UC Berkeley and ICSI.

Gisslen, L., Luciw, M., Graziano, V., and Schmidhuber, J. (2011). Sequential constant size compressor for reinforcement learning. In *Proc. Fourth Conference on Artificial General Intelligence (AGI), Google, Mountain View, CA*, pages 31–40. Springer.

Glasmachers, T., Schaul, T., Sun, Y., Wierstra, D., and Schmidhuber, J. (2010). Exponential Natural Evolution Strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 393–400. ACM.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier networks. In *AISTATS*, volume 15, pages 315–323.

Gloye, A., Wiesel, F., Tenchio, O., and Simon, M. (2005). Reinforcing the driving quality of soccer playing robots by anticipation. *IT - Information Technology*, 47(5).

Gödel, K. (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26.

Golub, G., Heath, H., and Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–224.

Gomez, F. J. (2003). *Robust Nonlinear Control through Neuroevolution*. PhD thesis, Department of Computer Sciences, University of Texas at Austin.

Gomez, F. J. and Miikkulainen, R. (2003). Active guidance for a finless rocket using neuroevolution. In *Proc. GECCO 2003, Chicago*.

Gomez, F. J. and Schmidhuber, J. (2005). Co-evolving recurrent neurons learn deep memory POMDPs. In *Proc. of the 2005 conference on genetic and evolutionary computation (GECCO), Washington, D. C.* ACM Press, New York, NY, USA.

Gomez, F. J., Schmidhuber, J., and Miikkulainen, R. (2008). Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 9(May):937–965.

Goodfellow, I., Mirza, M., Da, X., Courville, A., and Bengio, Y. (2014). An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *TR arXiv:1312.6211v2*.

Goodfellow, I. J., Courville, A., and Bengio, Y. (2011). Spike-and-slab sparse coding for unsupervised feature discovery. In *NIPS Workshop on Challenges in Learning Hierarchical Models*.

Goodfellow, I. J., Courville, A. C., and Bengio, Y. (2012). Large-scale feature learning with spike-and-slab sparse coding. In *Proceedings of the 29th International Conference on Machine Learning*.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. In *International Conference on Machine Learning (ICML)*.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2348–2356.

Graves, A., Eck, D., Beringer, N., and Schmidhuber, J. (2003). Isolated digit recognition with LSTM recurrent networks. In *First International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, Lausanne.

Graves, A., Fernandez, S., Gomez, F. J., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural nets. In *ICML'06: Proceedings of the 23rd International Conference on Machine Learning*, pages 369–376.

Graves, A., Fernandez, S., Liwicki, M., Bunke, H., and Schmidhuber, J. (2008). Unconstrained on-line handwriting recognition with recurrent neural networks. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 577–584. MIT Press, Cambridge, MA.

Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for improved unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5).

Graves, A., Mohamed, A.-r., and Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems 21*, pages 545–552. MIT Press, Cambridge, MA.

Graziano, M. (2009). *The Intelligent Movement Machine: An Ethological Perspective on the Primate Motor System*. Oxford University Press, USA.

Griewank, A. (2012). *Documenta Mathematica - Extra Volume ISMP*, pages 389–400.

Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(6):1291–1307.

Grossberg, S. (1969). Some networks that can learn, remember, and reproduce any number of complicated space-time patterns, I. *Journal of Mathematics and Mechanics*, 19:53–91.

Grossberg, S. (1976a). Adaptive pattern classification and universal recoding, 1: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:187–202.

Grossberg, S. (1976b). Adaptive pattern classification and universal recoding, 2: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics*, 23.

Gruau, F., Whitley, D., and Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. NeuroCOLT Technical Report NC-TR-96-048, ESPRIT Working Group in Neural and Computational Learning, NeuroCOLT 8556.

Grüttner, M., Sehnke, F., Schaul, T., and Schmidhuber, J. (2010). Multi-Dimensional Deep Memory Atari-Go Players for Parameter Exploring Policy Gradients. In *Proceedings of the International Conference on Artificial Neural Networks ICANN*, pages 114–123. Springer.

Guyon, I., Vapnik, V., Boser, B., Bottou, L., and Solla, S. A. (1992). Structural risk minimization for character recognition. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 4*, pages 471–479. Morgan Kaufmann.

Hadamard, J. (1908). *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*. Mémoires présentés par divers savants à l'Académie des sciences de l'Institut de France: Éxtrait. Imprimerie nationale.

Hansen, N., Müller, S. D., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Hanson, S. J. and Pratt, L. Y. (1989). Comparing biases for minimal network construction with backpropagation. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1*, pages 177–185. San Mateo, CA: Morgan Kaufmann.

Hashem, S. and Schmeiser, B. (1992). Improving model accuracy using optimal linear combinations of trained neural networks. *IEEE Transactions on Neural Networks*, 6:792–794.

Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 5*, pages 164–171. Morgan Kaufmann.

Hastie, T. J. and Tibshirani, R. J. (1990). Generalized additive models. *Monographs on Statisics and Applied Probability*, 43.

Hawkins, J. and George, D. (2006). *Hierarchical Temporal Memory - Concepts, Theory, and Terminology*. Numenta Inc.

Hebb, D. O. (1949). *The Organization of Behavior*. Wiley, New York.

Heemskerk, J. N. (1995). Overview of neural hardware. *Neurocomputers for Brain-Style Processing. Design, Implementation and Application*.

Hertz, J., Krogh, A., and Palmer, R. (1991). *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City.

Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–436.

Hihi, S. E. and Bengio, Y. (1996). Hierarchical recurrent neural networks for long-term dependencies. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 493–499. MIT Press.

Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial intelligence*, 40(1):185–234.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995). The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1160.

Hinton, G. E., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012a). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97.

Hinton, G. E. and Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society* **B**, 352:1177–1190.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hinton, G. E. and Sejnowski, T. E. (1986). Learning and relearning in Boltzmann machines. In *Parallel Distributed Processing*, volume 1, pages 282–317. MIT Press.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. Technical Report arXiv:1207.0580.

Hinton, G. E. and van Camp, D. (1993). Keeping neural networks simple. In *Proceedings of the International Conference on Artificial Neural Networks, Amsterdam*, pages 11–18. Springer.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München. Advisor: J. Schmidhuber.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001a). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer, S. C. and Kolen, J. F., editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

Hochreiter, S. and Obermayer, K. (2005). Sequence classification for protein analysis. In *Snowbird Workshop*, Snowbird, Utah. Computational and Biological Learning Society.

Hochreiter, S. and Schmidhuber, J. (1997a). Flat minima. *Neural Computation*, 9(1):1–42.

Hochreiter, S. and Schmidhuber, J. (1997b). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780. Based on TR FKI-207-95, TUM (1995).

Hochreiter, S. and Schmidhuber, J. (1999). Feature extraction through LOCOCODE. *Neural Computation*, 11(3):679–714.

Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001b). Learning to learn using gradient descent. In *Lecture Notes on Comp. Sci. 2130, Proc. Intl. Conf. on Artificial Neural Networks (ICANN-2001)*, pages 87–94. Springer: Berlin, Heidelberg.

Holden, S. B. (1994). *On the Theory of Generalization and Self-Structuring in Linearly Weighted Connectionist Networks*. PhD thesis, Cambridge University, Engineering Department.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences*, 79:2554–2558.

Hubel, D. H. and Wiesel, T. (1962). Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, 160:106–154.

Huffman, D. A. (1952). A method for construction of minimum-redundancy codes. *Proceedings IRE*, 40:1098–1101.

Hutter, M. (2002). The fastest and shortest algorithm for all well-defined problems. *International Journal of Foundations of Computer Science*, 13(3):431–443. (On J. Schmidhuber's SNF grant 20-61847).

Hutter, M. (2005). *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin. (On J. Schmidhuber's SNF grant 20-61847).

Hyvarinen, A., Hoyer, P., and Oja, E. (1999). Sparse code shrinkage: Denoising by maximum likelihood estimation. In Kearns, M., Solla, S. A., and Cohn, D., editors, *Advances in Neural Information Processing Systems 12*. MIT Press.

Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430.

ICPR 2012 Contest on Mitosis Detection in Breast Cancer Histological Images. (2012). IPAL Laboratory and TRIBVN Company and Pitie-Salpetriere Hospital and CIALAB of Ohio State Univ., http://ipal.cnrs.fr/ICPR2012/.

Igel, C. (2003). Neuroevolution for reinforcement learning using evolution strategies. In Reynolds, R., Abbass, H., Tan, K. C., Mckay, B., Essam, D., and Gedeon, T., editors, *Congress on Evolutionary Computation (CEC 2003)*, volume 4, pages 2588–2595. IEEE.

Indermuhle, E., Frinken, V., and Bunke, H. (2012). Mode detection in online handwritten documents using BLSTM neural networks. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 302–307. IEEE.

Indermuhle, E., Frinken, V., Fischer, A., and Bunke, H. (2011). Keyword spotting in online handwritten documents containing text and non-text using BLSTM neural networks. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 73–77. IEEE.

Jaakkola, T., Singh, S. P., and Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 345–352. MIT Press.

Jackel, L., Boser, B., Graf, H.-P., Denker, J., LeCun, Y., Henderson, D., Matan, O., Howard, R., and Baird, H. (1990). VLSI implementation of electronic neural networks: and example in character recognition. In IEEE, editor, *IEEE International Conference on Systems, Man, and Cybernetics*, pages 320–322, Los Angeles, CA.

Jacob, C., Lindenmayer, A., and Rozenberg, G. (1994). Genetic L-System Programming. In *Parallel Problem Solving from Nature III*, Lecture Notes in Computer Science.

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295–307.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology.

Jaeger, H. (2002). Short term memory in echo state networks. GMD-Report 152, GMD - German National Research Institute for Computer Science.

Jaeger, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80.

Jim, K., Giles, C. L., and Horne, B. G. (1995). Effects of noise on convergence and generalization in recurrent networks. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems (NIPS) 7*, page 649. San Mateo, CA: Morgan Kaufmann.

Jodogne, S. R. and Piater, J. H. (2007). Closed-loop learning of visual control policies. *J. Artificial Intelligence Research*, 28:349–391.

Jordan, M. I. (1986). Serial order: A parallel distributed processing approach. Technical Report ICS Report 8604, Institute for Cognitive Science, University of California, San Diego.

Jordan, M. I. (1988). Supervised learning and systems with excess degrees of freedom. Technical Report COINS TR 88-27, Massachusetts Institute of Technology.

Jordan, M. I. and Rumelhart, D. E. (1990). Supervised learning with a distal teacher. Technical Report Occasional Paper #40, Center for Cog. Sci., Massachusetts Institute of Technology.

Jordan, M. I. and Sejnowski, T. J. (2001). *Graphical models: Foundations of neural computation*. MIT press.

Jutten, C. and Herault, J. (1991). Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1–10.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1995). Planning and acting in partially observable stochastic domains. Technical report, Brown University, Providence RI.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of AI research*, 4:237–285.

Kalinke, Y. and Lehmann, H. (1998). Computation in recurrent neural networks: From counters to iterated function systems. In Antoniou, G. and Slaney, J., editors, *Advanced Topics in Artificial Intelligence, Proceedings of the 11th Australian Joint Conference on Artificial Intelligence*, volume 1502 of *LNAI*, Berlin, Heidelberg. Springer.

Kelley, H. J. (1960). Gradient theory of optimal flight paths. *ARS Journal*, 30(10):947–954.

Kerlirzin, P. and Vallet, F. (1993). Robustness in multilayer perceptrons. *Neural Computation*, 5(1):473–482.

Kimura, H., Miyazaki, K., and Kobayashi, S. (1997). Reinforcement learning in POMDPs with function approximation. In *ICML*, volume 97, pages 152–160.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476.

Klapper-Rybicka, M., Schraudolph, N. N., and Schmidhuber, J. (2001). Unsupervised learning in LSTM recurrent neural networks. In *Lecture Notes on Comp. Sci. 2130, Proc. Intl. Conf. on Artificial Neural Networks (ICANN-2001)*, pages 684–691. Springer: Berlin, Heidelberg.

Kohl, N. and Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 2619–2624. IEEE.

Kohonen, T. (1972). Correlation matrix memories. *Computers, IEEE Transactions on*, 100(4):353–359.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

Kohonen, T. (1988). *Self-Organization and Associative Memory*. Springer, second edition.

Kolmogorov, A. N. (1965a). On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Doklady Akademii. Nauk USSR,*, 114:679–681.

Kolmogorov, A. N. (1965b). Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–11.

Kompella, V. R., Luciw, M. D., and Schmidhuber, J. (2012). Incremental slow feature analysis: Adaptive low-complexity slow feature updating from high-dimensional input streams. *Neural Computation*, 24(11):2994–3024.

Korkin, M., de Garis, H., Gers, F., and Hemmi, H. (1997). CBM (CAM-Brain Machine) - a hardware tool which evolves a neural net module in a fraction of a second and runs a million neuron artificial brain in real time.

Koutník, J., Cuccu, G., Schmidhuber, J., and Gomez, F. (July 2013). Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1061–1068, Amsterdam. ACM.

Koutník, J., Gomez, F., and Schmidhuber, J. (2010). Evolving neural networks in compressed weight space. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 619–626.

Koutník, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014). A Clockwork RNN. Technical Report arXiv:1402.3511 [cs.NE], The Swiss AI Lab IDSIA.

Kramer, M. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233–243.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS 2012)*, page 4.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan Kaufmann.

Kurzweil, R. (2012). *How to Create a Mind: The Secret of Human Thought Revealed*.

Lagoudakis, M. G. and Parr, R. (2003). Least-squares policy iteration. *JMLR*, 4:1107–1149.

Lang, K., Waibel, A., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43.

Lange, S. and Riedmiller, M. (2010). Deep auto-encoder neural networks in reinforcement learning. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8.

Lapedes, A. and Farber, R. (1986). A self-optimizing, nonsymmetrical neural net for content addressable memory and pattern recognition. *Physica D*, 22:247–259.

Larraanaga, P. and Lozano, J. A. (2001). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Norwell, MA, USA.

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Corrado, G., Chen, K., Dean, J., and Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. In *Proc. ICML'12*.

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau à seuil asymétrique. *Proceedings of Cognitiva 85, Paris*, pages 599–604.

LeCun, Y. (1988). A theoretical framework for back-propagation. In Touretzky, D., Hinton, G., and Sejnowski, T., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 21–28, CMU, Pittsburgh, Pa. Morgan Kaufmann.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Back-propagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990a). Handwritten digit recognition with a back-propagation network. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan Kaufmann.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Denker, J. S., and Solla, S. A. (1990b). Optimal brain damage. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan Kaufmann.

LeCun, Y., Muller, U., Cosatto, E., and Flepp, B. (2006). Off-road obstacle avoidance through end-to-end learning. In *Advances in Neural Information Processing Systems (NIPS 2005)*.

LeCun, Y., Simard, P., and Pearlmutter, B. (1993). Automatic learning rate maximization by on-line estimation of the Hessian's eigenvectors. In Hanson, S., Cowan, J., and Giles, L., editors, *Advances in Neural Information Processing Systems (NIPS 1992)*, volume 5. Morgan Kaufmann Publishers, San Mateo, CA.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007a). Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems 19*, pages 801–808.

Lee, H., Ekanadham, C., and Ng, A. Y. (2007b). Sparse deep belief net model for visual area V2. In *Advances in Neural Information Processing Systems (NIPS)*, volume 7, pages 873–880.

Lee, L. (1996). Learning of context-free languages: A survey of the literature. Technical Report TR-12-96, Center for Research in Computing Technology, Harvard University, Cambridge, Massachusetts.

Legenstein, R., Wilbert, N., and Wiskott, L. (2010). Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, 6(8).

Legenstein, R. A. and Maass, W. (2002). Neural circuits for pattern recognition with small total wire length. *Theor. Comput. Sci.*, 287(1):239–249.

Leibniz, G. W. (1676). Memoir using the chain rule (cited in TMME 7:2&3 p 321-332, 2010).

Lenat, D. B. (1983). Theory formation by heuristic search. *Machine Learning*, 21.

Lenat, D. B. and Brown, J. S. (1984). Why AM an EURISKO appear to work. *Artificial Intelligence*, 23(3):269–294.

Levenberg, K. (1944). A method for the solution of certain problems in least squares. *Quarterly of applied mathematics*, 2:164–168.

Levin, A. U., Leen, T. K., and Moody, J. E. (1994). Fast pruning using principal components. In *Advances in Neural Information Processing Systems 6*, page 35. Morgan Kaufmann.

Levin, A. U. and Narendra, K. S. (1995). Control of nonlinear dynamical systems using neural networks. ii. observability, identification, and control. *IEEE transactions on neural networks/a publication of the IEEE Neural Networks Council*, 7(1):30–42.

Levin, L. A. (1973a). On the notion of a random sequence. *Soviet Math. Dokl.*, 14(5):1413–1416.

Levin, L. A. (1973b). Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266.

Lewicki, M. S. and Olshausen, B. A. (1998). Inferring sparse, overcomplete image codes using an efficient coding framework. In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, *Advances in Neural Information Processing Systems 10*, pages 815–821.

L'Hôpital, G. F. A. (1696). *Analyse des infiniment petits, pour l'intelligence des lignes courbes*. Paris: L'Imprimerie Royale.

Li, M. and Vitányi, P. M. B. (1997). *An Introduction to Kolmogorov Complexity and its Applications (2nd edition)*. Springer.

Lin, L. (1993). *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh.

Lin, T., Horne, B., Tino, P., and Giles, C. (1996). Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338.

Lin, T., Horne, B. G., Tino, P., and Giles, C. L. (1995). Learning long-term dependencies is not as difficult with NARX recurrent neural networks. Technical Report UMIACS-TR-95-78 and CS-TR-3500, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

Lindenmayer, A. (1968). Mathematical models for cellular interaction in development. *J. Theoret. Biology*, 18:280–315.

Linnainmaa, S. (1970). The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. Master's thesis, Univ. Helsinki.

Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16(2):146–160.

Linsker, R. (1988). Self-organization in a perceptual network. *IEEE Computer*, 21:105–117.

Littman, M. L. (1996). *Algorithms for Sequential Decision Making*. PhD thesis, Brown University.

Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In Prieditis, A. and Russell, S., editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 362–370. Morgan Kaufmann Publishers, San Francisco, CA.

Ljung, L. (1998). *System identification*. Springer.

Loiacono, D., Cardamone, L., and Lanzi, P. L. (2011). Simulated car racing championship competition software manual. Technical report, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy.

Loiacono, D., Lanzi, P. L., Togelius, J., Onieva, E., Pelta, D. A., Butz, M. V., Lönneker, T. D., Cardamone, L., Perez, D., Sáez, Y., Preuss, M., and Quadflieg, J. (2009). The 2009 simulated car racing championship.

Luciw, M., Kompella, V. R., Kazerounian, S., and Schmidhuber, J. (2013). An intrinsic value system for developing multiple invariant representations with incremental slowness learning. *Frontiers in Neurorobotics*, 7(9).

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*.

Maass, W. (2000). On the computational power of winner-take-all. *Neural Computation*, 12:2519–2535.

Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560.

MacKay, D. J. C. (1992). A practical Bayesian framework for backprop networks. *Neural Computation*, 4:448–472.

MacKay, D. J. C. and Miller, K. D. (1990). Analysis of Linsker's simulation of Hebbian rules. *Neural Computation*, 2:173–187.

Maei, H. R. and Sutton, R. S. (2010). GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the Third Conference on Artificial General Intelligence*, volume 1, pages 91–96.

Mahadevan, S. (1996). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22:159.

Manolios, P. and Fanelli, R. (1994). First-order recurrent neural networks and deterministic finite state automata. *Neural Computation*, 6:1155–1173.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441.

Martens, J. (2010). Deep learning via Hessian-free optimization. In Fürnkranz, J. and Joachims, T., editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742, Haifa, Israel. Omnipress.

Martens, J. and Sutskever, I. (2011). Learning recurrent neural networks with Hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1033–1040.

Martinetz, T. M., Ritter, H. J., and Schulten, K. J. (1990). Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Transactions on Neural Networks*, 1(1):131–136.

Masci, J., Ciresan, D. C., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2013a). Fast image scanning with deep max-pooling convolutional neural networks. In *Proc. ICIP*.

Masci, J., Giusti, A., Ciresan, D. C., Fricout, G., and Schmidhuber, J. (2013b). A fast learning algorithm for image segmentation with max-pooling convolutional networks. In *International Conference on Image Processing (ICIP13)*, pages 2713–2717.

Matsuoka, K. (1992). Noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):436–440.

Mayer, H., Gomez, F., Wierstra, D., Nagy, I., Knoll, A., and Schmidhuber, J. (2008). A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics*, 22(13-14):1521–1537.

McCallum, R. A. (1996). Learning to use selective attention and short-term memory in sequential tasks. In Maes, P., Mataric, M., Meyer, J.-A., Pollack, J., and Wilson, S. W., editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, Cambridge, MA*, pages 315–324. MIT Press, Bradford Books.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 7:115–133.

Melnik, O., Levy, S. D., and Pollack, J. B. (2000). RAAM for infinite context-free languages. In *Proc. IJCNN (5)*, pages 585–590.

Menache, I., Mannor, S., and Shimkin, N. (2002). Q-cut - dynamic discovery of sub-goals in reinforcement learning. In *Proc. ECML'02*, pages 295–306.

Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P., Courville, A., and Bergstra, J. (2011). Unsupervised and transfer learning challenge: a deep learning approach. In *JMLR W&CP: Proc. Unsupervised and Transfer Learning*, volume 7.

Meuleau, N., Peshkin, L., Kim, K. E., and Kaelbling, L. P. (1999). Learning finite state controllers for partially observable environments. In *15th International Conference of Uncertainty in AI*, pages 427–436.

Miglino, O., Lund, H., and Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434.

Miller, G., Todd, P., and Hedge, S. (1989). Designing neural networks using genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 379–384. Morgan Kauffman.

Miller, K. D. (1994). A model for the development of simple cell receptive fields and the ordered arrangement of orientation columns through activity-dependent competition between on- and off-center inputs. *Journal of Neuroscience*, 14(1):409–441.

Minai, A. A. and Williams, R. D. (1994). Perturbation response in feedforward networks. *Neural Networks*, 7(5):783–796.

Minsky, M. (1963). Steps toward artificial intelligence. In Feigenbaum, E. and Feldman, J., editors, *Computers and Thought*, pages 406–450. McGraw-Hill, New York.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (Dec 2013). Playing Atari with deep reinforcement learning. Technical Report arXiv:1312.5602 [cs.LG], Deepmind Technologies.

Mohamed, A., Dahl, G. E., and Hinton, G. E. (2009). Deep belief networks for phone recognition. In *NIPS'22 workshop on deep learning for speech recognition*.

Mohamed, A. and Hinton, G. E. (2010). Phone recognition using restricted Boltzmann machines. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4354–4357.

Molgedey, L. and Schuster, H. G. (1994). Separation of independent signals using time-delayed correlations. *Phys. Reviews Letters*, 72(23):3634–3637.

Møller, M. F. (1993). Exact calculation of the product of the Hessian matrix of feed-forward network error functions and a vector in O(N) time. Technical Report PB-432, Computer Science Department, Aarhus University, Denmark.

Montavon, G., Orr, G., and Müller, K. (2012). *Neural Networks: Tricks of the Trade*. Number LNCS 7700 in Lecture Notes in Computer Science Series. Springer Verlag.

Moody, J. E. (1989). Fast learning in multi-resolution hierarchies. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1*, pages 29–39. Morgan Kaufmann.

Moody, J. E. (1992). The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 4*, pages 847–854. Morgan Kaufmann.

Moody, J. E. and Utans, J. (1994). Architecture selection strategies for neural networks: Application to corporate bond rating prediction. In Refenes, A. N., editor, *Neural Networks in the Capital Markets*. John Wiley & Sons.

Moore, A. and Atkeson, C. (1995). The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning*, 21(3):199–233.

Moore, A. and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130.

Moriarty, D. E. (1997). *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin.

Moriarty, D. E. and Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32.

Morimoto, J. and Doya, K. (2000). Robust reinforcement learning. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 1061–1067. MIT Press.

Mosteller, F. and Tukey, J. W. (1968). Data analysis, including statistics. In Lindzey, G. and Aronson, E., editors, *Handbook of Social Psychology, Vol. 2*. Addison-Wesley.

Mozer, M. C. (1989). A focused back-propagation algorithm for temporal sequence recognition. *Complex Systems*, 3:349–381.

Mozer, M. C. (1991). Discovering discrete distributed representations with iterative competitive learning. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 627–634. Morgan Kaufmann.

Mozer, M. C. (1992). Induction of multiscale temporal structure. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 4*, pages 275–282. Morgan Kaufmann.

Mozer, M. C. and Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1*, pages 107–115. Morgan Kaufmann.

Munro, P. W. (1987). A dual back-propagation scheme for scalar reinforcement learning. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society, Seattle, WA*, pages 165–176.

Murray, A. F. and Edwards, P. J. (1993). Synaptic weight noise during MLP learning enhances fault-tolerance, generalisation and learning trajectory. In S. J. Hanson, J. D. C. and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pages 491–498. San Mateo, CA: Morgan Kaufmann.

Nadal, J.-P. and Parga, N. (1994). Non-linear neurons in the low noise limit: a factorial code maximises information transfer. *Network*, 5:565–581.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*.

Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *Neural Networks, IEEE Transactions on*, 1(1):4–27.

Narendra, K. S. and Thathatchar, M. A. L. (1974). Learning automata - a survey. *IEEE Transactions on Systems, Man, and Cybernetics*, 4:323–334.

Neal, R. M. (2006). Classification with Bayesian neural networks. In Quinonero-Candela, J., Magnini, B., Dagan, I., and D'Alche-Buc, F., editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*, volume 3944 of *Lecture Notes in Computer Science*, pages 28–32. Springer.

Neal, R. M. and Zhang, J. (2006). High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees. In Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. A., editors, *Feature Extraction: Foundations and Applications, Studies in Fuzziness and Soft Computing*, pages 265–295. Springer.

Neti, C., Schneider, M. H., and Young, E. D. (1992). Maximally fault tolerant neural networks. In *IEEE Transactions on Neural Networks*, volume 3, pages 14–23.

Neuneier, R. and Zimmermann, H.-G. (1996). How to train neural networks. In Orr, G. B. and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 373–423. Springer.

Nguyen, N. and Widrow, B. (1989). The truck backer-upper: An example of self learning in neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 357–363. IEEE Press.

Nicholas Refenes, A., Zapranis, A., and Francis, G. (1994). Stock performance modeling using neural networks: a comparative study with regression models. *Neural Networks*, 7(2):375–388.

Nilsson, N. J. (1980). *Principles of artificial intelligence*. Morgan Kaufmann, San Francisco, CA, USA.

Nolfi, S., Floreano, D., Miglino, O., and Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. In Brooks, R. A. and Maes, P., editors, *Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*, pages 190–197. MIT.

Nowlan, S. J. and Hinton, G. E. (1992). Simplifying neural networks by soft weight sharing. *Neural Computation*, 4:173–193.

Oh, K.-S. and Jung, K. (2004). GPU implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314.

Oja, E. (1989). Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1(1):61–68.

Oja, E. (1991). Data compression, feature extraction, and autoassociation in feedforward neural networks. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, volume 1, pages 737–745. Elsevier Science Publishers B.V., North-Holland.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609.

Omlin, C. and Giles, C. L. (1996). Extraction of rules from discrete-time recurrent neural networks. *Neural Networks*, 9(1):41–52.

O'Reilly, R. (2003). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. Technical Report ICS-03-03, ICS.

Orr, G. and Müller, K. (1998). *Neural Networks: Tricks of the Trade*. Number LNCS 1524 in Lecture Notes in Computer Science Series. Springer Verlag.

Ostrovskii, G. M., Volin, Y. M., and Borisov, W. W. (1971). Über die Berechnung von Ableitungen. *Wiss. Z. Tech. Hochschule für Chemie*, 13:382–384.

Otte, S., Krechel, D., Liwicki, M., and Dengel, A. (2012). Local feature based online mode detection with recurrent neural networks. In *Proceedings of the 2012 International Conference on Frontiers in Handwriting Recognition*, pages 533–537. IEEE Computer Society.

Oudeyer, P.-Y., Baranes, A., and Kaplan, F. (2013). Intrinsically motivated learning of real world sensorimotor skills with developmental constraints. In Baldassarre, G. and Mirolli, M., editors, *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer.

Pachitariu, M. and Sahani, M. (2013). Regularization and nonlinearities for neural language models: when are they needed? *arXiv preprint arXiv:1301.5650*.

Palm, G. (1980). On associative memory. *Biological Cybernetics*, 36.

Palm, G. (1992). On the information storage capacity of local learning rules. *Neural Computation*, 4(2):703–711.

Parker, D. B. (1985). Learning-logic. Technical Report TR-47, Center for Comp. Research in Economics and Management Sci., MIT.

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2013a). How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013b). On the difficulty of training recurrent neural networks. In *ICML'13: JMLR: W&CP volume 28*.

Pasemann, F., Steinmetz, U., and Dieckman, U. (1999). Evolving structure and function of neurocontrollers. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1973–1978, Mayflower Hotel, Washington D.C., USA. IEEE Press.

Pearlmutter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269.

Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160.

Pearlmutter, B. A. (1995). Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228.

Pearlmutter, B. A. and Hinton, G. E. (1986). G-maximization: An unsupervised learning procedure for discovering regularities. In Denker, J. S., editor, *Neural Networks for Computing: American Institute of Physics Conference Proceedings 151*, volume 2, pages 333–338.

Peng, J. and Williams, R. J. (1996). Incremental multi-step Q-learning. *Machine Learning*, 22:283–290.

Pérez-Ortiz, J. A., Gers, F. A., Eck, D., and Schmidhuber, J. (2003). Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks*, (16):241–250.

Peters, J. (2010). Policy gradient methods. *Scholarpedia*, 5(11):3698.

Peters, J. and Schaal, S. (2008a). Natural actor-critic. *Neurocomputing*, 71:1180–1190.

Peters, J. and Schaal, S. (2008b). Reinforcement learning of motor skills with policy gradients. *Neural Network*, 21(4):682–697.

Pham, V., Kermorvant, C., and Louradour, J. (2013). Dropout Improves Recurrent Neural Networks for Handwriting Recognition. *arXiv preprint arXiv:1312.4569*.

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 19(59):2229–2232.

Plate, T. A. (1993). Holographic recurrent networks. In S. J. Hanson, J. D. C. and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pages 34–41. Morgan Kaufmann.

Plumbley, M. D. (1991). On information theory and unsupervised neural networks. Dissertation, published as technical report CUED/F-INFENG/TR.78, Engineering Department, Cambridge University.

Pollack, J. B. (1988). Implications of recursive distributed representations. In *Proc. NIPS*, pages 527–536.

Pollack, J. B. (1990). Recursive distributed representation. *Artificial Intelligence*, 46:77–105.

Pontryagin, L. S., Boltyanskii, V. G., Gamrelidze, R. V., and Mishchenko, E. F. (1961). *The Mathematical Theory of Optimal Processes*.

Post, E. L. (1936). Finite combinatory processes-formulation 1. *The Journal of Symbolic Logic*, 1(3):103–105.

Precup, D., Sutton, R. S., and Singh, S. (1998). Multi-time models for temporally abstract planning. pages 1050–1056. Morgan Kaufmann.

Raina, R., Madhavan, A., and Ng, A. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 873–880. ACM.

Ramacher, U., Raab, W., Anlauf, J., Hachmann, U., Beichter, J., Bruels, N., Wesseling, M., Sicheneder, E., Maenner, R., Glaess, J., and Wurz, A. (1993). Multiprocessor and memory architecture of the neurocomputer SYNAPSE-1. *International Journal of Neural Systems*, 04(04):333–336.

Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In et al., J. P., editor, *Advances in Neural Information Processing Systems (NIPS 2006)*. MIT Press.

Ranzato, M. A., Huang, F., Boureau, Y., and LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07)*, pages 1–8. IEEE Press.

Rechenberg, I. (1971). Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Dissertation. Published 1973 by Fromman-Holzboog.

Redlich, A. N. (1993). Redundancy reduction as a strategy for unsupervised learning. *Neural Computation*, 5:289–304.

Riedmiller, M. (2005). Neural fitted Q iteration—first experiences with a data efficient neural reinforcement learning method. In *Proc. ECML-2005*, pages 317–328. Springer-Verlag Berlin Heidelberg.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The Rprop algorithm. In *Proc. IJCNN*, pages 586–591. IEEE Press.

Riedmiller, M., Lange, S., and Voigtlaender, A. (2012). Autonomous reinforcement learning on raw visual input data in a real world application. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Brisbane, Australia.

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nat. Neurosci.*, 2(11):1019–1025.

Ring, M., Schaul, T., and Schmidhuber, J. (2011). The two-dimensional organization of behavior. In *Proceedings of the First Joint Conference on Development Learning and on Epigenetic Robotics ICDL-EPIROB*, Frankfurt.

Ring, M. B. (1991). Incremental development of complex behaviors through automatic construction of sensory-motor hierarchies. In Birnbaum, L. and Collins, G., editors, *Machine Learning: Proceedings of the Eighth International Workshop*, pages 343–347. Morgan Kaufmann.

Ring, M. B. (1993). Learning sequential tasks by incrementally adding higher orders. In S. J. Hanson, J. D. C. and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pages 115–122. Morgan Kaufmann.

Ring, M. B. (1994). *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, Austin, Texas 78712.

Rissanen, J. (1986). Stochastic complexity and modeling. *The Annals of Statistics*, 14(3):1080–1100.

Ritter, H. and Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, 61(4):241–254.

Robinson, A. J. and Fallside, F. (1987). The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department.

Robinson, T. and Fallside, F. (1989). Dynamic reinforcement driven error propagation networks with application to game playing. In *Proceedings of the 11th Conference of the Cognitive Science Society, Ann Arbor*, pages 836–843.

Rodriguez, P. and Wiles, J. (1998). Recurrent neural networks can learn to implement symbol-sensitive counting. In *Advances in Neural Information Processing Systems*, volume 10, pages 87–93. The MIT Press.

Rodriguez, P., Wiles, J., and Elman, J. (1999). A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40.

Rohwer, R. (1989). The 'moving targets' training method. In Kindermann, J. and Linden, A., editors, *Proceedings of 'Distributed Adaptive Neural Information Processing', St.Augustin, 24.-25.5,*. Oldenbourg.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan, New York.

Roux, L., Racoceanu, D., Lomenie, N., Kulikova, M., Irshad, H., Klossa, J., Capron, F., Genestie, C., Naour, G. L., and Gurcan, M. N. (2013). Mitosis detection in breast cancer histological images - an ICPR 2012 contest. *J. Pathol. Inform.*, 4:8.

Rubner, J. and Schulten, K. (1990). Development of feature detectors by self-organization: A network model. *Biological Cybernetics*, 62:193–199.

Rubner, J. and Tavan, P. (1989). A self-organization network for principal-component analysis. *Europhysics Letters*, 10:693–698.

Rückstieß, T., Felder, M., and Schmidhuber, J. (2008). State-Dependent Exploration for policy gradient methods. In et al., W. D., editor, *European Conference on Machine Learning (ECML) and Principles and Practice of Knowledge Discovery in Databases 2008, Part II, LNAI 5212*, pages 234–249.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press.

Rummery, G. and Niranjan, M. (1994). On-line Q-learning using connectionist sytems. Technical Report CUED/F-INFENG-TR 166, Cambridge University, UK.

Russell, S. and Norvig, P. (1994). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ.

Saito, K. and Nakano, R. (1997). Partial BFGS update and efficient step-length calculation for three-layer neural networks. *Neural Computation*, 9(1):123–141.

Sałustowicz, R. P. and Schmidhuber, J. (1997). Probabilistic incremental program evolution. *Evolutionary Computation*, 5(2):123–141.

Samejima, K., Doya, K., and Kawato, M. (2003). Inter-module credit assignment in modular reinforcement learning. *Neural Networks*, 16(7):985–994.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, 3:210–229.

Sanger, T. D. (1989). An optimality principle for unsupervised learning. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1*, pages 11–19. Morgan Kaufmann.

Santamaría, J. C., Sutton, R. S., and Ram, A. (1997). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, 6(2):163–217.

Saravanan, N. and Fogel, D. B. (1995). Evolving neural control systems. *IEEE Expert*, pages 23–27.

Saund, E. (1994). Unsupervised learning of mixtures of multiple causes in binary data. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 27–34. Morgan Kaufmann.

Schäfer, A. M., Udluft, S., and Zimmermann, H.-G. (2006). Learning long term dependencies with recurrent neural networks. In Kollias, S. D., Stafylopatis, A., Duch, W., and Oja, E., editors, *ICANN (1)*, volume 4131 of *Lecture Notes in Computer Science*, pages 71–80. Springer.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5:197–227.

Schaul, T. and Schmidhuber, J. (2010). Metalearning. *Scholarpedia*, 6(5):4650.

Schaul, T., Zhang, S., and LeCun, Y. (2013). No more pesky learning rates. In *Proc. 30th International Conference on Machine Learning (ICML)*.

Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Proc. International Conference on Artificial Neural Networks (ICANN)*, pages 92–101.

Schmidhuber, J. (1987). Evolutionary principles in self-referential learning. Diploma thesis, Institut für Informatik, Technische Universität München. http://www.idsia.ch/˜juergen/diploma.html.

Schmidhuber, J. (1989a). Accelerated learning in back-propagation nets. In Pfeifer, R., Schreter, Z., Fogelman, Z., and Steels, L., editors, *Connectionism in Perspective*, pages 429 – 438. Amsterdam: Elsevier, North-Holland.

Schmidhuber, J. (1989b). A local learning algorithm for dynamic feedforward and recurrent networks. *Connection Science*, 1(4):403–412.

Schmidhuber, J. (1990a). Dynamische neuronale Netze und das fundamentale raumzeitliche Lernproblem. Dissertation, Institut für Informatik, Technische Universität München.

Schmidhuber, J. (1990b). Learning algorithms for networks with internal and external feedback. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E., editors, *Proc. of the 1990 Connectionist Models Summer School*, pages 52–61. Morgan Kaufmann.

Schmidhuber, J. (1990c). The Neural Heat Exchanger. Talks at TU Munich (1990), University of Colorado at Boulder (1992), and Z. Li's NIPS*94 workshop on unsupervised learning. Also published at the *Intl. Conference on Neural Information Processing* (ICONIP'96), vol. 1, pages 194-197, 1996.

Schmidhuber, J. (1990d). An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, San Diego*, volume 2, pages 253–258.

Schmidhuber, J. (1991a). Curious model-building control systems. In *Proceedings of the International Joint Conference on Neural Networks, Singapore*, volume 2, pages 1458–1463. IEEE press.

Schmidhuber, J. (1991b). Learning to generate sub-goals for action sequences. In Kohonen, T., Mäkisara, K., Simula, O., and Kangas, J., editors, *Artificial Neural Networks*, pages 967–972. Elsevier Science Publishers B.V., North-Holland.

Schmidhuber, J. (1991c). Reinforcement learning in Markovian and non-Markovian environments. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3 (NIPS 3)*, pages 500–506. Morgan Kaufmann.

Schmidhuber, J. (1992a). A fixed size storage $O(n^3)$ time complexity learning algorithm for fully recurrent continually running networks. *Neural Computation*, 4(2):243–248.

Schmidhuber, J. (1992b). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242. (Based on TR FKI-148-91, TUM, 1991).

Schmidhuber, J. (1992c). Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879.

Schmidhuber, J. (1993a). An introspective network that can learn to run its own weight change algorithm. In *Proc. of the Intl. Conf. on Artificial Neural Networks, Brighton*, pages 191–195. IEE.

Schmidhuber, J. (1993b). Netzwerkarchitekturen, Zielfunktionen und Kettenregel. (Network Architectures, Objective Functions, and Chain Rule.) Habilitationsschrift (Habilitation Thesis), Institut für Informatik, Technische Universität München.

Schmidhuber, J. (1995). Discovering solutions with low Kolmogorov complexity and high generalization capability. In Prieditis, A. and Russell, S., editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 488–496. Morgan Kaufmann Publishers, San Francisco, CA.

Schmidhuber, J. (1997). Discovering neural nets with low Kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873.

Schmidhuber, J. (2002). The Speed Prior: a new simplicity measure yielding near-optimal computable predictions. In Kivinen, J. and Sloan, R. H., editors, *Proceedings of the 15th Annual Conference on Computational Learning Theory (COLT 2002)*, Lecture Notes in Artificial Intelligence, pages 216–228. Springer, Sydney, Australia.

Schmidhuber, J. (2004). Optimal ordered problem solver. *Machine Learning*, 54:211–254.

Schmidhuber, J. (2006a). Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187.

Schmidhuber, J. (2006b). Gödel machines: Fully self-referential optimal universal self-improvers. In Goertzel, B. and Pennachin, C., editors, *Artificial General Intelligence*, pages 199–226. Springer Verlag. Variant available as arXiv:cs.LO/0309048.

Schmidhuber, J. (2007). Prototype resilient, self-modeling robots. *Science*, 316(5825):688.

Schmidhuber, J. (2012). Self-delimiting neural networks. Technical Report IDSIA-08-12, arXiv:1210.0118v1 [cs.NE], The Swiss AI Lab IDSIA.

Schmidhuber, J. (2013a). My first Deep Learning system of 1991 + Deep Learning timeline 1962-2013. Technical Report arXiv:1312.5548v1 [cs.NE], The Swiss AI Lab IDSIA.

Schmidhuber, J. (2013b). POWERPLAY: Training an Increasingly General Problem Solver by Continually Searching for the Simplest Still Unsolvable Problem. *Frontiers in Psychology*.

Schmidhuber, J., Ciresan, D., Meier, U., Masci, J., and Graves, A. (2011). On fast deep nets for AGI vision. In *Proc. Fourth Conference on Artificial General Intelligence (AGI), Google, Mountain View, CA*, pages 243–246.

Schmidhuber, J., Eldracher, M., and Foltin, B. (1996). Semilinear predictability minimization produces well-known feature detectors. *Neural Computation*, 8(4):773–786.

Schmidhuber, J. and Hochreiter, S. (1996). Guessing can outperform many long time lag algorithms. Technical Report IDSIA-19-96, The Swiss AI Lab IDSIA.

Schmidhuber, J., Hochreiter, S., and Bengio, Y. (2001). Evaluating benchmark problems by random guessing. In Kremer, S. C. and Kolen, J. F., editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

Schmidhuber, J. and Huber, R. (1991). Learning to generate artificial fovea trajectories for target detection. *International Journal of Neural Systems*, 2(1 & 2):135–141.

Schmidhuber, J., Mozer, M. C., and Prelinger, D. (1993). Continuous history compression. In Hüning, H., Neuhauser, S., Raus, M., and Ritschel, W., editors, *Proc. of Intl. Workshop on Neural Networks, RWTH Aachen*, pages 87–95. Augustinus.

Schmidhuber, J. and Prelinger, D. (1993). Discovering predictable classifications. *Neural Computation*, 5(4):625–635.

Schmidhuber, J. and Wahnsiedler, R. (1992). Planning simple trajectories using neural subgoal generators. In Meyer, J. A., Roitblat, H. L., and Wilson, S. W., editors, *Proc. of the 2nd International Conference on Simulation of Adaptive Behavior*, pages 196–202. MIT Press.

Schmidhuber, J., Wierstra, D., Gagliolo, M., and Gomez, F. J. (2007). Training recurrent networks by Evolino. *Neural Computation*, 19(3):757–779.

Schmidhuber, J., Zhao, J., and Schraudolph, N. (1997a). Reinforcement learning with self-modifying policies. In Thrun, S. and Pratt, L., editors, *Learning to learn*, pages 293–309. Kluwer.

Schmidhuber, J., Zhao, J., and Wiering, M. (1997b). Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. *Machine Learning*, 28:105–130.

Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors (1998). *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA.

Schraudolph, N. and Sejnowski, T. J. (1993). Unsupervised discrimination of clustered data via optimization of binary information gain. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems*, volume 5, pages 499–506. Morgan Kaufmann, San Mateo.

Schraudolph, N. N. (2002). Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738.

Schraudolph, N. N. and Sejnowski, T. J. (1996). Tempering backpropagation networks: Not all weights are created equal. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 8, pages 563–569. The MIT Press, Cambridge, MA.

Schuster, H. G. (1992). Learning by maximization the information transfer through nonlinear noisy neurons and "noise breakdown". *Phys. Rev.* **A**, 46(4):2131–2138.

Schuster, M. (1999). *On supervised learning from sequential data with applications for speech recognition*. PhD thesis, Nara Institute of Science and Technolog, Kyoto, Japan.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45:2673–2681.

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proc. ICML*, pages 298–305.

Schwefel, H. P. (1974). Numerische Optimierung von Computer-Modellen. Dissertation. Published 1977 by Birkhäuser, Basel.

Segmentation of Neuronal Structures in EM Stacks Challenge (2012). IEEE International Symposium on Biomedical Imaging (ISBI), http://tinyurl.com/d2fgh7g.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. (2010). Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559.

Sermanet, P. and LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN'11)*, pages 2809–2813.

Shanno, D. F. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656.

Shannon, C. E. (1948). A mathematical theory of communication (parts I and II). *Bell System Technical Journal*, XXVII:379–423.

Siegelmann, H. (1992). *Theoretical Foundations of Recurrent Neural Networks*. PhD thesis, Rutgers, New Brunswick Rutgers, The State of New Jersey.

Siegelmann, H. T. and Sontag, E. D. (1991). Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80.

Silva, F. M. and Almeida, L. B. (1990). Speeding up back-propagation. In Eckmiller, R., editor, *Advanced Neural Computers*, pages 151–158, Amsterdam. Elsevier.

Simard, P., Steinkraus, D., and Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition*, pages 958–963.

Sims, K. (1994). Evolving virtual creatures. In Glassner, A., editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 1994)*, Computer Graphics Proceedings, Annual Conference, pages 15–22. ACM SIGGRAPH, ACM Press. ISBN 0-89791-667-0.

Simsek, Ö. and Barto, A. G. (2008). Skill characterization based on betweenness. In *NIPS'08*, pages 1497–1504.

Singh, S., Barto, A. G., and Chentanez, N. (2005). Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 17 (NIPS)*. MIT Press, Cambridge, MA.

Singh, S. P. (1994). Reinforcement learning algorithms for average-payoff Markovian decision processes. In *National Conference on Artificial Intelligence*, pages 700–705.

Smith, S. F. (1980). *A Learning System Based on Genetic Adaptive Algorithms,*. PhD thesis, Univ. Pittsburgh.

Smolensky, P. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Information Processing in Dynamical Systems: Foundations of Harmony Theory, pages 194–281. MIT Press, Cambridge, MA, USA.

Solla, S. A. (1988). Accelerated learning in layered neural networks. *Complex Systems*, 2:625–640.

Solomonoff, R. J. (1964). A formal theory of inductive inference. Part I. *Information and Control*, 7:1–22.

Speelpenning, B. (1980). *Compiling Fast Partial Derivatives of Functions Given by Algorithms*. PhD thesis, Department of Computer Science, University of Illinois, Urbana-Champaign.

Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., and Schmidhuber, J. (2013). Compete to compute. In *Advances in Neural Information Processing Systems*, pages 2310–2318.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2011). INI German Traffic Sign Recognition Benchmark for the IJCNN'11 Competition.

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127.

Steijvers, M. and Grunwald, P. (1996). A recurrent network that performs a contextsensitive prediction task. In *Proceedings of the 18th Annual Conference of the Cognitive Science Society*. Erlbaum.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Roy. Stat. Soc.*, 36:111–147.

Sun, G., Chen, H., and Lee, Y. (1993a). Time warping invariant neural networks. In S. J. Hanson, J. D. C. and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pages 180–187. Morgan Kaufmann.

Sun, G. Z., Giles, C. L., Chen, H. H., and Lee, Y. C. (1993b). The neural network pushdown automaton: Model, stack and learning simulations. Technical Report CS-TR-3118, University of Maryland, College Park.

Sun, Y., Gomez, F., Schaul, T., and Schmidhuber, J. (2013). A Linear Time Natural Evolution Strategy for Non-Separable Functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 61, Amsterdam, NL. ACM.

Sun, Y., Wierstra, D., Schaul, T., and Schmidhuber, J. (2009). Efficient natural evolution strategies. In *Proc. 11th Genetic and Evolutionary Computation Conference (GECCO)*, pages 539–546.

Sutton, R. and Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA, MIT Press.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (1999a). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12 (NIPS)*, pages 1057–1063.

Sutton, R. S., Precup, D., and Singh, S. P. (1999b). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211.

Sutton, R. S., Szepesvári, C., and Maei, H. R. (2008). A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation. In *Advances in Neural Information Processing Systems (NIPS'08)*, volume 21, pages 1609–1616.

Szegedy, C., Toshev, A., and Erhan, D. (2013). Deep neural networks for object detection. pages 2553–2561.

Teller, A. (1994). The evolution of mental models. In Kenneth E. Kinnear, J., editor, *Advances in Genetic Programming*, pages 199–219. MIT Press.

Tenenberg, J., Karlsson, J., and Whitehead, S. (1993). Learning via task decomposition. In Meyer, J. A., Roitblat, H., and Wilson, S., editors, *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 337–343. MIT Press.

Tesauro, G. (1994). TD-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Tiňo, P. and Hammer, B. (2004). Architectural bias in recurrent neural networks: Fractal analysis. *Neural Computation*, 15(8):1931–1957.

Tonkes, B. and Wiles, J. (1997). Learning a context-free task with a recurrent neural network: An analysis of stability. In *Proceedings of the Fourth Biennial Conference of the Australasian Cognitive Science Society*.

Tsitsiklis, J. N. and van Roy, B. (1996). Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1-3):59–94.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 41:230–267.

Ueda, N. (2000). Optimal linear combination of neural networks for improving classification performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(2):207–215.

Urlbe, A. P. (1999). *Structure-adaptable digital neural networks*. PhD thesis, Universidad del Valle.

Vahed, A. and Omlin, C. W. (2004). A machine learning method for extracting symbolic knowledge from recurrent neural networks. *Neural Computation*, 16(1):59–71.

Vapnik, V. (1992). Principles of risk minimization for learning theory. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 4*, pages 831–838. Morgan Kaufmann.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.

Veta, M., Viergever, M., Pluim, J., Stathonikos, N., and van Diest, P. J. (2013). MICCAI 2013 Grand Challenge on Mitosis Detection.

Vincent, P., Hugo, L., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1096–1103, New York, NY, USA. ACM.

Vogl, T., Mangis, J., Rigler, A., Zink, W., and Alkon, D. (1988). Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, 59:257–263.

von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14(2):85–100.

Wallace, C. S. and Boulton, D. M. (1968). An information theoretic measure for classification. *Computer Journal*, 11(2):185–194.

Wan, E. A. (1994). Time series prediction by using a connectionist network with internal delay lines. In Weigend, A. S. and Gershenfeld, N. A., editors, *Time series prediction: Forecasting the future and understanding the past*, pages 265–295. Addison-Wesley.

Wang, C., Venkatesh, S. S., and Judd, J. S. (1994). Optimal stopping and effective machine complexity in learning. In *Advances in Neural Information Processing Systems (NIPS'6)*, pages 303–310. Morgan Kaufmann.

Wang, S. and Manning, C. (2013). Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 118–126.

Watanabe, O. (1992). *Kolmogorov complexity and computational complexity*. EATCS Monographs on Theoretical Computer Science, Springer.

Watanabe, S. (1985). *Pattern Recognition: Human and Mechanical*. Willey, New York.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Oxford.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.

Weigend, A. S. and Gershenfeld, N. A. (1993). Results of the time series prediction competition at the santa fe institute. In *Neural Networks, 1993., IEEE International Conference on*, pages 1786–1793. IEEE.

Weigend, A. S., Rumelhart, D. E., and Huberman, B. A. (1991). Generalization by weight-elimination with application to forecasting. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 875–882. San Mateo, CA: Morgan Kaufmann.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University.

Werbos, P. J. (1981). Applications of advances in nonlinear sensitivity analysis. In *Proceedings of the 10th IFIP Conference, 31.8 - 4.9, NYC*, pages 762–770.

Werbos, P. J. (1987). Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 17.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1.

Werbos, P. J. (1989a). Backpropagation and neurocontrol: A review and prospectus. In *IEEE/INNS International Joint Conference on Neural Networks, Washington, D.C.*, volume 1, pages 209–216.

Werbos, P. J. (1989b). Neural networks for control and system identification. In *Proceedings of IEEE/CDC Tampa, Florida*.

Werbos, P. J. (1992). Neural networks, system identification, and control in the chemical industries. In D. A. White, D. A. S., editor, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, pages 283–356. Thomson Learning.

Werbos, P. J. (2006). Backwards differentiation in AD and neural nets: Past links and new opportunities. In *Automatic Differentiation: Applications, Theory, and Implementations*, pages 15–34. Springer.

West, A. H. L. and Saad, D. (1995). Adaptive back-propagation in on-line learning of multilayer networks. In Touretzky, D. S., Mozer, M., and Hasselmo, M. E., editors, *NIPS*, pages 323–329. MIT Press.

White, H. (1989). Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1(4):425–464.

Whiteson, S., Kohl, N., Miikkulainen, R., and Stone, P. (2005). Evolving keepaway soccer players through task decomposition. *Machine Learning*, 59(1):5–30.

Widrow, B. and Hoff, M. (1962). Associative storage and retrieval of digital information in networks of adaptive neurons. *Biological Prototypes and Synthetic Systems*, 1:160.

Widrow, B., Rumelhart, D. E., and Lehr, M. A. (1994). Neural networks: Applications in industry, business and science. *Commun. ACM*, 37(3):93–105.

Wiering, M. and Schmidhuber, J. (1996). Solving POMDPs with Levin search and EIRA. In Saitta, L., editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 534–542. Morgan Kaufmann Publishers, San Francisco, CA.

Wiering, M. and Schmidhuber, J. (1998a). HQ-learning. *Adaptive Behavior*, 6(2):219–246.

Wiering, M. A. and Schmidhuber, J. (1998b). Fast online Q(λ). *Machine Learning*, 33(1):105–116.

Wierstra, D., Foerster, A., Peters, J., and Schmidhuber, J. (2007). Solving deep memory POMDPs with recurrent policy gradients. In *ICANN (1)*, volume 4668 of *Lecture Notes in Computer Science*, pages 697–706. Springer.

Wierstra, D., Foerster, A., Peters, J., and Schmidhuber, J. (2010). Recurrent policy gradients. *Logic Journal of IGPL*, 18(2):620–634.

Wierstra, D., Schaul, T., Peters, J., and Schmidhuber, J. (2008). Natural evolution strategies. In *Congress of Evolutionary Computation (CEC 2008)*.

Wiesel, D. H. and Hubel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.*, 148:574–591.

Wiles, J. and Elman, J. (1995). Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. In *In Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pages pages 482 – 487, Cambridge, MA. MIT Press.

Wilkinson, J. H., editor (1965). *The Algebraic Eigenvalue Problem*. Oxford University Press, Inc., New York, NY, USA.

Williams, R. J. (1986). Reinforcement-learning in connectionist networks: A mathematical analysis. Technical Report 8605, Institute for Cognitive Science, University of California, San Diego.

Williams, R. J. (1988). Toward a theory of reinforcement-learning connectionist systems. Technical Report NU-CCS-88-3, College of Comp. Sci., Northeastern University, Boston, MA.

Williams, R. J. (1989). Complexity of exact gradient computation algorithms for recurrent neural networks. Technical Report Technical Report NU-CCS-89-27, Boston: Northeastern University, College of Computer Science.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Williams, R. J. and Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 4:491–501.

Williams, R. J. and Zipser, D. (1988). A learning algorithm for continually running fully recurrent networks. Technical Report ICS Report 8805, Univ. of California, San Diego, La Jolla.

Williams, R. J. and Zipser, D. (1989a). Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, 1(1):87–111.

Williams, R. J. and Zipser, D. (1989b). A learning algorithm for continually running fully recurrent networks. *Neural Computation*, 1(2):270–280.

Willshaw, D. J. and von der Malsburg, C. (1976). How patterned neural connections can be set up by self-organization. *Proc. R. Soc. London B*, 194:431–445.

Wiskott, L. and Sejnowski, T. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

Wolpert, D. H. (1994). Bayesian backpropagation over i-o functions rather than weights. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 200–207. Morgan Kaufmann.

Yamauchi, B. M. and Beer, R. D. (1994). Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2(3):219–246.

Yamins, D., Hong, H., Cadieu, C., and DiCarlo, J. J. (2013). Hierarchical Modular Optimization of Convolutional Networks Achieves Representations Similar to Macaque IT and Human Ventral Stream. *Advances in Neural Information Processing Systems NIPS*, pages 1–9.

Yao, X. (1993). A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 4:203–222.

Yu, X.-H., Chen, G.-A., and Cheng, S.-X. (1995). Dynamic learning rate optimization of the backpropagation algorithm. *IEEE Transactions on Neural Networks*, 6(3):669–677.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701.

Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. Technical Report arXiv:1311.2901 [cs.CV], NYU.

Zemel, R. S. (1993). *A minimum description length framework for unsupervised learning*. PhD thesis, University of Toronto.

Zemel, R. S. and Hinton, G. E. (1994). Developing population codes by minimizing description length. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 11–18. Morgan Kaufmann.

Zeng, Z., Goodman, R., and Smyth, P. (1994). Discrete recurrent neural networks for grammatical inference. *IEEE Transactions on Neural Networks*, 5(2).

Zimmermann, H.-G., Tietz, C., and Grothmann, R. (2012). Forecasting with recurrent neural networks: 12 tricks. In Montavon, G., Orr, G. B., and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pages 687–707. Springer.