Neural Networks 61 (2015) 85–117

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Review

# Deep learning in neural networks: An overview

Jürgen Schmidhuber

*The Swiss AI Lab IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, University of Lugano & SUPSI, Galleria 2, 6928 Manno-Lugano, Switzerland*

ARTICLE INFO

*Article history:*
Received 2 May 2014
Received in revised form 12 September 2014
Accepted 14 September 2014
Available online 13 October 2014

*Keywords:*
Deep learning
Supervised learning
Unsupervised learning
Reinforcement learning
Evolutionary computation

ABSTRACT

In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarizes 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.



**Contents**



*E-mail address:* juergen@idsia.ch.

http://dx.doi.org/10.1016/j.neunet.2014.09.003
0893-6080/

## Preface

This is the preprint 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 preprint mostly consists of references. 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, this work 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.

## 1. Introduction to Deep Learning (DL) in Neural Networks (NNs)

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 (Section 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).

A standard neural network (NN) consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations. Input neurons get activated through sensors perceiving the environment, other neurons get activated through weighted connections from previously active neurons (details in Section 2). Some neurons may influence the environment by triggering actions. *Learning* or *credit assignment* is about finding weights that make the NN exhibit *desired* behavior, such as driving a car. Depending on the problem and how the neurons are connected, such behavior may require long causal chains of computational stages (Section 3), where each stage transforms (often in a non-linear way) the aggregate activation of the network. Deep Learning is about accurately assigning credit across *many* such stages.

*Shallow* NN-like models with *few* such stages have been around for many decades if not centuries (Section 5.1). Models with several successive nonlinear layers of neurons date back at least to the 1960s (Section 5.3) and 1970s (Section 5.5). 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 (Section 5.5). BP-based training of *deep* NNs with *many* layers, however, had been found to be difficult in practice by the late 1980s (Section 5.6), and had become an explicit research subject by the early 1990s (Section 5.9). DL became practically feasible to some extent through the help of *Unsupervised Learning* (UL), e.g., Section 5.10 (1991), Section 5.15 (2006). The 1990s and 2000s also saw many improvements of purely supervised DL (Section 5). In the new millennium, deep NNs have finally attracted wide-spread attention, mainly by outperforming alternative machine learning methods such as kernel machines (Schölkopf, Burges, & Smola, 1998; Vapnik, 1995) in numerous important applications. In fact, since 2009, supervised deep NNs have won many official international pattern recognition competitions (e.g., Sections 5.17, 5.19, 5.21 and 5.22), achieving the first superhuman visual pattern recognition results in limited domains (Section 5.19, 2011). Deep NNs also have become relevant for the more general field of *Reinforcement Learning* (RL) where there is no supervising teacher (Section 6).

Both *feedforward* (acyclic) NNs (FNNs) and *recurrent* (cyclic) NNs (RNNs) have won contests (Sections 5.12, 5.14, 5.17, 5.19, 5.21, 5.22). In a sense, RNNs are the deepest of all NNs (Section 3)—they are general computers more powerful than FNNs, and can in principle create and process memories of arbitrary sequences of input patterns (e.g., Schmidhuber, 1990a; Siegelmann & Sontag, 1991). Unlike traditional methods for automatic sequential program synthesis (e.g., Balzer, 1985; Deville & Lau, 1994; Soloway,

**Abbreviations in alphabetical order**

AE: Autoencoder
AI: Artificial Intelligence
ANN: Artificial Neural Network
BFGS: Broyden–Fletcher–Goldfarb–Shanno
BNN: Biological Neural Network
BM: Boltzmann Machine
BP: Backpropagation
BRNN: Bi-directional Recurrent Neural Network
CAP: Credit Assignment Path
CEC: Constant Error Carousel
CFL: Context Free Language
CMA-ES: Covariance Matrix Estimation ES
CNN: Convolutional Neural Network
CoSyNE: Co-Synaptic Neuro-Evolution
CSL: Context Sensitive Language
CTC: Connectionist Temporal Classification
DBN: Deep Belief Network
DCT: Discrete Cosine Transform
DL: Deep Learning
DP: Dynamic Programming
DS: Direct Policy Search
EA: Evolutionary Algorithm
EM: Expectation Maximization
ES: Evolution Strategy
FMS: Flat Minimum Search
FNN: Feedforward Neural Network
FSA: Finite State Automaton
GMDH: Group Method of Data Handling
GOFAI: Good Old-Fashioned AI
GP: Genetic Programming
GPU: Graphics Processing Unit
GPU-MPCNN: GPU-Based MPCNN
HMM: Hidden Markov Model
HRL: Hierarchical Reinforcement Learning
HTM: Hierarchical Temporal Memory
HMAX: Hierarchical Model "and X"
LSTM: Long Short-Term Memory (RNN)
MDL: Minimum Description Length
MDP: Markov Decision Process
MNIST: Mixed National Institute of Standards and Technology Database
MP: Max-Pooling
MPCNN: Max-Pooling CNN
NE: NeuroEvolution
NEAT: NE of Augmenting Topologies
NES: Natural Evolution Strategies
NFQ: Neural Fitted Q-Learning
NN: Neural Network
OCR: Optical Character Recognition
PCC: Potential Causal Connection
PDCC: Potential Direct Causal Connection
PM: Predictability Minimization
POMDP: Partially Observable MDP
RAAM: Recursive Auto-Associative Memory
RBM: Restricted Boltzmann Machine
ReLU: Rectified Linear Unit
RL: Reinforcement Learning
RNN: Recurrent Neural Network
R-prop: Resilient Backpropagation
SL: Supervised Learning
SLIM NN: Self-Delimiting Neural Network
SOTA: Self-Organizing Tree Algorithm
SVM: Support Vector Machine
TDNN: Time-Delay Neural Network
TIMIT: TI/SRI/MIT Acoustic-Phonetic Continuous Speech Corpus
UL: Unsupervised Learning
WTA: Winner-Take-All

1986; Waldinger & Lee, 1969), RNNs can learn programs that mix sequential and parallel information processing in a natural and efficient way, exploiting the massive parallelism viewed as crucial for sustaining the rapid decline of computation cost observed over the past 75 years.

The rest of this paper is structured as follows. Section 2 introduces a compact, event-oriented notation that is simple yet general enough to accommodate both FNNs and RNNs. Section 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. Section 4 lists recurring themes of DL in SL, UL, and RL. Section 5 focuses on SL and UL, and on how UL can facilitate SL, although pure SL has become dominant in recent competitions (Sections 5.17–5.23). Section 5 is arranged in a historical timeline format with subsections on important inspirations and technical contributions. Section 6 on deep RL discusses traditional *Dynamic Programming* (DP)-based RL combined with gradient-based search techniques for SL or UL in deep NNs, as well as 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 NNs

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., Sections 5.3, 5.6.3). 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. In sequence-processing, fully connected RNNs, all units have connections to all non-input units.

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 non-linear real-valued *activation function* such as tanh. In many recent competition-winning NNs (Sections 5.19, 5.21, 5.22) there also are events of the type $x_t = \max_{k \in in_t}(x_k)$; some network types may also use complex polynomial activation functions (Section 5.3). $x_t$ 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 of the 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 (Sections 5.4 *and* 5.8). 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 (Section 4.4).

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 assumption does not hold in the broader fields of *Sequential Decision Making* and *Reinforcement Learning* (RL) (Hutter, 2005; Kaelbling, Littman, & Moore, 1996; Sutton & Barto, 1998; Wiering & van Otterlo, 2012) (Section 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.

Section 5.5 will use the notation above to compactly describe a central algorithm of DL, namely, backpropagation (BP) for supervised weight-sharing FNNs and RNNs. (FNNs may be viewed as RNNs with certain fixed zero weights.) Section 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 the events of Section 2, e.g., from input through hidden to output layers in FNNs, or through transformations over time in RNNs.

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 defined to be a CAP (a minimal one) 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 called the *solution depth*. There may be other solutions (yielding different event sequences) with different depths. Given some fixed NN topology, the smallest depth of any solution is called the *problem depth*.

Sometimes we also speak of the *depth of an architecture*: SL FNNs with fixed topology imply a problem-independent maximal problem depth bounded by the number of non-input layers. Certain SL RNNs with fixed weights for all connections except those to output units (Jaeger, 2001, 2004; Maass, Natschläger, & Markram, 2002; Schrauwen, Verstraeten, & Van Campenhout, 2007) have a maximal problem depth of 1, because only the final links in the corresponding CAPs are modifiable. In general, however, RNNs may learn to solve problems of potentially unlimited depth.

Note that the definitions above are solely based on the depths of causal chains, and agnostic to 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.

At which problem depth does *Shallow Learning* end, and *Deep Learning* begin? Discussions with DL experts have not yet yielded a conclusive response to this question. Instead of committing myself to a precise answer, let me just define for the purposes of this overview: problems of depth $>10$ require *Very Deep Learning*.

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 (Section 5.9) or other types of direct search (Section 6.6) or indirect search (Section 6.7) in weight space, or through training an NN first on shallow problems whose solutions may then generalize to deep problems, or through collapsing sequences of (non)linear operations into a single (non)linear operation (but see an analysis of non-trivial aspects of deep linear networks, Baldi & Hornik, 1995, Section B). In general, however, finding an NN that precisely models a given training set is an NP-complete problem (Blum & Rivest, 1992; Judd, 1990), also in the case of deep NNs (de Souto, Souto, & Oliveira, 1999; Síma, 1994; Windisch, 2005); compare a survey of negative results (Síma, 2002, Section 1).

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 (including reward signals) from former input events and actions (Section 6.1). Its weights are frozen, but can help to assign credit to other, still modifiable weights used to compute actions (Section 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 (Section 4). In particular, sometimes UL is used to make SL problems less deep, e.g., Section 5.10. Often *Dynamic Programming* (Section 4.1) is used to facilitate certain traditional RL problems, e.g., Section 6.2. Section 5 focuses on CAPs for SL, Section 6 on the more complex case of RL.

## 4. Recurring themes of Deep Learning

### 4.1. Dynamic programming for Supervised/Reinforcement Learning (SL/RL)

One recurring theme of DL is *Dynamic Programming* (DP) (Bellman, 1957), which can help to facilitate credit assignment under

[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., Section 6.1.

certain assumptions. For example, in SL NNs, backpropagation itself can be viewed as a DP-derived method (Section 5.5). In traditional RL based on strong Markovian assumptions, DP-derived methods can help to greatly reduce problem depth (Section 6.2). DP algorithms are also essential for systems that combine concepts of NNs and graphical models, such as *Hidden Markov Models* (HMMs) (Baum & Petrie, 1966; Stratonovich, 1960) and *Expectation Maximization* (EM) (Dempster, Laird, & Rubin, 1977; Friedman, Hastie, & Tibshirani, 2001), e.g., Baldi and Chauvin (1996), Bengio (1991), Bishop (2006), Bottou (1991), Bourlard and Morgan (1994), Dahl, Yu, Deng, and Acero (2012), Hastie, Tibshirani, and Friedman (2009), Hinton, Deng, et al. (2012), Jordan and Sejnowski (2001), Poon and Domingos (2011) and Wu and Shao (2014).

### 4.2. Unsupervised Learning (UL) facilitating SL and RL

Another recurring theme is how UL can facilitate both SL (Section 5) and RL (Section 6). UL (Section 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 (Sections 5.10, 5.15) or RL machines (Section 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 (Sections 4.4, 5.6.3).

### 4.3. Learning hierarchical representations through deep SL, UL, RL

Many methods of *Good Old-Fashioned Artificial Intelligence* (GOFAI) (Nilsson, 1980) as well as more recent approaches to AI (Russell, Norvig, Canny, Malik, & Edwards, 1995) and *Machine Learning* (Mitchell, 1997) 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 & Brown, 1984) continually learns concepts by combining previously learnt concepts. Such hierarchical representation learning (Bengio, Courville, & Vincent, 2013; Deng & Yu, 2014; Ring, 1994) is also a recurring theme of DL NNs for SL (Section 5), UL-aided SL (Sections 5.7, 5.10, 5.15), and hierarchical RL (Section 6.5). Often, abstract hierarchical representations are natural by-products of data compression (Section 4.4), e.g., Section 5.10.

### 4.4. Occam's razor: compression and Minimum Description Length (MDL)

Occam's razor favors simple solutions over complex ones. Given some programming language, the principle of *Minimum 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., Blumer, Ehrenfeucht, Haussler, & Warmuth, 1987; Chaitin, 1966; Grünwald, Myung, & Pitt, 2005; Kolmogorov, 1965b; Levin, 1973a; Li & Vitányi, 1997; Rissanen, 1986; Solomonoff, 1964, 1978; Wallace & Boulton, 1968). Some methods explicitly take into account program runtime (Allender, 1992; Schmidhuber, 1997, 2002; Watanabe, 1992); many consider only programs with constant runtime, written in non-universal programming languages (e.g., Hinton & van Camp, 1993; Rissanen, 1986). In the NN case, the MDL principle suggests that low NN weight complexity corresponds to high NN probability in the Bayesian view (e.g., Buntine & Weigend, 1991; De Freitas, 2003; MacKay, 1992; Neal, 1995), and to high generalization performance (e.g., Baum & Haussler, 1989), without overfitting the training data. Many methods have been proposed for *regularizing* NNs, that is, searching for solution-computing but simple, low-complexity SL NNs (Section 5.6.3) and RL NNs (Section 6.7). This is closely related to certain UL methods (Sections 4.2, 5.6.4).

### 4.5. Fast Graphics Processing Units (GPUs) for DL in NNs

While the previous millennium saw several attempts at creating fast NN-specific hardware (e.g., Faggin, 1992; Heemskerk, 1995; Jackel et al., 1990; Korkin, de Garis, Gers, & Hemmi, 1997; Ramacher et al., 1993; Urlbe, 1999; Widrow, Rumelhart, & Lehr, 1994), and at exploiting standard hardware (e.g., Anguita & Gomes, 1996; Anguita, Parodi, & Zunino, 1994; Muller, Gunzinger, & Guggenbühl, 1995), the new millennium brought a DL breakthrough in form of cheap, multi-processor graphics cards or GPUs. GPUs are widely used for video games, a huge and competitive market that has driven down hardware prices. GPUs excel at the 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 (Sections 5.16–5.19) have greatly contributed to recent successes in contests for pattern recognition (Sections 5.19–5.22), image segmentation (Section 5.21), and object detection (Sections 5.21–5.22).

## 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 (Sections 5.17–5.23). Several methods, however, use additional *Unsupervised Learning* (UL) to facilitate SL (Sections 5.7, 5.10, 5.15). It does make sense to treat SL and UL in the same section: often gradient-based methods, such as BP (Section 5.5.1), 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., Sections 5.10, 5.12.

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). Section 5.1 briefly mentions early, shallow NN models since the 1940s (and 1800s), Section 5.2 additional early neurobiological inspiration relevant for modern Deep Learning (DL). Section 5.3 is about GMDH networks (since 1965), to my knowledge the first (feedforward) DL systems. Section 5.4 is about the relatively deep *Neocognitron* NN (1979) which is very similar to certain modern deep FNN architectures, as it combines convolutional NNs (CNNs), weight pattern replication, and subsampling mechanisms. Section 5.5 uses the notation of Section 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. Section 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. Section 5.7 discusses a first hierarchical stack (1987) of coupled UL-based Autoencoders (AEs)—this concept resurfaced in the new millennium (Section 5.15). Section 5.8 is about applying BP to CNNs (1989), which is important for today's DL applications. Section 5.9 explains BP's *Fundamental DL Problem* (of vanishing/exploding gradients) discovered in 1991. Section 5.10 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, Section 3) of depth 1000 and more. Section 5.11 discusses a particular *winner-take-all* (WTA) method called *Max-Pooling* (MP, 1992) widely used in today's deep FNNs. Section 5.12 mentions a first important contest won by SL NNs in 1994. Section 5.13 describes a purely supervised DL RNN (*Long Short-Term Memory*, LSTM, 1995) for problems of depth 1000 and more. Section 5.14 mentions an early contest of 2003 won by an ensemble of shallow FNNs, as well as good pattern recognition results with CNNs and deep FNNs and LSTM RNNs

(2003). Section 5.15 is mostly about *Deep Belief Networks* (DBNs, 2006) and related stacks of *Autoencoders* (AEs, Section 5.7), both pre-trained by UL to facilitate subsequent BP-based SL (compare Sections 5.6.1, 5.10). Section 5.16 mentions the first SL-based GPU-CNNs (2006), BP-trained MPCNNs (2007), and LSTM stacks (2007). Sections 5.17–5.22 focus on official competitions with secret test sets won by (mostly purely supervised) deep NNs since 2009, in sequence recognition, image classification, image segmentation, and object detection. Many RNN results depended on LSTM (Section 5.13); many FNN results depended on GPU-based FNN code developed since 2004 (Sections 5.16–5.19), in particular, GPU-MPCNNs (Section 5.19). Section 5.24 mentions recent tricks for improving DL in NNs, many of them closely related to earlier tricks from the previous millennium (e.g., Sections 5.6.2, 5.6.3). Section 5.25 discusses how artificial NNs can help to understand biological NNs; Section 5.26 addresses the possibility of DL in NNs with spiking neurons.

### 5.1. Early NNs since the 1940s (and the 1800s)

Early NN architectures (McCulloch & Pitts, 1943) did not learn. The first ideas about UL were published a few years later (Hebb, 1949). The following decades brought simple NNs trained by SL (e.g., Narendra & Thathatchar, 1974; Rosenblatt, 1958, 1962; Widrow & Hoff, 1962) and UL (e.g., Grossberg, 1969; Kohonen, 1972; von der Malsburg, 1973; Willshaw & von der Malsburg, 1976), as well as closely related associative memories (e.g., Hopfield, 1982; Palm, 1980).

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, 1809, 1821; Legendre, 1805); Gauss also refers to his work of 1795. Early NNs had a maximal CAP depth of 1 (Section 3).

### 5.2. Around 1960: visual cortex provides inspiration for DL (Sections 5.4, 5.11)

Simple cells and complex cells were found in the cat's visual cortex (e.g., Hubel & Wiesel, 1962; Wiesel & 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 (Sections 5.4, 5.11) used in certain modern award-winning Deep Learners (Sections 5.19–5.22).

### 5.3. 1965: deep networks based on the Group Method of Data Handling

Networks trained by the *Group Method of Data Handling* (GMDH) (Ivakhnenko, 1968, 1971; Ivakhnenko & Lapa, 1965; Ivakhnenko, Lapa, & McDonough, 1967) were perhaps the first DL systems of the *Feedforward Multilayer Perceptron* type, although there was earlier work on NNs with a single hidden layer (e.g., Joseph, 1961; Viglione, 1970). The units of GMDH nets may have polynomial activation functions implementing *Kolmogorov–Gabor polynomials* (more general than other widely used NN activation functions, Section 2). Given a training set, layers are incrementally grown and trained by regression analysis (e.g., Gauss, 1809, 1821; Legendre, 1805) (Section 5.1), then pruned with the help of a separate *validation set* (using today's terminology), where *Decision Regularization* is used to weed out superfluous units (compare Section 5.6.3). The numbers of layers and units per layer can be learned in problem-dependent fashion. To my knowledge, this was the first example of open-ended, hierarchical representation learning in NNs (Section 4.3). A paper of 1971 already described a deep GMDH network with 8 layers (Ivakhnenko, 1971). There have been numerous applications of GMDH-style nets, e.g. Farlow (1984), Ikeda, Ochiai, and Sawaragi (1976), Ivakhnenko (1995), Kondo (1998), Kondo and Ueno (2008), Kordík, Náplava, Snorek, and Genyk-Berezovskyj (2003), Madala and Ivakhnenko (1994) and Witczak, Korbicz, Mrugalski, and Patton (2006).

### 5.4. 1979: convolution + weight replication + subsampling (Neocognitron)

Apart from deep GMDH networks (Section 5.3), 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 Section 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 (a *filter*) is shifted step by step across a 2-dimensional array of input values, such as the pixels of an image (usually there are several such filters). 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* (Section 2), relatively few parameters (Section 4.4) may be necessary to describe the behavior of such a *convolutional layer*.

*Subsampling* or *downsampling* layers consist of units whose fixed-weight connections originate from physical neighbors in the convolutional layers below. Subsampling units become active if at least one of their inputs is active; their responses are insensitive to certain small image shifts (compare Section 5.2).

The Neocognitron is very similar to the architecture of modern, contest-winning, purely *supervised*, feedforward, gradient-based Deep Learners with alternating convolutional and downsampling layers (e.g., Sections 5.19–5.22). Fukushima, however, did not set the weights by supervised backpropagation (Sections 5.5, 5.8), but by local, WTA-based *un*supervised learning rules (e.g., Fukushima, 2013b), or by pre-wiring. In that sense he did not care for the DL problem (Section 5.9), although his architecture was comparatively deep indeed. For downsampling purposes he used *Spatial Averaging* (Fukushima, 1980, 2011) instead of *Max-Pooling* (MP, Section 5.11), currently a particularly convenient and popular WTA mechanism. Today's DL combinations of CNNs and MP and BP also profit a lot from later work (e.g., Sections 5.8, 5.16, 5.19).

### 5.5. 1960–1981 and beyond: development of backpropagation (BP) for NNs

The minimization of errors through *gradient descent* (Hadamard, 1908) in the parameter space of complex, nonlinear, differentiable (Leibniz, 1684), multi-stage, NN-related systems has been discussed at least since the early 1960s (e.g., Amari, 1967; Bryson, 1961; Bryson & Denham, 1961; Bryson & Ho, 1969; Director & Rohrer, 1969; Dreyfus, 1962; Kelley, 1960; Pontryagin, Boltyanskii, Gamrelidze, & Mishchenko, 1961; Wilkinson, 1965), initially within the framework of Euler–Lagrange equations in the *Calculus of Variations* (e.g., Euler, 1744).

*Steepest descent* in the weight space of such systems can be performed (Bryson, 1961; Bryson & Ho, 1969; Kelley, 1960) by iterating the chain rule (Leibniz, 1676; L'Hôpital, 1696) à la *Dynamic Programming* (DP) (Bellman, 1957). A simplified derivation of this backpropagation method 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, without explicitly addressing either direct links across several layers or potential additional efficiency gains due to network sparsity (but perhaps such enhancements seemed obvious to the authors). Given all the prior work on learning in multilayer NN-like systems (see also Section 5.3 on deep nonlinear nets since

1965), it seems surprising in hindsight that a book (Minsky & Papert, 1969) on the limitations of simple linear perceptrons with a single layer (Section 5.1) discouraged some researchers from further studying NNs.

Explicit, efficient error backpropagation (BP) in arbitrary, discrete, possibly sparsely connected, NN-like networks apparently was first described in a 1970 master's thesis (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) and closely related work (Ostrovskii, Volin, & Borisov, 1971).

Efficient BP was soon explicitly used to minimize cost functions by adapting control parameters (weights) (Dreyfus, 1973). Compare some preliminary, 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 as above was described in 1981 (Werbos, 1981, 2006). Related work was published several years later (LeCun, 1985, 1988; Parker, 1985). A paper of 1986 significantly contributed to the popularization of BP for NNs (Rumelhart, Hinton, & Williams, 1986), experimentally demonstrating the emergence of useful internal representations in hidden layers. See generalizations for sequence-processing recurrent NNs (e.g., Atiya & Parlos, 2000; Baldi, 1995; Gherrity, 1989; Kremer & Kolen, 2001; Pearlmutter, 1989, 1995; Robinson & Fallside, 1987; Rohwer, 1989; Schmidhuber, 1992a; Werbos, 1988; Williams, 1989; Williams & Peng, 1990; Williams & Zipser, 1988, 1989a, 1989b), also for equilibrium RNNs (Almeida, 1987; Pineda, 1987) with stationary inputs.

#### 5.5.1. BP for weight-sharing feedforward NNs (FNNs) and recurrent NNs (RNNs)

Using the notation of Section 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 5.5.1 (for the additive case), where each weight $w_i$ is associated with a real-valued variable $\triangle_i$ initialized by 0.

**Algorithm 5.5.1: One iteration of BP for weight-sharing FNNs or RNNs**

**for** $t = T, \ldots, 1$ **do**
  to compute $\frac{\partial E}{\partial net_t}$, initialize real-valued error signal variable $\delta_t$ 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 $\delta_t$ 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 $\delta_t$ 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**
change each $w_i$ in proportion to $\triangle_i$ and a small real-valued learning rate

The computational costs of the backward (BP) pass are essentially those of the forward pass (Section 2). Forward and backward passes are re-iterated until sufficient performance is reached.

As of 2014, this simple BP method is still the central learning algorithm for FNNs and RNNs. Notably, most contest-winning NNs up to 2014 (Sections 5.12, 5.14, 5.17, 5.19, 5.21, 5.22) did *not* augment supervised BP by some sort of *un*supervised learning as discussed in Sections 5.7, 5.10, 5.15.

### 5.6. Late 1980s–2000 and beyond: numerous improvements of NNs

By the late 1980s it seemed clear that BP by itself (Section 5.5) was no panacea. Most FNN applications focused on FNNs with few hidden layers. Additional hidden layers often did not seem to offer empirical benefits. Many practitioners found solace in a theorem (Hecht-Nielsen, 1989; Hornik, Stinchcombe, & White, 1989; Kolmogorov, 1965a) stating that an NN with a single layer of enough hidden units can approximate any multivariate continuous function with arbitrary accuracy.

Likewise, most RNN applications did not require backpropagating errors far. Many researchers helped their RNNs by first training them on shallow problems (Section 3) whose solutions then generalized to deeper problems. In fact, some popular RNN algorithms restricted credit assignment to a single step backwards (Elman, 1990; Jordan, 1986, 1997), also in more recent studies (Jaeger, 2001, 2004; Maass et al., 2002).

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 (Section 5.9).

#### 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* based on decay factors for activations of units in RNNs (Mozer, 1989, 1992), *Time-Delay Neural Networks* (TDNNs) (Lang, Waibel, & Hinton, 1990) and their adaptive extension (Bodenhausen & Waibel, 1991), *Nonlinear AutoRegressive with eXogenous inputs* (NARX) RNNs (Lin, Horne, Tino, & Giles, 1996), certain hierarchical RNNs (Hihi & Bengio, 1996) (compare Section 5.10, 1991), RL economies in RNNs with WTA units and local learning rules (Schmidhuber, 1989b), and other methods (e.g., Bengio, Simard, & Frasconi, 1994; de Vries & Principe, 1991; Plate, 1993; Ring, 1993, 1994; Sun, Chen, & Lee, 1993). 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 (Hochreiter & Schmidhuber, 1996).

While the RNN methods above were designed for DL of temporal sequences, 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 & Hinton, 1996; Dayan, Hinton, Neal, & Zemel, 1995) may be viewed as an unsupervised (Section 5.6.4) variant thereof (Peter Dayan, personal communication, 1994).

A hybrid approach (Shavlik & Towell, 1989; Towell & Shavlik, 1994) initializes a potentially deep FNN through a domain theory in propositional logic, which may be acquired through explanation-based learning (DeJong & Mooney, 1986; Minton et al., 1989; Mitchell, Keller, & Kedar-Cabelli, 1986). The NN is then fine-tuned through BP (Section 5.5). The NN's depth reflects the longest chain

of reasoning in the original set of logical rules. An extension of this approach (Maclin & Shavlik, 1993; Shavlik, 1994) initializes an RNN by domain knowledge expressed as a Finite State Automaton (FSA). BP-based fine-tuning has become important for later DL systems pre-trained by UL, e.g., Sections 5.10, 5.15.

*5.6.2. Better BP through advanced gradient descent (compare Section 5.24)*

Numerous improvements of steepest descent through BP (Section 5.5) have been proposed. Least-squares methods (Gauss–Newton, Levenberg–Marquardt) (Gauss, 1809; Levenberg, 1944; Marquardt, 1963; Newton, 1687; Schaback & Werner, 1992) and quasi-Newton methods (Broyden–Fletcher–Goldfarb–Shanno, BFGS) (Broyden et al., 1965; Fletcher & Powell, 1963; Goldfarb, 1970; Shanno, 1970) are computationally too expensive for large NNs. Partial BFGS (Battiti, 1992; Saito & Nakano, 1997) and conjugate gradient (Hestenes & Stiefel, 1952; Møller, 1993) as well as other methods (Cauwenberghs, 1993; Schmidhuber, 1989a; Solla, 1988) provide sometimes useful fast alternatives. BP can be treated as a linear least-squares problem (Biegler-König & Bärmann, 1993), where second-order gradient information is passed back to preceding layers.

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 & Saad, 1995).

Only the signs of the error derivatives are taken into account by the successful and widely used BP variant *R-prop* (Riedmiller & Braun, 1993) and the robust variation *iRprop+* (Igel & Hüsken, 2003), which was also successfully applied to RNNs.

The local gradient can be normalized based on the NN architecture (Schraudolph & Sejnowski, 1996), through a diagonalized Hessian approach (Becker & Le Cun, 1989), or related efficient methods (Schraudolph, 2002).

Some algorithms for controlling BP step size adapt a global learning rate (Battiti, 1989; Lapedes & Farber, 1986; LeCun, Simard, & Pearlmutter, 1993; Vogl, Mangis, Rigler, Zink, & Alkon, 1988; Yu, Chen, & Cheng, 1995), while others compute individual learning rates for each weight (Jacobs, 1988; Silva & Almeida, 1990). In online learning, where BP is applied after each pattern presentation, the vario-$\eta$ algorithm (Neuneier & 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, Almeida, Langlois, Amaral, & Redol, 1997).

Many additional tricks for improving NNs have been described (e.g., Montavon, Orr, & Müller, 2012; Orr & Müller, 1998). Compare Section 5.6.3 and recent developments mentioned in Section 5.24.

*5.6.3. Searching for simple, low-complexity, problem-solving NNs (Section 5.24)*

Many researchers used BP-like methods to search for "simple", low-complexity NNs (Section 4.4) with high generalization capability. Most approaches address the *bias/variance dilemma* (Geman, Bienenstock, & Doursat, 1992) through strong prior assumptions. For example, *weight decay* (Hanson & Pratt, 1989; Krogh & Hertz, 1992; Weigend, Rumelhart, & Huberman, 1991) encourages near-zero weights, by penalizing large weights. In a Bayesian framework (Bayes, 1763), weight decay can be derived (Hinton & van Camp, 1993) from Gaussian or Laplacian weight priors (Gauss, 1809; Laplace, 1774); see also Murray and Edwards (1993). An extension of this approach postulates that a distribution of networks with many similar weights generated by Gaussian mixtures is "better" *a priori* (Nowlan & Hinton, 1992).

Often weight priors are implicit in additional penalty terms (MacKay, 1992) or in methods based on *validation sets* (Craven & Wahba, 1979; Eubank, 1988; Golub, Heath, & Wahba, 1979; Hastie & Tibshirani, 1990; Mosteller & Tukey, 1968; Stone, 1974), Akaike's information criterion and *final prediction error* (Akaike, 1970, 1973, 1974), or *generalized prediction error* (Moody, 1992; Moody & Utans, 1994). See also Amari and Murata (1993), Guyon, Vapnik, Boser, Bottou, and Solla (1992), Holden (1994), Vapnik (1992), Wang, Venkatesh, and Judd (1994) and Wolpert (1994). Similar priors (or biases towards simplicity) are implicit in constructive and pruning algorithms, e.g., layer-by-layer *sequential network construction* (e.g., Ash, 1989; Burgess, 1994; Fahlman, 1991; Fritzke, 1994; Gallant, 1988; Honavar & Uhr, 1988, 1993; Ivakhnenko, 1968, 1971; Moody, 1989; Parekh, Yang, & Honavar, 2000; Ring, 1991; Utgoff & Stracuzzi, 2002; Weng, Ahuja, & Huang, 1992) (see also Sections 5.3, 5.11), *input pruning* (Moody, 1992; Refenes, Zapranis, & Francis, 1994), *unit pruning* (e.g., Ivakhnenko, 1968, 1971; Levin, Leen, & Moody, 1994; Mozer & Smolensky, 1989; White, 1989), *weight pruning*, e.g., *optimal brain damage* (LeCun, Denker, & Solla, 1990), and *optimal brain surgeon* (Hassibi & Stork, 1993).

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, with a bias towards fast and short programs (Schmidhuber, 1997) (Section 6.7).

*Flat Minimum Search* (FMS) (Hochreiter & 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* (Bishop, 1993; Carter, Rudolph, & Nucci, 1990; Hanson, 1990; Kerlirzin & Vallet, 1993; Matsuoka, 1992; Minai & Williams, 1994; Murray & Edwards, 1993; Neti, Schneider, & Young, 1992). An MDL-based, Bayesian argument suggests that flat minima correspond to "simple" NNs and low expected overfitting. Compare Section 5.6.4 and more recent developments mentioned in Section 5.24.

*5.6.4. Potential benefits of UL for SL (compare Sections 5.7, 5.10, 5.15)*

The notation of Section 2 introduced teacher-given labels $d_t$. Many papers of the previous millennium, however, were about *unsupervised learning (UL) without a teacher* (e.g., Atick, Li, & Redlich, 1992; Baldi & Hornik, 1989; Barlow, Kaushal, & Mitchison, 1989; Barrow, 1987; Deco & Parra, 1997; Field, 1987; Földiák, 1990; Földiák & Young, 1995; Grossberg, 1976a, 1976b; Hebb, 1949; Kohonen, 1972, 1982, 1988; Kosko, 1990; Martinetz, Ritter, & Schulten, 1990; Miller, 1994; Mozer, 1991; Oja, 1989; Palm, 1992; Pearlmutter & Hinton, 1986; Ritter & Kohonen, 1989; Rubner & Schulten, 1990; Sanger, 1989; Saund, 1994; von der Malsburg, 1973; Watanabe, 1985; Willshaw & von der Malsburg, 1976); see also post-2000 work (e.g., Carreira-Perpinan, 2001; Franzius, Sprekeler, & Wiskott, 2007; Waydo & Koch, 2008; Wiskott & Sejnowski, 2002).

Many UL methods are designed to maximize entropy-related, information-theoretic (Boltzmann, 1909; Kullback & Leibler, 1951; Shannon, 1948) objectives (e.g., Amari, Cichocki, & Yang, 1996; Barlow et al., 1989; Dayan & Zemel, 1995; Deco & Parra, 1997; Field, 1994; Hinton, Dayan, Frey, & Neal, 1995; Linsker, 1988; MacKay & Miller, 1990; Plumbley, 1991; Redlich, 1993; Schmidhuber, 1992b, 1992c; Schraudolph & Sejnowski, 1993; Zemel, 1993; Zemel & Hinton, 1994).

Many do this to uncover and disentangle hidden underlying sources of signals (e.g., Andrade, Chacon, Merelo, & Moran, 1993; Bell & Sejnowski, 1995; Belouchrani, Abed-Meraim, Cardoso, & Moulines, 1997; Cardoso, 1994; Comon, 1994; Hyvärinen,

Karhunen, & Oja, 2001; Jutten & Herault, 1991; Karhunen & Joutsensalo, 1995; Molgedey & Schuster, 1994; Schuster, 1992; Shan & Cottrell, 2014; Shan, Zhang, & Cottrell, 2007; Szabó, Póczos, & Lőrincz, 2006).

Many UL methods automatically and robustly generate distributed, sparse representations of input patterns (Falconbridge, Stamps, & Badcock, 2006; Földiák, 1990; Hinton & Ghahramani, 1997; Hochreiter & Schmidhuber, 1999; Hyvärinen, Hoyer, & Oja, 1999; Lewicki & Olshausen, 1998) through well-known feature detectors (e.g., Olshausen & Field, 1996; Schmidhuber, Eldracher, & Foltin, 1996), such as *off-center-on-surround*-like structures, as well as orientation sensitive edge detectors and Gabor filters (Gabor, 1946). They extract simple features related to those observed in early visual pre-processing stages of biological systems (e.g., De Valois, Albrecht, & Thorell, 1982; Jones & Palmer, 1987).

UL can also serve to extract invariant features from different data items (e.g., Becker, 1991) through *coupled NNs* observing two different inputs (Schmidhuber & Prelinger, 1992), also called *Siamese NNs* (e.g., Bromley et al., 1993; Chen & Salman, 2011; Hadsell, Chopra, & LeCun, 2006; Taylor, Spiro, Bregler, & Fergus, 2011).

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, 1989; Barlow et al., 1989), to disentangle the unknown factors of variation (compare 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) trivializing the task of subsequent naive yet optimal Bayes classifiers (Schmidhuber et al., 1996).

Most early UL FNNs had a single layer. Methods for deeper UL FNNs include hierarchical (Section 4.3) self-organizing Kohonen maps (e.g., Dittenbach, Merkl, & Rauber, 2000; Koikkalainen & Oja, 1990; Lampinen & Oja, 1992; Rauber, Merkl, & Dittenbach, 2002; Versino & Gambardella, 1996), hierarchical *Gaussian potential function* networks (Lee & Kil, 1991), layer-wise UL of feature hierarchies fed into SL classifiers (Behnke, 1999, 2003a), the *Self-Organizing Tree Algorithm* (SOTA) (Herrero, Valencia, & Dopazo, 2001), and nonlinear *Autoencoders* (AEs) with more than 3 (e.g., 5) layers (DeMers & Cottrell, 1993; Kramer, 1991; Oja, 1991). Such AE NNs (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. Certain nonlinear AEs suffer from certain limitations (Baldi, 2012).

Lococode (Hochreiter & Schmidhuber, 1999) uses FMS (Section 5.6.3) to find low-complexity AEs with low-precision weights describable by few bits of information, often producing sparse or factorial codes. *Predictability Minimization* (PM) (Schmidhuber, 1992c) searches for factorial codes through nonlinear feature detectors that fight nonlinear predictors, trying to become both as informative and as unpredictable as possible. PM-based UL was applied not only to FNNs but also to RNNs (e.g., Lindstädt, 1993; Schmidhuber, 1993b). Compare Section 5.10 on UL-based RNN stacks (1991), as well as later UL RNNs (e.g., Klapper-Rybicka, Schraudolph, & Schmidhuber, 2001; Steil, 2007).

### 5.7. 1987: UL through Autoencoder (AE) hierarchies (compare Section 5.15)

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 (Section 5.15). 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 Section 5.5.1) 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) (Melnik, Levy, & Pollack, 2000; Pollack, 1988, 1990), originally used to encode sequential linguistic structures of arbitrary size through a fixed number of hidden units. More recently, RAAMs were also used as unsupervised pre-processors to facilitate deep credit assignment for RL (Gisslen, Luciw, Graziano, & Schmidhuber, 2011) (Section 6.4).

In principle, many UL methods (Section 5.6.4) could be stacked like the AEs above, the history-compressing RNNs of Section 5.10, the *Restricted Boltzmann Machines* (RBMs) of Section 5.15, or hierarchical Kohonen nets (Section 5.6.4), to facilitate subsequent SL. Compare *Stacked Generalization* (Ting & Witten, 1997; Wolpert, 1992), and FNNs that profit from pre-training by *competitive* UL (e.g., Rumelhart & Zipser, 1986) prior to BP-based fine-tuning (Maclin & Shavlik, 1995). See also more recent methods using UL to improve subsequent SL (e.g., Behnke, 1999, 2003a; Escalante-B & Wiskott, 2013).

### 5.8. 1989: BP for convolutional NNs (CNNs, Section 5.4)

In 1989, backpropagation (Section 5.5) was applied (LeCun et al., 1989; LeCun, Boser, et al., 1990; LeCun, Bottou, Bengio, & Haffner, 1998) to Neocognitron-like, weight-sharing, convolutional neural layers (Section 5.4) with adaptive connections. This combination, augmented by *Max-Pooling* (MP, Sections 5.11, 5.16), and sped up on graphics cards (Section 5.19), has become an essential ingredient of many modern, competition-winning, feedforward, visual Deep Learners (Sections 5.19–5.23). 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. CNNs helped to achieve good performance on MNIST (LeCun, Boser, et al., 1990) (CAP depth 5) and on fingerprint recognition (Baldi & Chauvin, 1993); similar CNNs were used commercially in the 1990s.

### 5.9. 1991: Fundamental Deep Learning Problem of gradient descent

A diploma thesis (Hochreiter, 1991) represented a milestone of explicit DL research. As mentioned in Section 5.6, by the late 1980s, experiments had indicated that traditional deep feedforward or recurrent networks are hard to train by backpropagation (BP) (Section 5.5). 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 (Section 1), cumulative backpropagated error signals (Section 5.5.1) either shrink rapidly, or grow out of bounds. In fact, they decay exponentially in the number of layers or CAP depth (Section 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. Later work (Bengio et al., 1994) 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, Bengio, Frasconi, and Schmidhuber (2001) and 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 (the *History Compressor*, Section 5.10) alleviates the problem through unsupervised pre-training for a hierarchy of RNNs. This greatly facilitates subsequent supervised credit assignment through BP (Section 5.5). In the FNN case, similar effects can be achieved through conceptually related AE stacks (Sections 5.7, 5.15) and *Deep Belief Networks* (DBNs, Section 5.15).
II. LSTM-like networks (Sections 5.13, 5.16, 5.17, 5.21–5.23) 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 (Section 5.18). That is basically what is winning many of the image recognition competitions now (Sections 5.19, 5.21, 5.22). (Although this does not really overcome the problem in a fundamental way.)
IV. Hessian-free optimization (Section 5.6.2) can alleviate the problem for FNNs (Martens, 2010; Møller, 1993; Pearlmutter, 1994; Schraudolph, 2002) (Section 5.6.2) and RNNs (Martens & Sutskever, 2011) (Section 5.20).
V. The space of NN weight matrices can also be searched without relying on error gradients, thus avoiding the *Fundamental Deep Learning Problem* altogether. Random weight guessing sometimes works better than more sophisticated methods (Hochreiter & Schmidhuber, 1996). 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, 1997). Some are better solved by using linear methods to obtain optimal weights for connections to output events (Section 2), and *evolving* weights of connections to other events—this is called *Evolino* (Schmidhuber, Wierstra, Gagliolo, & Gomez, 2007). Compare also related RNNs pre-trained by certain UL rules (Steil, 2007), also in the case of *spiking neurons* (Klampfl & Maass, 2013; Yin, Meng, & Jin, 2012) (Section 5.26). Direct search methods are relevant not only for SL but also for more general RL, and are discussed in more detail in Section 6.6.

### 5.10. 1991: UL-based history compression through a deep stack of RNNs

A working *Very Deep Learner* (Section 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 hierarchy 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, Martin, & Atlas, 1994; Dorffner, 1996). 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-organizing 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., Sections 4.4, 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* or *Neural Sequence Chunker*, which can compress data in both space (like feedforward NNs) and time. This is another good example of hierarchical representation learning (Section 4.3). There also is a continuous variant of the history compressor (Schmidhuber, Mozer, & Prelinger, 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. Compare a similar observation for feedforward *Deep Belief Networks* (DBNs, 2006, Section 5.15).

The system was able to learn many previously unlearnable DL tasks. One ancient illustrative DL experiment (Schmidhuber, 1993b) required CAPs (Section 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 BP-based SL became possible. Essentially the system used UL to greatly reduce problem depth. Compare earlier BP-based fine-tuning of NNs initialized by rules of propositional logic (Shavlik & Towell, 1989) (Section 5.6.1).

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 earlier AE hierarchies (1987, Section 5.7) and later *Deep Belief Networks* (2006, Section 5.15), but more general in the sense that it uses sequence-processing RNNs instead of FNNs with unchanging inputs. More recently, well-known entrepreneurs (Hawkins & George, 2006; Kurzweil, 2012) also got interested in HTMs; compare also hierarchical HMMs (e.g., Fine, Singer, & Tishby, 1998), as well as later UL-based recurrent systems (Klampfl & Maass, 2013; Klapper-Rybicka et al., 2001; Steil, 2007; Young, Davis, Mishtal, & Arel, 2014). Clockwork RNNs (Koutník, Greff, Gomez, & Schmidhuber, 2014) also consist of interacting RNN modules with different clock rates, but do not use UL to set those rates. Stacks of RNNs were used in later work on SL with great success, e.g., Sections 5.13, 5.16, 5.17, 5.22.

### 5.11. 1992: Max-Pooling (MP): towards MPCNNs (compare Sections 5.16, 5.19)

The *Neocognitron* (Section 5.4) inspired the *Cresceptron* (Weng et al., 1992), which adapts its topology during training (Section 5.6.3); compare the incrementally growing and shrinking GMDH networks (1965, Section 5.3).

Instead of using alternative local subsampling or WTA methods (e.g., Fukushima, 1980, 2013a; Maass, 2000; Schmidhuber, 1989b), the Cresceptron uses *Max-Pooling* (MP) layers. Here a 2-dimensional layer or array of unit activations is partitioned into smaller rectangular arrays. Each is replaced in a downsampling layer by the activation of its maximally active unit. A later, more complex version of the Cresceptron (Weng, Ahuja, & Huang, 1997) also included "*blurring*" layers to improve object location tolerance.

The neurophysiologically plausible topology of the feedforward HMAX model (Riesenhuber & Poggio, 1999) is very similar to the one of the 1992 Cresceptron (and thus to the 1979 Neocognitron). HMAX does not learn though. Its units have hand-crafted weights; biologically plausible learning rules were later proposed for similar models (e.g., Serre, Riesenhuber, Louie, & Poggio, 2002; Teichmann, Wiltschut, & Hamker, 2012).

When CNNs or convnets (Sections 5.4, 5.8) are combined with MP, they become Cresceptron-like or HMAX-like *MPCNNs* with alternating convolutional and max-pooling layers. Unlike Cresceptron and HMAX, however, MPCNNs are trained by BP (Sections 5.5, 5.16) (Ranzato, Huang, Boureau, & LeCun, 2007). Advantages of doing this were pointed out subsequently (Scherer, Müller, & Behnke, 2010). BP-trained MPCNNs have become central to many modern, competition-winning, feedforward, visual Deep Learners (Sections 5.17, 5.19–5.23).

### 5.12. 1994: early contest-winning 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 & Gershenfeld, 1993). No very deep CAPs (Section 3) were needed though.

### 5.13. 1995: supervised recurrent very Deep Learner (LSTM RNN)

Supervised *Long Short-Term Memory* (LSTM) RNNs (Gers, Schmidhuber, & Cummins, 2000; Hochreiter & Schmidhuber, 1997b; Pérez-Ortiz, Gers, Eck, & Schmidhuber, 2003) could eventually perform similar feats as the deep RNN hierarchy of 1991 (Section 5.10), overcoming the *Fundamental Deep Learning Problem* (Section 5.9) 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*, Section 5.10), dealing with very deep problems (Section 3) (e.g., Gers, Schraudolph, & Schmidhuber, 2002).

The basic LSTM idea is very simple. Some of the units are called *Constant Error Carousels* (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 (Section 5.9) but stay as they are (unless they "flow out" of the CEC to other, typically *adaptive* parts of the NN). CECs are connected to several *nonlinear adaptive* units (some with multiplicative activation functions) needed for learning nonlinear behavior. Weight changes of these units often profit from error signals propagated far back in time through CECs. CECs are 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.

Many different LSTM variants and topologies are allowed. It is possible to evolve good problem-specific topologies (Bayer, Wierstra, Togelius, & Schmidhuber, 2009). Some LSTM variants also use *modifiable* self-connections of CECs (Gers & Schmidhuber, 2001).

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 (Gers et al., 2000; Hochreiter & Schmidhuber, 1997b); Stable generation of precisely timed rhythms, as well as smooth and non-smooth periodic trajectories (Gers & Schmidhuber, 2000). LSTM clearly outperformed previous RNNs on tasks that require learning the rules of regular languages describable by deterministic *Finite State Automata* (FSAs) (Blair & Pollack, 1997; Casey, 1996; Kalinke & Lehmann, 1998; Manolios & Fanelli, 1994; Omlin & Giles, 1996; Siegelmann, 1992; Vahed & Omlin, 2004; Watrous & Kuhn, 1992; Zeng, Goodman, & Smyth, 1994), both in terms of reliability and speed.

LSTM also worked on tasks involving context free languages (CFLs) that cannot be represented by HMMs or similar FSAs discussed in the RNN literature (Andrews, Diederich, & Tickle, 1995; Rodriguez & Wiles, 1998; Rodriguez, Wiles, & Elman, 1999; Steijvers & Grunwald, 1996; Sun, Giles, Chen, & Lee, 1993; Tonkes & Wiles, 1997; Wiles & Elman, 1995). CFL recognition (Lee, 1996) requires the functional equivalent of a runtime stack. Some previous RNNs failed to learn small CFL training sets (Rodriguez & Wiles, 1998). Those that did not (Bodén & Wiles, 2000; Rodriguez et al., 1999) failed to extract the general rules, and did not generalize well on substantially larger test sets. Similar for context-sensitive languages (CSLs) (e.g., Chalup & 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 a decoupled extended Kalman filter (Feldkamp, Prokhorov, Eagen, & Yuan, 1998; Feldkamp, Prokhorov, & Feldkamp, 2003; Haykin, 2001; Kalman, 1960; Puskorius & Feldkamp, 1994; Williams, 1992b) and an LSTM RNN (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 $a^{10,000,000} b^{9,999,999} c^{10,000,000}$. Compare also more recent RNN algorithms able to deal with long time lags (Koutník et al., 2014; Martens & Sutskever, 2011; Schäfer, Udluft, & Zimmermann, 2006; Zimmermann, Tietz, & Grothmann, 2012).

Bi-directional RNNs (BRNNs) (Schuster, 1999; Schuster & Paliwal, 1997) 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, Schuster, and Sagisaka (1999). 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). BRNNs were successfully applied to secondary protein structure prediction (Baldi, Brunak, Frasconi, Pollastri, & Soda, 1999). DAG-RNNs (Baldi & Pollastri, 2003; Wu & Baldi, 2008) generalize BRNNs to multiple dimensions. They learned to predict properties of small organic molecules (Lusci, Pollastri, & Baldi, 2013) as well as protein contact maps (Tegge, Wang, Eickholt, & Cheng, 2009), also in conjunction with a growing deep FNN (Di Lena, Nagata, & Baldi, 2012) (Section 5.21). BRNNs and DAG-RNNs unfold their full potential when combined with the LSTM concept (Graves et al., 2009; Graves & Schmidhuber, 2005, 2009).

Particularly successful in recent competitions are stacks (Section 5.10) of LSTM RNNs (Fernandez, Graves, & Schmidhuber, 2007b; Graves & Schmidhuber, 2009) trained by *Connectionist Temporal Classification* (CTC) (Graves, Fernandez, Gomez, & Schmidhuber, 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 (Section 5.22).

In the early 2000s, speech recognition was dominated by HMMs combined with FNNs (e.g., Bourlard & 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 (Beringer, Graves, Schiel, & Schmidhuber, 2005; Graves, Eck, Beringer, & Schmidhuber, 2003; Graves et al., 2006). In 2007, LSTM outperformed HMMs in keyword spotting tasks (Fernández, Graves, & Schmidhuber, 2007a); compare recent improvements (Indermuhle, Frinken, Fischer, & Bunke, 2011; Wöllmer, Schuller, & Rigoll, 2013). By 2013, LSTM also achieved best known results on the famous TIMIT phoneme recognition benchmark (Graves, Mohamed, & Hinton, 2013) (Section 5.22). Recently, LSTM RNN/HMM hybrids obtained best known performance on medium-vocabulary (Geiger, Zhang, Weninger, Schuller, & Rigoll, 2014) and large-vocabulary speech recognition (Sak, Senior, & Beaufays, 2014).

LSTM is also applicable to robot localization (Förster, Graves, & Schmidhuber, 2007), robot control (Mayer et al., 2008), online driver distraction detection (Wöllmer et al., 2011), and many

other tasks. For example, it helped to improve the state of the art in diverse applications such as protein analysis (Hochreiter & Obermayer, 2005), handwriting recognition (Bluche et al., 2014; Graves, Fernandez, Liwicki, Bunke, & Schmidhuber, 2008; Graves et al., 2009; Graves & Schmidhuber, 2009), voice activity detection (Eyben, Weninger, Squartini, & Schuller, 2013), optical character recognition (Breuel, Ul-Hasan, Al-Azawi, & Shafait, 2013), language identification (Gonzalez-Dominguez, Lopez-Moreno, Sak, Gonzalez-Rodriguez, & Moreno, 2014), prosody contour prediction (Fernandez, Rendel, Ramabhadran, & Hoory, 2014), audio onset detection (Marchi et al., 2014), text-to-speech synthesis (Fan, Qian, Xie, & Soong, 2014), social signal classification (Brueckner & Schuller, 2014), machine translation (Sutskever, Vinyals, & Le, 2014), and others.

RNNs can also be used for metalearning (Prokhorov, Feldkamp, & Tyukin, 2002; Schaul & Schmidhuber, 2010; Schmidhuber, 1987), because they can in principle learn to run their own weight change algorithm (Schmidhuber, 1993a). A successful metalearner (Hochreiter, Younger, & Conwell, 2001) used an LSTM RNN to quickly *learn a learning algorithm* for quadratic functions (compare Section 6.8).

Recently, LSTM RNNs won several international pattern recognition competitions and set numerous benchmark records on large and complex data sets, e.g., Sections 5.17, 5.21, 5.22. Gradient-based LSTM is no panacea though—other methods sometimes outperformed it at least on certain tasks (Jaeger, 2004; Koutník et al., 2014; Martens & Sutskever, 2011; Pascanu, Mikolov, & Bengio, 2013; Schmidhuber et al., 2007); compare Section 5.20.

### 5.14. 2003: more contest-winning/record-setting NNs; successful 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) (Schölkopf et al., 1998; Vapnik, 1995). Nevertheless, at least in certain domains, NNs outperformed other techniques.

A *Bayes NN* (Neal, 2006) based on an ensemble (Breiman, 1996; Dietterich, 2000a; Hashem & Schmeiser, 1992; Schapire, 1990; Ueda, 2000; Wolpert, 1992) of NNs won the *NIPS 2003 Feature Selection Challenge* with secret test set (Neal & Zhang, 2006). The NN was not very deep though—it had two hidden layers and thus rather shallow CAPs (Section 3) of depth 3.

Important for many present competition-winning pattern recognizers (Sections 5.19, 5.21, 5.22) were developments in the CNN department. A BP-trained (LeCun et al., 1989) CNN (Sections 5.4, 5.8) set a new MNIST record of 0.4% (Simard, Steinkraus, & Platt, 2003), using training pattern deformations (Baird, 1990) but no unsupervised pre-training (Sections 5.7, 5.10, 5.15). A standard BP net achieved 0.7% (Simard et al., 2003). Again, the corresponding CAP depth was low. Compare further improvements in Sections 5.16, 5.18, 5.19.

Good image interpretation results (Behnke, 2003b) were achieved with rather deep NNs trained by the BP variant *R-prop* (Riedmiller & Braun, 1993) (Section 5.6.2); here feedback through recurrent connections helped to improve image interpretation. FNNs with CAP depth up to 6 were used to successfully classify high-dimensional data (Vieira & Barradas, 2003).

Deep LSTM RNNs started to obtain certain first speech recognition results comparable to those of HMM-based systems (Graves et al., 2003); compare Sections 5.13, 5.16, 5.21, 5.22.

### 5.15. 2006/7: UL for deep belief networks/AE stacks fine-tuned by BP

While learning networks with numerous non-linear layers date back at least to 1965 (Section 5.3), and explicit DL research results have been published at least since 1991 (Sections 5.9, 5.10), the expression *Deep Learning* was actually coined around 2006, when unsupervised pre-training of deep FNNs helped to accelerate subsequent SL through BP (Hinton, Osindero, & Teh, 2006; Hinton & Salakhutdinov, 2006). Compare earlier terminology on *loading deep networks* (Síma, 1994; Windisch, 2005) and *learning deep memories* (Gomez & Schmidhuber, 2005). Compare also BP-based (Section 5.5) fine-tuning (Section 5.6.1) of (not so deep) FNNs pre-trained by competitive UL (Maclin & Shavlik, 1995).

The *Deep Belief Network* (DBN) is a stack of *Restricted Boltzmann Machines* (RBMs) (Smolensky, 1986), which in turn are *Boltzmann Machines* (BMs) (Hinton & Sejnowski, 1986) with a single layer of feature-detecting units; compare also *Higher-Order BMs* (Memisevic & Hinton, 2010). 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 (Hinton et al., 2006) (equivalent to the data's description length—compare the corresponding observation for RNN stacks, Section 5.10). There are extensions for *Temporal RBMs* (Sutskever, Hinton, & Taylor, 2008).

Without any training pattern deformations (Section 5.14), a DBN fine-tuned by BP achieved 1.2% error rate (Hinton & Salakhutdinov, 2006) on the MNIST handwritten digits (Sections 5.8, 5.14). This result helped to arouse interest in DBNs. DBNs also achieved good results on phoneme recognition, with an error rate of 26.7% on the TIMIT core test set (Mohamed & Hinton, 2010); compare further improvements through FNNs (Deng & Yu, 2014; Hinton, Deng, et al., 2012) and LSTM RNNs (Section 5.22).

A DBN-based technique called *Semantic Hashing* (Salakhutdinov & Hinton, 2009) maps semantically similar documents (of variable size) to nearby addresses in a space of document representations. It outperformed previous searchers for similar documents, such as *Locality Sensitive Hashing* (Buhler, 2001; Datar, Immorlica, Indyk, & Mirrokni, 2004). See the RBM/DBN tutorial (Fischer & Igel, 2014).

Autoencoder (AE) stacks (Ballard, 1987) (Section 5.7) became a popular alternative way of pre-training deep FNNs in unsupervised fashion, before fine-tuning (Section 5.6.1) them through BP (Section 5.5) (Bengio, Lamblin, Popovici, & Larochelle, 2007; Erhan et al., 2010; Vincent, Hugo, Bengio, & Manzagol, 2008). Sparse coding (Section 5.6.4) was formulated as a combination of convex optimization problems (Lee, Battle, Raina, & Ng, 2007). Recent surveys of stacked RBM and AE methods focus on post-2006 developments (Arel, Rose, & Karnowski, 2010; Bengio, 2009). Unsupervised DBNs and AE stacks are conceptually similar to, but in a certain sense less general than, the unsupervised RNN stack-based *History Compressor* of 1991 (Section 5.10), which can process and re-encode not only stationary input patterns, but entire pattern sequences.

### 5.16. 2006/7: improved CNNs/GPU-CNNs/BP for MPCNNs/LSTM stacks

Also in 2006, a BP-trained (LeCun et al., 1989) CNN (Sections 5.4, 5.8) set a new MNIST record of 0.39% (Ranzato, Poultney, Chopra, & LeCun, 2006), using training pattern deformations (Section 5.14) but no unsupervised pre-training. Compare further improvements in Sections 5.18, 5.19. Similar CNNs were used for off-road obstacle avoidance (LeCun, Muller, Cosatto, & Flepp, 2006). A combination of CNNs and TDNNs later learned to map fixed-size representations of variable-size sentences to features relevant for language processing, using a combination of SL and UL (Collobert & Weston, 2008).

2006 also saw an early GPU-based CNN implementation (Chellapilla, Puri, & Simard, 2006) up to 4 times faster than CPU-CNNs; compare also earlier GPU implementations of standard FNNs with a reported speed-up factor of 20 (Oh & Jung, 2004). GPUs

or graphics cards have become more and more important for DL in subsequent years (Sections 5.18–5.22).

In 2007, BP (Section 5.5) was applied for the first time (Ranzato et al., 2007) to Neocognitron-inspired (Section 5.4), Cresceptron-like (or HMAX-like) MPCNNs (Section 5.11) with alternating convolutional and max-pooling layers. BP-trained MPCNNs have become an essential ingredient of many modern, competition-winning, feedforward, visual Deep Learners (Sections 5.17, 5.19–5.23).

Also in 2007, hierarchical stacks of LSTM RNNs were introduced (Fernandez et al., 2007b). They can be trained by hierarchical *Connectionist Temporal Classification* (CTC) (Graves et al., 2006). For tasks of sequence labeling, every LSTM RNN level (Section 5.13) predicts a sequence of labels fed to the next level. Error signals at every level are back-propagated through all the lower levels. On spoken digit recognition, LSTM stacks outperformed HMMs, despite making fewer assumptions about the domain. LSTM stacks do not necessarily require unsupervised pre-training like the earlier UL-based RNN stacks (Schmidhuber, 1992b) of Section 5.10.

### 5.17. 2009: first official competitions won by RNNs, and with MPCNNs

Stacks of LSTM RNNs trained by CTC (Sections 5.13, 5.16) became the first RNNs to win official international pattern recognition contests (with secret test sets known only to the organizers). 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, performing simultaneous segmentation and recognition. Compare Graves and Jaitly (2014), Graves and Schmidhuber (2005), Graves et al. (2009), Graves et al. (2013) and Schmidhuber, Ciresan, Meier, Masci, and Graves (2011) (Section 5.22).

To detect human actions in surveillance videos, a 3-dimensional CNN (e.g., Jain & Seung, 2009; Prokhorov, 2010), combined with SVMs, was part of a larger system (Yang et al., 2009) using a *bag of features* approach (Nowak, Jurie, & Triggs, 2006) to extract regions of interest. The system won three 2009 TRECVID competitions. These were possibly the first official international contests won with the help of (MP)CNNs (Section 5.16). An improved version of the method was published later (Ji, Xu, Yang, & Yu, 2013).

2009 also saw a GPU-DBN implementation (Raina, Madhavan, & Ng, 2009) orders of magnitudes faster than previous CPU-DBNs (see Section 5.15); see also Coates et al. (2013). The *Convolutional DBN* (Lee, Grosse, Ranganath, & Ng, 2009) (with a probabilistic variant of MP, Section 5.11) combines ideas from CNNs and DBNs, and was successfully applied to audio classification (Lee, Pham, Largman, & Ng, 2009).

### 5.18. 2010: plain backprop (+ distortions) on GPU breaks MNIST record

In 2010, a new MNIST (Section 5.8) record of 0.35% error rate was set by good old BP (Section 5.5) in deep but otherwise standard NNs (Ciresan, Meier, Gambardella, & Schmidhuber, 2010), using neither unsupervised pre-training (e.g., Sections 5.7, 5.10, 5.15) nor convolution (e.g., Sections 5.4, 5.8, 5.14, 5.16). However, training pattern deformations (e.g., Section 5.14) 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 Section 5.15.

Since BP was 3–5 decades old by then (Section 5.5), and pattern deformations 2 decades (Baird, 1990) (Section 5.14), these results seemed to suggest that advances in exploiting modern computing hardware were more important than advances in algorithms.

### 5.19. 2011: MPCNNs on GPU achieve superhuman vision performance

In 2011, a flexible *GPU-implementation* (Ciresan, Meier, Masci, Gambardella, & Schmidhuber, 2011) of *Max-Pooling (MP) CNNs or Convnets* was described (a GPU-MPCNN), building on earlier MP work (Weng et al., 1992) (Section 5.11) CNNs (Fukushima, 1979; LeCun et al., 1989) (Sections 5.4, 5.8, 5.16), and on early GPU-based CNNs *without* MP (Chellapilla et al., 2006) (Section 5.16); compare early GPU-NNs (Oh & Jung, 2004) and GPU-DBNs (Raina et al., 2009) (Section 5.17). MPCNNs have alternating convolutional layers (Section 5.4) and max-pooling layers (MP, Section 5.11) topped by standard fully connected layers. All weights are trained by BP (Sections 5.5, 5.8, 5.16) (Ranzato et al., 2007; Scherer et al., 2010). GPU-MPCNNs have become essential for many contest-winning FNNs (Sections 5.21, 5.22).

*Multi-Column* GPU-MPCNNs (Ciresan, Meier, Masci, & Schmidhuber, 2011) are committees (Breiman, 1996; Dietterich, 2000a; Hashem & Schmeiser, 1992; Schapire, 1990; Ueda, 2000; Wolpert, 1992) 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), the contest-winning ensemble Bayes-NN (Neal, 2006) of Section 5.14, and recent related work (Shao, Wu, & Li, 2014).

An ensemble of GPU-MPCNNs was the first system to achieve superhuman visual pattern recognition (Ciresan, Meier, Masci, Schmidhuber, 2011; Ciresan, Meier, Masci, & Schmidhuber, 2012) in a controlled competition, namely, the IJCNN 2011 traffic sign recognition contest in San Jose (CA) (Stallkamp, Schlipsing, Salmen, & Igel, 2011, 2012). This is of interest for fully autonomous, self-driving cars in traffic (e.g., Dickmanns et al., 1994). The GPU-MPCNN ensemble obtained 0.56% error rate and was twice better than human test subjects, three times better than the closest artificial NN competitor (Sermanet & 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, Meier, Masci, Schmidhuber, 2011) vs 1.03% for second place (Sermanet & LeCun, 2011). After the deadline, the organizers 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., 2012) (the organizers 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 recognizers. Nevertheless, by 2011, deep NNs could already learn to rival them in important limited visual domains.

An ensemble of GPU-MPCNNs was also the first method to achieve human-competitive performance (around 0.2%) on MNIST (Ciresan, Meier, & Schmidhuber, 2012a). This represented a dramatic improvement, since by then the MNIST record had hovered around 0.4% for almost a decade (Sections 5.14, 5.16, 5.18).

Given all the prior work on (MP)CNNs (Sections 5.4, 5.8, 5.11, 5.16) and GPU-CNNs (Section 5.16), 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, most *feedforward* competition-winning deep NNs are (ensembles of) GPU-MPCNNs (Sections 5.21–5.23).

### 5.20. 2011: Hessian-free optimization for RNNs

Also in 2011 it was shown (Martens & Sutskever, 2011) that Hessian-free optimization (e.g., Møller, 1993; Pearlmutter, 1994; Schraudolph, 2002) (Section 5.6.2) can alleviate the *Fundamental Deep Learning Problem* (Section 5.9) in RNNs, outperforming standard gradient-based LSTM RNNs (Section 5.13) on several tasks. Compare other RNN algorithms (Jaeger, 2004; Koutník et al., 2014; Pascanu, Mikolov, et al., 2013; Schmidhuber et al., 2007) that also at least sometimes yield better results than steepest descent for LSTM RNNs.

### 5.21. 2012: first contests won on ImageNet, object detection, segmentation

In 2012, an ensemble of GPU-MPCNNs (Section 5.19) achieved best results on the *ImageNet* classification benchmark (Krizhevsky, Sutskever, & Hinton, 2012), which is popular in the computer vision community. Here relatively large image sizes of 256 × 256 pixels were necessary, as opposed to only 48 × 48 pixels for the 2011 traffic sign competition (Section 5.19). See further improvements in Section 5.22.

Also in 2012, the biggest NN so far ($10^9$ free parameters) was trained in unsupervised mode (Sections 5.7, 5.15) 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 1000 standard machines with 16,000 cores.

So by 2011/2012, excellent results had been achieved by Deep Learners in image *recognition and classification* (Sections 5.19, 5.21). 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 ensemble of GPU-MPCNNs, Section 5.19) to win a contest on visual *object detection* (Ciresan, Giusti, Gambardella, & Schmidhuber, 2013) in large images of several million pixels (ICPR, 2012; Roux 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 (Ciresan, Giusti, Gambardella, & Schmidhuber, 2012), again through an GPU-MPCNN ensemble (Segmentation of Neuronal Structures in EM Stacks Challenge, 2012).[2] EM stacks are relevant for the recently approved huge brain projects in Europe and the US (e.g., Markram, 2012). 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 (e.g., Turaga et al., 2010). Deep Multi-Column 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, Giusti, et al., 2012) profit from fast MPCNN-based image scans that avoid redundant computations. Recent MPCNN scanners speed up naive implementations by up to three orders of magnitude (Giusti, Ciresan, Masci, Gambardella, & Schmidhuber, 2013; Masci, Giusti, Ciresan, Fricout, & Schmidhuber, 2013); compare earlier efficient methods for CNNs *without* MP (Vaillant, Monrocq, & LeCun, 1994).

Also in 2012, a system consisting of growing deep FNNs and 2D-BRNNs (Di Lena et al., 2012) won the CASP 2012 contest on protein contact map prediction. On the IAM-OnDoDB benchmark, LSTM RNNs (Section 5.13) outperformed all other methods (HMMs, SVMs) on online mode detection (Indermuhle, Frinken, & Bunke, 2012; Otte, Krechel, Liwicki, & Dengel, 2012) and keyword spotting (Indermuhle et al., 2011). On the long time lag problem of language modeling, LSTM RNNs outperformed all statistical approaches on the IAM-DB benchmark (Frinken et al., 2012); improved results were later obtained through a combination of NNs and HMMs (Zamora-Martínez et al., 2014). Compare earlier RNNs for object recognition through iterative image interpretation (Behnke, 2002, 2003b; Behnke & Rojas, 1998); see also more recent publications (O'Reilly, Wyatte, Herd, Mingus, & Jilk, 2013; Wyatte, Curran, & O'Reilly, 2012) extending work on biologically plausible learning rules for RNNs (O'Reilly, 1996).

### 5.22. 2013-: more contests and benchmark records

A stack (Fernandez et al., 2007b; Graves & Schmidhuber, 2009) (Section 5.10) of bi-directional LSTM RNNs (Graves & Schmidhuber, 2005) trained by CTC (Sections 5.13, 5.17) 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 (Section 5.15).

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). LSTM-based systems also set benchmark records in *language identification* (Gonzalez-Dominguez et al., 2014), medium-vocabulary *speech recognition* (Geiger et al., 2014), *prosody contour prediction* (Fernandez et al., 2014), *audio onset detection* (Marchi et al., 2014), *text-to-speech synthesis* (Fan et al., 2014), and *social signal classification* (Brueckner & Schulter, 2014).

An LSTM RNN was used to estimate the state posteriors of an HMM; this system beat the previous state of the art in *large vocabulary speech recognition* (Sak, Senior, et al., 2014; Sak, Vinyals, et al., 2014). Another LSTM RNN with hundreds of millions of

[2] 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—see Section 5.17.

connections was used to rerank hypotheses of a statistical machine translation system; this system beat the previous state of the art in *English to French translation* (Sutskever et al., 2014).

A new record on the *ICDAR Chinese handwriting recognition benchmark* (over 3700 classes) was set on a desktop machine by an ensemble of GPU-MPCNNs (Section 5.19) with almost human performance (Ciresan & Schmidhuber, 2013); compare (Yin, Wang, Zhang, & Liu, 2013).

The *MICCAI 2013 Grand Challenge on Mitosis Detection* (Veta, Viergever, Pluim, Stathonikos, & van Diest, 2013) also was won by an object-detecting GPU-MPCNN ensemble (Ciresan et al., 2013). Its data set was even larger and more challenging than the one of ICPR 2012 (Section 5.21): a real-world data set including many ambiguous cases and frequently encountered problems such as imperfect slide staining.

Three 2D-CNNs (with *mean-pooling* instead of MP, Section 5.11) observing three orthogonal projections of 3D images outperformed traditional full 3D methods on the task of segmenting tibial cartilage in low field knee MRI scans (Prasoon et al., 2013).

Deep GPU-MPCNNs (Section 5.19) also helped to achieve new best results on important benchmarks of the computer vision community: ImageNet classification (Szegedy et al., 2014; Zeiler & Fergus, 2013) and – in conjunction with traditional approaches – PASCAL object detection (Girshick, Donahue, Darrell, & Malik, 2013). They also learned to predict bounding box coordinates of objects in the Imagenet 2013 database, and obtained state-of-the-art results on tasks of localization and detection (Sermanet et al., 2013). GPU-MPCNNs also helped to recognize multi-digit numbers in Google Street View images (Goodfellow, Bulatov, Ibarz, Arnoud, & Shet, 2014), where part of the NN was trained to *count* visible digits; compare earlier work on detecting "numerosity" through DBNs (Stoianov & Zorzi, 2012). This system also excelled at recognizing distorted synthetic text in *reCAPTCHA* puzzles. Other successful CNN applications include scene parsing (Farabet, Couprie, Najman, & LeCun, 2013), object detection (Szegedy, Toshev, & Erhan, 2013), shadow detection (Khan, Bennamoun, Sohel, & Togneri, 2014), video classification (Karpathy et al., 2014), and Alzheimer's disease neuroimaging (Li et al., 2014).

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.

### 5.23. Currently successful techniques: LSTM RNNs and 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) (Sections 5.13, 5.17, 5.21, 5.22), or (b) feedforward GPU-MPCNNs (2011, Sections 5.19, 5.21, 5.22) based on CNNs (1979, Section 5.4) with MP (1992, Section 5.11) trained through BP (1989–2007, Sections 5.8, 5.16).

Exceptions include two 2011 contests (Goodfellow, Courville, & Bengio, 2011, 2012; Mesnil et al., 2011) specialized on *Transfer Learning* from one data set to another (e.g., Caruana, 1997; Pan & Yang, 2010; Schmidhuber, 2004). However, deep GPU-MPCNNs do allow for *pure SL-based transfer* (Ciresan, Meier, & Schmidhuber, 2012b), where pre-training on one training set greatly improves performance on quite different sets, also in more recent studies (Donahue et al., 2013; Oquab, Bottou, Laptev, & Sivic, 2013). In fact, deep MPCNNs pre-trained by SL can extract useful features from quite diverse off-training-set images, yielding better results than traditional, widely used features such as SIFT (Lowe, 1999, 2004) on many vision tasks (Razavian, Azizpour, Sullivan, & Carlsson, 2014). To deal with changing data sets, slowly learning deep NNs were also combined with rapidly adapting "*surface*" NNs (Kak, Chen, & Wang, 2010).

Remarkably, in the 1990s a trend went from partially unsupervised RNN stacks (Section 5.10) to *purely* supervised LSTM RNNs (Section 5.13), just like in the 2000s a trend went from partially unsupervised FNN stacks (Section 5.15) to *purely* supervised MPCNNs (Sections 5.16–5.22). Nevertheless, in many applications it can still be advantageous to combine the best of both worlds—*supervised* learning and *unsupervised* pre-training (Sections 5.10, 5.15).

### 5.24. Recent tricks for improving SL deep NNs (compare Sections 5.6.2, 5.6.3)

DBN training (Section 5.15) can be improved through gradient enhancements and automatic learning rate adjustments during stochastic gradient descent (Cho, 2014; Cho, Raiko, & Ilin, 2013), and through Tikhonov-type (Tikhonov, Arsenin, & John, 1977) regularization of RBMs (Cho, Ilin, & Raiko, 2012). Contractive AEs (Rifai, Vincent, Muller, Glorot, & Bengio, 2011) discourage hidden unit perturbations in response to input perturbations, similar to how FMS (Section 5.6.3) for Lococode AEs (Section 5.6.4) discourages output perturbations in response to weight perturbations.

Hierarchical CNNs in a *Neural Abstraction Pyramid* (e.g., Behnke, 2003b, 2005) were trained to reconstruct images corrupted by structured noise (Behnke, 2001), thus enforcing increasingly abstract image representations in deeper and deeper layers. *Denoising AEs* later used a similar procedure (Vincent et al., 2008).

*Dropout* (Ba & Frey, 2013; Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012) removes units from NNs during training to improve generalization. Some view it as an ensemble method that trains multiple data models simultaneously (Baldi & Sadowski, 2014). 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 (Pachitariu & Sahani, 2013; Pascanu, Gulcehre, Cho, & Bengio, 2013; Pham, Kermorvant, & Louradour, 2013). A deterministic approximation coined *fast dropout* (Wang & Manning, 2013) can lead to faster learning and evaluation and was adapted for RNNs (Bayer, Osendorfer, Chen, Urban, & van der Smagt, 2013). Dropout is closely related to older, biologically plausible techniques for adding noise to neurons or synapses during training (e.g., An, 1996; Hanson, 1990; Jim, Giles, & Horne, 1995; Murray & Edwards, 1993; Nadal & Parga, 1994; Schuster, 1992), which in turn are closely related to finding perturbation-resistant low-complexity NNs, e.g., through FMS (Section 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 (Section 5.6.3) represent a bias towards limited memory capacity (e.g., Pascanu, Mikolov, et al., 2013). Compare recent work on variational recurrent AEs (Bayer & Osendorfer, 2014).

The activation function $f$ of *Rectified Linear Units* (ReLUs) is $f(x) = x$ for $x > 0$, $f(x) = 0$ otherwise—compare the old concept of half-wave rectified units (Malik & Perona, 1990). ReLU NNs are useful for RBMs (Maas, Hannun, & Ng, 2013; Nair & Hinton, 2010), outperformed sigmoidal activation functions in deep NNs (Glorot, Bordes, & Bengio, 2011), and helped to obtain best results on several benchmark problems across multiple domains (e.g., Dahl, Sainath, & Hinton, 2013; Krizhevsky et al., 2012).

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, Masci, Kazerounian, Gomez, & Schmidhuber, 2013). In this context, choosing a learning algorithm may be more important than choosing activation functions (Goodfellow, Mirza, Da, Courville, & Bengio, 2014). *Maxout* NNs (Goodfellow, Warde-Farley, Mirza, Courville, & Bengio, 2013) combine competitive interactions and dropout (see above) to achieve excellent results on certain

benchmarks. 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 (Bishop, 2006; Hertz, Krogh, & Palmer, 1991), 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 regularizers (Sections 4.4, 5.6.3), becoming fast and *slim* when necessary. One may penalize the task-specific total length of connections (e.g., Clune, Mouret, & Lipson, 2013; Legenstein & Maass, 2002; Schmidhuber, 2012, 2013b) and communication costs of SLIM NNs implemented on the 3-dimensional brain-like multi-processor hardware to be expected in the future.

*RmsProp* (Schaul, Zhang, & LeCun, 2013; Tieleman & Hinton, 2012) can speed up first order gradient descent methods (Sections 5.5, 5.6.2); compare vario-$\eta$ (Neuneier & Zimmermann, 1996), *Adagrad* (Duchi, Hazan, & Singer, 2011) and *Adadelta* (Zeiler, 2012). DL in NNs can also be improved by transforming hidden unit activations such that they have zero output and slope on average (Raiko, Valpola, & LeCun, 2012). Many additional, older tricks (Sections 5.6.2, 5.6.3) should also be applicable to today's deep NNs; compare (Montavon et al., 2012; Orr & Müller, 1998).

### 5.25. 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 Section 5.21 (Ciresan, Giusti, et al., 2012; Segmentation of Neuronal Structures in EM Stacks Challenge, 2012).

The feature detectors learned by single-layer visual ANNs are similar to those found in early visual processing stages of BNNs (e.g., Section 5.6.4). Likewise, the feature detectors learned in *deep* layers of visual ANNs should be highly predictive of what neuroscientists will find in *deep* layers of BNNs. While the visual cortex of BNNs may use quite different learning algorithms, its objective function to be minimized may be quite similar to the one of visual ANNs. In fact, results obtained with relatively deep artificial DBNs (Lee, Ekanadham, & Ng, 2007) and CNNs (Yamins, Hong, Cadieu, & DiCarlo, 2013) seem compatible with insights about the visual pathway in the primate cerebral cortex, which has been studied for many decades (e.g., Bichot, Rossi, & Desimone, 2005; Connor, Brincat, & Pasupathy, 2007; Desimone, Albright, Gross, & Bruce, 1984; DiCarlo, Zoccolan, & Rust, 2012; Felleman & Van Essen, 1991; Hubel & Wiesel, 1968; Hung, Kreiman, Poggio, & DiCarlo, 2005; Kobatake & Tanaka, 1994; Kriegeskorte et al., 2008; Lennie & Movshon, 2005; Logothetis, Pauls, & Poggio, 1995; Perrett, Hietanen, Oram, Benson, & Rolls, 1992; Perrett, Rolls, & Caan, 1982); compare a computer vision-oriented survey (Kruger et al., 2013).

### 5.26. DL with spiking neurons?

Many recent DL results profit from GPU-based traditional deep NNs, e.g., Sections 5.16–5.19. Current GPUs, however, are little ovens, much hungrier for energy than biological brains, whose neurons efficiently communicate by brief spikes (FitzHugh, 1961; Hodgkin & Huxley, 1952; Nagumo, Arimoto, & Yoshizawa, 1962), and often remain quiet. Many computational models of such *spiking neurons* have been proposed and analyzed (e.g., Amit & Brunel, 1997; Bohte, Kok, & La Poutre, 2002; Brea, Senn, & Pfister, 2013; Brette et al., 2007; Brunel, 2000; Deco & Rolls, 2005; Gerstner & Kistler, 2002; Gerstner & van Hemmen, 1992; Hoerzer, Legenstein, & Maass, 2014; Izhikevich et al., 2003; Kasabov, 2014; Kempter, Gerstner, & Van Hemmen, 1999; Kistler, Gerstner, & van Hemmen, 1997; Maass, 1996, 1997; Maex & Orban, 1996; Nessler, Pfeiffer, Buesing, & Maass, 2013; Rezende & Gerstner, 2014; Seung, 2003; Song, Miller, & Abbott, 2000; Stemmler, 1996; Stoop, Schindler, & Bunimovich, 2000; Tsodyks, Pawelzik, & Markram, 1998; Tsodyks, Skaggs, Sejnowski, & McNaughton, 1996; Zipser, Kehoe, Littlewort, & Fuster, 1993).

Future energy-efficient hardware for DL in NNs may implement aspects of such models (e.g., Fieres, Schemmel, & Meier, 2008; Glackin, McGinnity, Maguire, Wu, & Belatreche, 2005; Indiveri et al., 2011; Jin et al., 2010; Khan et al., 2008; Liu et al., 2001; Merolla et al., 2014; Neil & Liu, 2014; Roggen, Hofmann, Thoma, & Floreano, 2003; Schemmel, Grubl, Meier, & Mueller, 2006; Serrano-Gotarredona et al., 2009). A simulated, event-driven, spiking variant (Neftci, Das, Pedroni, Kreutz-Delgado, & Cauwenberghs, 2014) of an RBM (Section 5.15) was trained by a variant of the *Contrastive Divergence* algorithm (Hinton, 2002). Spiking nets were evolved to achieve reasonable performance on small face recognition data sets (Wysoski, Benuskova, & Kasabov, 2010) and to control simple robots (Floreano & Mattiussi, 2001; Hagras, Pounds-Cornish, Colley, Callaghan, & Clarke, 2004). A spiking DBN with about 250,000 neurons (as part of a larger NN; Eliasmith, 2013; Eliasmith et al., 2012) achieved 6% error rate on MNIST; compare similar results with a spiking DBN variant of depth 3 using a neuromorphic event-based sensor (O'Connor, Neil, Liu, Delbruck, & Pfeiffer, 2013). In practical applications, however, current artificial networks of spiking neurons cannot yet compete with the best traditional deep NNs (e.g., compare MNIST results of Section 5.19).

## 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 (see surveys, e.g., Kaelbling et al., 1996; Sutton & Barto, 1998; Wiering & van Otterlo, 2012). 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 (Section 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 (Section 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 $P = NP$ (Cook, 1971; Levin, 1973b) would also set limits for what is achievable by general RL. Compare more specific limitations, e.g., Blondel and Tsitsiklis (2000), Madani, Hanks, and Condon (2003) and Vlassis, Littman, and Barber (2012). The following subsections mostly focus on certain obvious intersections between 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 FNN controller $C$ interacting with a *deterministic, predictable* environment, a separate FNN called $M$ can learn to become $C$'s *world model* through *system identification*, predicting $C$'s inputs from previous actions and inputs (e.g., Cochocki & Unbehauen, 1993; Ge, Hang, Lee, & Zhang, 2010; Gomi & Kawato, 1993; Jordan, 1988; Jordan & Rumelhart, 1990; Levin & Narendra, 1995; Ljung, 1998; Miller, Werbos, & Sutton, 1995; Munro, 1987; Narendra & Parthasarathy, 1990; Prokhorov, Puskorius, & Feldkamp, 2001; Robinson & Fallside, 1989; Schmidhuber, 1990d; Werbos, 1981, 1987, 1989a, 1989b, 1992). Assume $M$ has

learned to produce accurate predictions. We can use $M$ to substitute the environment. Then $M$ and $C$ form an RNN where $M$'s outputs become inputs of $C$, whose outputs (actions) in turn become inputs of $M$. Now BP for RNNs (Section 5.5.1) 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 (Section 6.2). Decades ago, the method was used to learn to back up a model truck (Nguyen & Widrow, 1989). An RL active vision system used it to learn sequential shifts (saccades) of a fovea, to detect targets in visual scenes (Schmidhuber & Huber, 1991), thus learning to control selective attention. Compare RL-based attention learning without NNs (Whitehead, 1992).

To allow for memories of previous events in *partially observable worlds* (Section 6.3), the most general variant of this technique uses RNNs instead of FNNs to implement both $M$ and $C$ (Feldkamp & Puskorius, 1998; Schmidhuber, 1990d, 1991c). This may cause deep CAPs not only for $C$ but also for $M$.

$M$ can also be used to optimize 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 optimizing 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, Wiesel, Tenchio, & Simon, 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* (e.g., 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 (Oudeyer, Baranes, & Kaplan, 2013; Singh, Barto, & Chentanez, 2005). 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 (Bertsekas & Tsitsiklis, 1996; Samuel, 1959) 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 (Sections 3, 6.1) by using the *Dynamic Programming* (DP) trick (Bellman, 1957). The latter is often explained in a probabilistic framework (e.g., Sutton & Barto, 1998), but its basic idea can already be conveyed in a deterministic setting. For simplicity, using the notation of Section 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. Now search for 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., Abounadi, Bertsekas, & Borkar, 2002; Baird, 1995; Baird & Moore, 1999; Barto, Sutton, & Anderson, 1983; Bertsekas, 2001; Bradtke, Barto, & Kaelbling, 1996; Brafman & Tennenholtz, 2002; Kaelbling, Littman, & Cassandra, 1995; Lagoudakis & Parr, 2003; Maei & Sutton, 2010; Mahadevan, 1996; Meuleau, Peshkin, Kim, & Kaelbling, 1999; Moore & Atkeson, 1993; Morimoto & Doya, 2000; Peng & Williams, 1996; Prokhorov & Wunsch, 1997; Rummery & Niranjan, 1994; Santamaría, Sutton, & Ram, 1997; Schwartz, 1993; Singh, 1994; Sutton & Barto, 1998; Sutton, Szepesvári, & Maei, 2008; Tsitsiklis & van Roy, 1996; van Hasselt, 2012; Watkins, 1989; Watkins & Dayan, 1992; Wiering & Schmidhuber, 1998b). 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-CNN was used in a traditional RL framework to play several Atari 2600 computer games directly from 84 × 84 pixel 60 Hz video input (Mnih et al., 2013), using experience replay (Lin, 1993), extending previous work on *Neural Fitted Q-Learning* (NFQ) (Riedmiller, 2005). Even better results are achieved by using (slow) Monte Carlo tree planning to train comparatively fast deep NNs (Guo, Singh, Lee, Lewis, & Wang, 2014). Compare RBM-based RL (Sallans & Hinton, 2004) with high-dimensional inputs (Elfwing, Otsuka, Uchibe, & Doya, 2010), earlier RL Atari players (Grüttner, Sehnke, Schaul, & Schmidhuber, 2010), and an earlier, raw video-based RL NN for computer games (Koutník, Cuccu, Schmidhuber, & Gomez, 2013) trained by *Indirect Policy Search* (Section 6.7).

### 6.3. Deep RL RNNs for partially observable MDPs (POMDPs)

The *Markov assumption* (Section 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., Boutilier & Poole, 1996; Jaakkola, Singh, & Jordan, 1995; Kaelbling et al., 1995; Kimura, Miyazaki, & Kobayashi, 1997; Lin, 1993; Littman, Cassandra, & Kaelbling, 1995; McCallum, 1996; Otsuka, Yoshimoto, & Doya, 2010; Ring, 1991, 1993, 1994; Schmidhuber, 1990d, 1991c; Teller, 1994; Wiering & Schmidhuber, 1996, 1998a; Williams, 1992a). A naive way of implementing memories without leaving the MDP framework (Section 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., Bakker, 2002; Lin, 1993; Schmidhuber, 1990b, 1991c). For example, deep LSTM RNNs were used in this way for RL robots (Bakker, Zhumatiy, Gruener, & Schmidhuber, 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 Section 6.1.
3. Use an RNN for RL by *Direct Search* (Section 6.6) or *Indirect Search* (Section 6.7) in weight space.

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

### 6.4. RL facilitated by deep UL in FNNs and RNNs

RL machines may profit from UL for input preprocessing (e.g., Jodogne & Piater, 2007). In particular, an UL NN can learn to compactly encode environmental inputs such as images or videos, e.g., Sections 5.7, 5.10, 5.15. The compact codes (instead of the high-dimensional raw data) can be fed into an RL machine, whose job thus may become much easier (Cuccu, Luciw, Schmidhuber, & Gomez, 2011; Legenstein, Wilbert, & Wiskott, 2010), just like SL may profit from UL, e.g., Sections 5.7, 5.10, 5.15. For example, NFQ (Riedmiller, 2005) was applied to real-world control tasks (Lange & Riedmiller, 2010; Riedmiller, Lange, & Voigtlaender, 2012) where purely visual inputs were compactly encoded by deep autoencoders (Sections 5.7, 5.15). RL combined with UL based on *Slow Feature Analysis* (Kompella, Luciw, & Schmidhuber, 2012; Wiskott & Sejnowski, 2002) enabled a real humanoid robot to learn skills from raw high-dimensional video streams (Luciw, Kompella, Kazerounian, & Schmidhuber, 2013). To deal with POMDPs (Section 6.3) involving high-dimensional inputs, RBM-based RL was used (Otsuka, 2010), and a RAAM (Pollack, 1988) (Section 5.7) was employed as a deep unsupervised sequence encoder for RL (Gisslen et al., 2011). Certain types of RL and UL also were combined in biologically plausible RNNs with spiking neurons (Section 5.26) (e.g., Klampfl & Maass, 2013; Rezende & Gerstner, 2014; Yin et al., 2012).

### 6.5. Deep hierarchical RL (HRL) and subgoal learning with FNNs and RNNs

Multiple learnable levels of abstraction (Bengio et al., 2013; Deng & Yu, 2014; Fu, 1977; Lenat & Brown, 1984; Ring, 1994) 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 & Wahnsiedler, 1992). Numerous alternative HRL techniques have been proposed (e.g., Bakker & Schmidhuber, 2004; Barto & Mahadevan, 2003; Dietterich, 2000b; Doya, Samejima, Katagiri, & Kawato, 2002; Ghavamzadeh & Mahadevan, 2003; Jameson, 1991; Menache, Mannor, & Shimkin, 2002; Moore & Atkeson, 1995; Precup, Sutton, & Singh, 1998; Ring, 1991, 1994; Samejima, Doya, & Kawato, 2003; Simsek & Barto, 2008; Tenenberg, Karlsson, & Whitehead, 1993; Weiss, 1994; Whiteson, Kohl, Miikkulainen, & Stone, 2005). While HRL frameworks such as *Feudal RL* (Dayan & Hinton, 1993) and *options* (Barto, Singh, & Chentanez, 2004; Singh et al., 2005; Sutton, Precup, & Singh, 1999) do not directly address the problem of automatic subgoal discovery, *HQ-Learning* (Wiering & Schmidhuber, 1998a) automatically decomposes POMDPs (Section 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-based RL sub-modules into self-organizing, 2-dimensional motor control maps (Ring, Schaul, & Schmidhuber, 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 Section 6.8, yet both practical and more general than most traditional RL algorithms (Section 6.2), are methods for *Direct Policy Search* (DS). Without a need for value functions or Markovian assumptions (Sections 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 (Sections 5.5, 6.3, 6.1), CAP depth (Sections 3, 5.9) 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 (Aberdeen, 2003; Baxter & Bartlett, 2001; Ghavamzadeh & Mahadevan, 2003; Grondman, Busoniu, Lopes, & Babuska, 2012; Grüttner et al., 2010; Heess, Silver, & Teh, 2012; Kohl & Stone, 2004; Peters, 2010; Peters & Schaal, 2008a, 2008b; Rückstieß, Felder, & Schmidhuber, 2008; Sehnke et al., 2010; Sutton, McAllester, Singh, & Mansour, 1999; Wierstra, Foerster, Peters, & Schmidhuber, 2010; Wierstra, Schaul, Peters, & Schmidhuber, 2008; Williams, 1986, 1988, 1992a). 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) (Fogel, Owens, & Walsh, 1966; Goldberg, 1989; Holland, 1975; Rechenberg, 1971; Schwefel, 1974) in a series of trials. Here several policies are represented by a population of NNs improved through mutations and/or repeated recombinations of the population's fittest individuals (e.g., Fogel, Fogel, & Porto, 1990; Happel & Murre, 1994; Maniezzo, 1994; Montana & Davis, 1989; Nolfi, Parisi, & Elman, 1994). Compare *Genetic Programming* (GP) (Cramer, 1985) (see also Smith, 1980) which can be used to evolve computer programs of variable size (Dickmanns, Schmidhuber, & Winklhofer, 1987; Koza, 1992), and *Cartesian GP* (Miller & Harding, 2009; Miller & Thomson, 2000) for evolving graph-like programs, including NNs (Khan, Khan, & Miller, 2010) and their topology (Turner & Miller, 2013). Related methods include *probability distribution-based EAs* (Baluja, 1994; Larraanaga & Lozano, 2001; Sałustowicz & Schmidhuber, 1997; Saravanan & Fogel, 1995), *Covariance Matrix Estimation Evolution Strategies* (CMA-ES) (Hansen, Müller, & Koumoutsakos, 2003; Hansen & Ostermeier, 2001; Heidrich-Meisner & Igel, 2009; Igel, 2003), and *NeuroEvolution of Augmenting Topologies* (NEAT) (Stanley & Miikkulainen, 2002). Hybrid methods combine traditional NN-based RL (Section 6.2) and EAs (e.g., Whiteson & Stone, 2006).

Since RNNs are general computers, RNN evolution is like GP in the sense that it can evolve general programs. Unlike sequential programs learned by traditional GP, however, RNNs can mix sequential and parallel information processing in a natural and efficient way, as already mentioned in Section 1. Many RNN evolvers have been proposed (e.g., Cliff, Husbands, & Harvey, 1993; Juang, 2004; Miglino, Lund, & Nolfi, 1995; Miller, Todd, & Hedge, 1989; Moriarty, 1997; Nolfi, Floreano, Miglino, & Mondada, 1994; Pasemann, Steinmetz, & Dieckman, 1999; Sims, 1994; Whiteson, 2012; Wieland, 1991; Yamauchi & Beer, 1994; Yao, 1993). 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 (Gomez, 2003; Gomez & Miikkulainen, 2003; Moriarty & Miikkulainen, 1996). This can help to solve deep POMDPs (Gomez & Schmidhuber, 2005). *Co-Synaptic Neuro-Evolution* (CoSyNE) does something similar on the level of synapses or weights (Gomez, Schmidhuber, & Miikkulainen, 2008); benefits of this were shown on difficult nonlinear POMDP benchmarks.

*Natural Evolution Strategies* (NES) (Glasmachers, Schaul, Sun, Wierstra, & Schmidhuber, 2010; Sun, Gomez, Schaul, & Schmidhuber, 2013; Sun, Wierstra, Schaul, & Schmidhuber, 2009; Wierstra et al., 2008) 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) (Section 5.9).

### 6.7. Deep RL by indirect policy search/compressed NN search

Some DS methods (Section 6.6) can evolve NNs with hundreds or thousands 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 (Sections 5.4, 5.8, 5.16, 5.21), or in RNNs for SL (Sections 5.5, 5.13, 5.17) and RL (Sections 6.1, 6.3, 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 (Section 6.6), one can sometimes greatly reduce the search space by evolving *compact encodings* of NNs, e.g., through *Lindenmeyer Systems* (Jacob, Lindenmayer, & Rozenberg, 1994; Lindenmayer, 1968), *graph rewriting* (Kitano, 1990), *Cellular Encoding* (Gruau, Whitley, & Pyeatt, 1996), *HyperNEAT* (Clune, Stanley, Pennock, & Ofria, 2011; D'Ambrosio & Stanley, 2007; Stanley, D'Ambrosio, & Gauci, 2009; van den Berg & Whiteson, 2013) (extending NEAT; Section 6.6), and extensions thereof (e.g., Risi & Stanley, 2012). This helps to avoid overfitting (compare Sections 5.6.3, 5.24) and is closely related to the topics of regularization and MDL (Section 4.4).

A general approach (Schmidhuber, 1997) for both SL and RL seeks to compactly encode weights of large NNs (Schmidhuber, 1997) through programs written in a *universal programming language* (Church, 1936; Gödel, 1931; Post, 1936; Turing, 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, 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, 1997). 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., 2013; Koutník, Gomez, & Schmidhuber, 2010). Compact DCT-based descriptions can be evolved through NES or CoSyNE (Section 6.6). An RNN with over a million weights learned (without a teacher) to drive a simulated car in the TORCS driving game (Loiacono, Cardamone, & Lanzi, 2011; Loiacono et al., 2009), 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 (Sections 6.4, 4.2) 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, Zhao, & Schraudolph, 1997; Schmidhuber, Zhao, & Wiering, 1997). The most general type of RL is constrained only by the fundamental limitations of computability identified by the founders of theoretical computer science (Church, 1936; Gödel, 1931; Post, 1936; Turing, 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, 2002, 2005; Schmidhuber, 2002, 2006b). In particular, 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 will not further discuss universal RL methods, which go beyond what is usually called DL.

## 7. Conclusion and outlook

*Deep Learning* (DL) in *Neural Networks* (NNs) is relevant for *Supervised Learning* (SL) (Section 5), *Unsupervised Learning* (UL) (Section 5), and *Reinforcement Learning* (RL) (Section 6). By alleviating problems with deep *Credit Assignment Paths* (CAPs, Sections 3, 5.9), UL (Section 5.6.4) cannot only facilitate SL of sequences (Section 5.10) and stationary patterns (Sections 5.7, 5.15), but also RL (Sections 6.4, 4.2). *Dynamic Programming* (DP, Section 4.1) is important for both deep SL (Section 5.5) and traditional RL with deep NNs (Section 6.2). A search for solution-computing, perturbation-resistant (Sections 5.6.3, 5.15, 5.24), low-complexity NNs describable by few bits of information (Section 4.4) can reduce overfitting and improve deep SL & UL (Sections 5.6.3, 5.6.4) as well as RL (Section 6.7), also in the case of partially observable environments (Section 6.3). Deep SL, UL, RL often create hierarchies of more and more abstract representations of stationary data (Sections 5.3, 5.7, 5.15), sequential data (Section 5.10), or RL policies (Section 6.5). While UL can facilitate SL, pure SL for feedforward NNs (FNNs) (Sections 5.5, 5.8, 5.16, 5.18) and recurrent NNs (RNNs) (Sections 5.5, 5.13) did not only win early contests (Sections 5.12, 5.14) but also most of the recent ones (Sections 5.17–5.22). Especially DL in FNNs profited from GPU implementations (Sections 5.16–5.19). In particular, GPU-based (Section 5.19) Max-Pooling (Section 5.11) Convolutional NNs (Sections 5.4, 5.8, 5.16) won competitions not only in pattern recognition (Sections 5.19–5.22) but also image segmentation (Section 5.21) and object detection (Sections 5.21, 5.22).

Unlike these systems, humans *learn to actively perceive* patterns by sequentially directing attention to relevant parts of the available data. Near future deep NNs will do so, too, extending previous work since 1990 on NNs that learn selective attention through RL of (a) *motor actions* such as saccade control (Section 6.1) and (b) *internal actions* controlling spotlights of attention within RNNs, thus closing the general sensorimotor loop through both external and internal feedback (e.g., Sections 2, 5.21, 6.6, 6.7).

Many future deep NNs will also take into account that it costs energy to activate neurons, and to send signals between them. Brains seem to minimize such computational costs during problem solving in at least two ways: (1) At a given time, only a small fraction of all neurons is active because local competition through winner-take-all mechanisms shuts down many neighboring neurons, and only winners can activate other neurons through outgoing connections (compare SLIM NNs; Section 5.24). (2) Numerous neurons are sparsely connected in a compact 3D volume by many short-range and few long-range connections (much like microchips in traditional supercomputers). Often neighboring neurons are allocated to solve a single task, thus reducing communication costs. Physics seems to dictate that any efficient computational hardware will in the future also have to be brain-like

in keeping with these two constraints. The most successful current deep RNNs, however, are not. Unlike certain spiking NNs (Section 5.26), they usually activate all units at least slightly, and tend to be strongly connected, ignoring natural constraints of 3D hardware. It should be possible to improve them by adopting (1) and (2), and by minimizing non-differentiable energy and communication costs through direct search in program (weight) space (e.g., Sections 6.6, 6.7). These more brain-like RNNs will allocate neighboring RNN parts to related behaviors, and distant RNN parts to less related ones, thus self-modularizing in a way more general than that of traditional self-organizing maps in FNNs (Section 5.6.4). They will also implement Occam's razor (Sections 4.4, 5.6.3) as a by-product of energy minimization, by finding simple (highly generalizing) problem solutions that require few active neurons and few, mostly short connections.

The more distant future may belong to general purpose learning algorithms that improve themselves in provably optimal ways (Section 6.8), but these are not yet practical or commercially relevant.

## Acknowledgments

Since 16 April 2014, drafts of this paper have undergone massive open online peer review through public mailing lists including connectionists@cs.cmu.edu, ml-news@googlegroups.com, comp-neuro@neuroinf.org, genetic_programming@yahoogroups.com, rl-list@googlegroups.com, imageworld@diku.dk, *Google+ machine learning forum*. Thanks to numerous NN/DL experts for valuable comments. Thanks to SNF, DFG, and the European Commission for partially funding my DL research group in the past quarter-century. The contents of this paper may be used for educational and non-commercial purposes, including articles for Wikipedia and similar sites.

## References

Aberdeen, D. (2003). *Policy-gradient algorithms for partially observable Markov decision processes* (Ph.D. thesis), Australian National University.
Abounadi, J., Bertsekas, D., & 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. *Annals of the Institute of Statistical Mathematics*, *22*, 203–217.
Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *Second intl. symposium on information theory* (pp. 267–281). Akademinai Kiado.
Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*(6), 716–723.
Allender, A. (1992). Application of time-bounded Kolmogorov complexity in complexity theory. In O. Watanabe (Ed.), *EATCS monographs on theoretical computer science*, *Kolmogorov complexity and computational complexity* (pp. 6–22). 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, vol. 2* (pp. 609–618).
Almeida, L. B., Almeida, L. B., Langlois, T., Amaral, J. D., & 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 Transactions on Electronic Computers*, *16*(3), 299–307.
Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*(2), 251–276.
Amari, S., Cichocki, A., & Yang, H. (1996). A new learning algorithm for blind signal separation. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems (NIPS), vol. 8*. The MIT Press.
Amari, S., & Murata, N. (1993). Statistical theory of learning curves under entropic loss criterion. *Neural Computation*, *5*(1), 140–153.
Amit, D. J., & Brunel, N. (1997). Dynamics of a recurrent network of spiking neurons before and following learning. *Network: Computation in Neural Systems*, *8*(4), 373–404.
An, G. (1996). The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, *8*(3), 643–674.
Andrade, M. A., Chacon, P., Merelo, J. J., & 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.
Andrews, R., Diederich, J., & Tickle, A. B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, *8*(6), 373–389.
Anguita, D., & Gomes, B. A. (1996). Mixing floating- and fixed-point formats for neural network learning on neuroprocessors. *Microprocessing and Microprogramming*, *41*(10), 757–769.
Anguita, D., Parodi, G., & Zunino, R. (1994). An efficient implementation of BP on RISC-based workstations. *Neurocomputing*, *6*(1), 57–65.
Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning—a new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, *5*(4), 13–18.
Ash, T. (1989). Dynamic node creation in backpropagation neural networks. *Connection Science*, *1*(4), 365–375.
Atick, J. J., Li, Z., & Redlich, A. N. (1992). Understanding retinal color coding from first principles. *Neural Computation*, *4*, 559–572.
Atiya, A. F., & Parlos, A. G. (2000). New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE Transactions on Neural Networks*, *11*(3), 697–709.
Ba, J., & Frey, B. (2013). Adaptive dropout for training deep neural networks. In *Advances in neural information processing systems (NIPS)* (pp. 3084–3092).
Baird, H. (1990). Document image defect models. In *Proceddings, IAPR workshop on syntactic and structural pattern recognition*.
Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *International conference on machine learning* (pp. 30–37).
Baird, L., & Moore, A. W. (1999). Gradient descent for general reinforcement learning. In *Advances in neural information processing systems, vol. 12 (NIPS)* (pp. 968–974). MIT Press.
Bakker, B. (2002). Reinforcement learning with long short-term memory. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems, vol. 14* (pp. 1475–1482). Cambridge, MA: MIT Press.
Bakker, B., & Schmidhuber, J. (2004). Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In F. Groen, et al. (Eds.), *Proc. 8th conference on intelligent autonomous systems IAS-8* (pp. 438–445). Amsterdam, NL: IOS Press.
Bakker, B., Zhumatiy, V., Gruener, G., & 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* (pp. 430–435).
Baldi, P. (1995). Gradient descent learning algorithms overview: A general dynamical systems perspective. *IEEE Transactions on Neural Networks*, *6*(1), 182–195.
Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. *Journal of Machine Learning Research*, *27*, 37–50. (Proc. 2011 ICML Workshop on Unsupervised and Transfer Learning).
Baldi, P., Brunak, S., Frasconi, P., Pollastri, G., & Soda, G. (1999). Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, *15*, 937–946.
Baldi, P., & Chauvin, Y. (1993). Neural networks for fingerprint recognition. *Neural Computation*, *5*(3), 402–418.
Baldi, P., & Chauvin, Y. (1996). Hybrid modeling, HMM/NN architectures, and protein applications. *Neural Computation*, *8*(7), 1541–1565.
Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks*, *2*, 53–58.
Baldi, P., & Hornik, K. (1995). Learning in linear networks: a survey. *IEEE Transactions on Neural Networks*, *6*(4), 837–858. 1995.
Baldi, P., & Pollastri, G. (2003). The principled design of large-scale recursive neural network architectures—DAG-RNNs and the protein structure prediction problem. *Journal of Machine Learning Research*, *4*, 575–602.
Baldi, P., & Sadowski, P. (2014). The dropout learning algorithm. *Artificial Intelligence*, *210C*, 78–122.
Ballard, D. H. (1987). Modular learning in neural networks. In *Proc. AAAI* (pp. 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.
Balzer, R. (1985). A 15 year perspective on automatic programming. *IEEE Transactions on Software Engineering*, *11*(11), 1257–1268.
Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, *1*(3), 295–311.
Barlow, H. B., Kaushal, T. P., & 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, vol. IV* (pp. 115–121). IEEE.
Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, *13*(4), 341–379.
Barto, A. G., Singh, S., & Chentanez, N. (2004). Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of international conference on developmental learning* (pp. 112–119). Cambridge, MA: MIT Press.
Barto, A. G., Sutton, R. S., & 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.
Baum, E. B., & Haussler, D. (1989). What size net gives valid generalization? *Neural Computation*, *1*(1), 151–160.
Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 1554–1563.
Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15*(1), 319–350.

Bayer, J., & Osendorfer, C. (2014). Variational inference of latent state sequences using recurrent networks. ArXiv Preprint arXiv:1406.1655.

Bayer, J., Osendorfer, C., Chen, N., Urban, S., & 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., & Schmidhuber, J. (2009). Evolving memory cell structures for sequence learning. In *Proc. ICANN (2)* (pp. 755–764).

Bayes, T. (1763). An essay toward solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, *53*, 370–418. Communicated by R. Price, in a letter to J. Canton.

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

Becker, S., & Le Cun, Y. (1989). Improving the convergence of back-propagation learning with second order methods. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), *Proc. 1988 connectionist models summer school, 1988* (pp. 29–37). San Mateo: Morgan Kaufmann.

Behnke, S. (1999). Hebbian learning and competition in the neural abstraction pyramid. In *Proceedings of the international joint conference on neural networks, vol. 2* (pp. 1356–1361).

Behnke, S. (2001). Learning iterative image reconstruction in the neural abstraction pyramid. *International Journal of Computational Intelligence and Applications*, *1*(4), 427–438.

Behnke, S. (2002). Learning face localization using hierarchical recurrent networks. In *Proceedings of the 12th international conference on artificial neural networks* (pp. 1319–1324).

Behnke, S. (2003a). Discovering hierarchical speech features using convolutional non-negative matrix factorization. In *Proceedings of the international joint conference on neural networks, vol. 4* (pp. 2758–2763).

Behnke, S. (2003b). *LNCS, Lecture notes in computer science*: *Vol. 2766*. *Hierarchical neural networks for image interpretation*. Springer.

Behnke, S. (2005). Face localization and tracking in the neural abstraction pyramid. *Neural Computing and Applications*, *14*(2), 97–103.

Behnke, S., & Rojas, R. (1998). Neural abstraction pyramid: a hierarchical image understanding architecture. In *Proceedings of international joint conference on neural networks, vol. 2* (pp. 820–825).

Bell, A. J., & 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* (1st ed). Princeton, NJ, USA: Princeton University Press.

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

Bengio, Y. (1991). *Artificial neural networks and their application to sequence recognition* (Ph.D. thesis), Montreal, QC, Canada: McGill University, (Computer Science).

Bengio, Y. (2009). *Foundations and trends in machine learning*: *Vol. 2(1). Learning deep architectures for AI*. Now Publishers.

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

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems, vol. 19 (NIPS)* (pp. 153–160). MIT Press.

Bengio, Y., Simard, P., & 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., & Schmidhuber, J. (2005). Classifying unprompted speech by retraining LSTM nets. In W. Duch, J. Kacprzyk, E. Oja, & S. Zadrozny (Eds.), *LNCS*: *Vol. 3696*. *Artificial neural networks: biological inspirations—ICANN 2005* (pp. 575–581). Berlin, Heidelberg: Springer-Verlag.

Bertsekas, D. P. (2001). *Dynamic programming and optimal control*. Athena Scientific.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.

Bichot, N. P., Rossi, A. F., & Desimone, R. (2005). Parallel and serial neural mechanisms for visual search in macaque area V4. *Science*, *308*, 529–534.

Biegler-König, F., & 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., & Pollack, J. B. (1997). Analysis of dynamical recognizers. *Neural Computation*, *9*(5), 1127–1142.

Blondel, V. D., & Tsitsiklis, J. N. (2000). A survey of computational complexity results in systems and control. *Automatica*, *36*(9), 1249–1274.

Bluche, T., Louradour, J., Knibbe, M., Moysset, B., Benzeghiba, F., & Kermorvant, C. (2014). The A2iA Arabic handwritten text recognition system at the OpenHaRT2013 evaluation. In *International workshop on document analysis systems*.

Blum, A. L., & Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. *Neural Networks*, *5*(1), 117–127.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1987). Occam's razor. *Information Processing Letters*, *24*, 377–380.

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

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

Bodenhausen, U., & Waibel, A. (1991). The Tempo 2 algorithm: adjusting time-delays by supervised learning. In D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems, vol. 3* (pp. 155–161). Morgan Kaufmann.

Bohte, S. M., Kok, J. N., & La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*(1), 17–37.

Boltzmann, L. (1909). In F. Hasenöhrl (Ed.), *Wissenschaftliche Abhandlungen*. Leipzig: Barth (collection of Boltzmann's articles in scientific journals).

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

Bourlard, H., & Morgan, N. (1994). *Connnectionist speech recognition: a hybrid approach*. Kluwer Academic Publishers.

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

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

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

Brea, J., Senn, W., & Pfister, J.-P. (2013). Matching recall and storage in sequence learning with spiking neural networks. *The Journal of Neuroscience*, *33*(23), 9565–9575.

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

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., et al. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of Computational Neuroscience*, *23*(3), 349–398.

Breuel, T. M., Ul-Hasan, A., Al-Azawi, M. A., & Shafait, F. (2013). High-performance OCR for printed English and Fraktur using LSTM networks. In *12th International conference on document analysis and recognition* (pp. 683–687). IEEE.

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., et al. (1993). Signature verification using a Siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, *7*(4), 669–688.

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

Brueckner, R., & Schulter, B. (2014). Social signal classification using deep BLSTM recurrent neural networks. In *Proceedings 39th IEEE international conference on acoustics, speech, and signal processing* (pp. 4856–4860).

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of Computational Neuroscience*, *8*(3), 183–208.

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., & Denham, W. F. (1961). *A steepest-ascent method for solving optimum programming problems. Technical report BR-1303*. Raytheon Company, Missle and Space Division.

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

Buhler, J. (2001). Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, *17*(5), 419–428.

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

Burgess, N. (1994). A constructive algorithm that converges for real-valued input patterns. *International Journal of Neural Systems*, *5*(1), 59–66.

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

Carreira-Perpinan, M. A. (2001). *Continuous latent variable models for dimensionality reduction and sequential data reconstruction* (Ph.D. thesis), UK: University of Sheffield.

Carter, M. J., Rudolph, F. J., & Nucci, A. J. (1990). Operational fault tolerance of CMAC networks. In D. S. Touretzky (Ed.), *Advances in neural information processing systems (NIPS), vol. 2* (pp. 340–347). San Mateo, CA: Morgan Kaufmann.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75.

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 D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems, vol. 5* (p. 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., & 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., & Simard, P. (2006). High performance convolutional neural networks for document processing. In *International workshop on Frontiers in handwriting recognition*.

Chen, K., & Salman, A. (2011). Learning speaker-specific characteristics with a deep neural architecture. *IEEE Transactions on Neural Networks*, *22*(11), 1744–1756.

Cho, K. (2014). *Foundations and advances in deep learning* (Ph.D. thesis), Aalto University School of Science.

Cho, K., Ilin, A., & Raiko, T. (2012). Tikhonov-type regularization for restricted Boltzmann machines. In *Intl. conf. on artificial neural networks 2012* (pp. 81–88). Springer.

Cho, K., Raiko, T., & Ilin, A. (2013). Enhanced gradient for training restricted Boltzmann machines. *Neural Computation*, *25*(3), 805–831.

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

Ciresan, D. C., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems (NIPS)* (pp. 2852–2860).

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

Ciresan, D. C., Meier, U., Gambardella, L. M., & 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., & Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *Intl. joint conference on artificial intelligence* (pp. 1237–1242).

Ciresan, D. C., Meier, U., Masci, J., & Schmidhuber, J. (2011). A committee of neural networks for traffic sign classification. In *International joint conference on neural networks (pp. 1918–1921)*.

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

Ciresan, D. C., Meier, U., & Schmidhuber, J. (2012a). Multi-column deep neural networks for image classification. In *IEEE Conference on computer vision and pattern recognition*. Long preprint arXiv:1202.2745v1 [cs.CV].

Ciresan, D. C., Meier, U., & Schmidhuber, J. (2012b). Transfer learning for Latin and Chinese characters with deep neural networks. In *International joint conference on neural networks* (pp. 1301–1306).

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

Cliff, D. T., Husbands, P., & Harvey, I. (1993). Evolving recurrent dynamical networks for robot control. In *Artificial neural nets and genetic algorithms* (pp. 428–435). Springer.

Clune, J., Mouret, J.-B., & Lipson, H. (2013). The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, *280*(1755), 20122863.

Clune, J., Stanley, K. O., Pennock, R. T., & Ofria, C. (2011). On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, *15*(3), 346–367.

Coates, A., Huval, B., Wang, T., Wu, D. J., Ng, A. Y., & Catanzaro, B. (2013). Deep learning with COTS HPC systems. In *Proc. international conference on machine learning*.

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

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167). ACM.

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

Connor, C. E., Brincat, S. L., & Pasupathy, A. (2007). Transformation of shape information in the ventral pathway. *Current Opinion in Neurobiology*, *17*(2), 140–147.

Connor, J., Martin, D. R., & 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* (pp. 151–158). New York: ACM.

Cramer, N. L. (1985). A representation for the adaptive generation of simple sequential programs. In J. Grefenstette (Ed.), *Proceedings of an international conference on genetic algorithms and their applications, Carnegie-Mellon University*. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Cuccu, G., Luciw, M., Schmidhuber, J., & 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, vol. 2* (pp. 1–7). IEEE.

Dahl, G. E., Sainath, T. N., & Hinton, G. E. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. In *IEEE International conference on acoustics, speech and signal processing* (pp. 8609–8613). IEEE.

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

D'Ambrosio, D. B., & 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* (pp. 974–981).

Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on *p*-stable distributions. In *Proceedings of the 20th annual symposium on computational geometry* (pp. 253–262). ACM.

Dayan, P., & Hinton, G. (1993). Feudal reinforcement learning. In D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems (NIPS), vol. 5* (pp. 271–278). Morgan Kaufmann.

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

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

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

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

Deco, G., & Rolls, E. T. (2005). Neurodynamics of biased competition and cooperation for attention: a model with spiking neurons. *Journal of Neurophysiology*, *94*(1), 295–313.

De Freitas, J. F. G. (2003). *Bayesian methods for neural networks* (Ph.D. thesis), University of Cambridge.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: an alternative view. *Machine Learning*, *1*(2), 145–176.

DeMers, D., & Cottrell, G. (1993). Non-linear dimensionality reduction. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems (NIPS), vol. 5* (pp. 580–587). Morgan Kaufmann.

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

Deng, L., & Yu, D. (2014). *Deep learning: methods and applications*. NOW Publishers.

Desimone, R., Albright, T. D., Gross, C. G., & Bruce, C. (1984). Stimulus-selective properties of inferior temporal neurons in the macaque. *The Journal of Neuroscience*, *4*(8), 2051–2062.

de Souto, M. C., Souto, M. C. P. D., & Oliveira, W. R. D. (1999). The loading problem for pyramidal neural networks. *Electronic Journal on Mathematics of Computation*.

De Valois, R. L., Albrecht, D. G., & Thorell, L. G. (1982). Spatial frequency selectivity of cells in macaque visual cortex. *Vision Research*, *22*(5), 545–559.

Deville, Y., & Lau, K. K. (1994). Logic program synthesis. *Journal of Logic Programming*, *19*(20), 321–350.

de Vries, B., & Principe, J. C. (1991). A theory for neural networks with time delays. In R. P. Lippmann, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems (NIPS), vol. 3* (pp. 162–168). Morgan Kaufmann.

DiCarlo, J. J., Zoccolan, D., & Rust, N. C. (2012). How does the brain solve visual object recognition? *Neuron*, *73*(3), 415–434.

Dickmanns, E. D., Behringer, R., Dickmanns, D., Hildebrandt, T., Maurer, M., & Thomanek, F., et al. (1994). The seeing passenger car 'VaMoRs-P'. In *Proc. int. symp. on intelligent vehicles* (pp. 68–73).

Dickmanns, D., Schmidhuber, J., & Winklhofer, A. (1987). *Der genetische algorithmus: eine implementierung in prolog. Technical report*. Inst. of Informatics, Tech. Univ. Munich. http://www.idsia.ch/~juergen/geneticprogramming.html.

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

Dietterich, T. G. (2000b). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research (JAIR)*, *13*, 227–303.

Di Lena, P., Nagata, K., & Baldi, P. (2012). Deep architectures for protein contact map prediction. *Bioinformatics*, *28*, 2449–2457.

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

Dittenbach, M., Merkl, D., & Rauber, A. (2000). The growing hierarchical self-organizing map. In *IEEE-INNS-ENNS International joint conference on neural networks, vol. 6* (p. 6015). IEEE Computer Society.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., & Tzeng, E., et al. (2013). DeCAF: a deep convolutional activation feature for generic visual recognition. ArXiv Preprint arXiv:1310.1531.

Dorffner, G. (1996). Neural networks for time series processing. In *Neural network world*.

Doya, K., Samejima, K., Ichi Katagiri, K., & 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., & 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., et al. (2004). FU-fighters small size 2004, team description. In *RoboCup 2004 symposium: papers and team description papers*. CD edition.

Elfwing, S., Otsuka, M., Uchibe, E., & Doya, K. (2010). Free-energy based reinforcement learning for vision-based navigation with high-dimensional sensory inputs. In *Neural information processing. theory and algorithms (ICONIP), vol. 1* (pp. 215–222). Springer.

Eliasmith, C. (2013). *How to build a brain: a neural architecture for biological cognition*. New York, NY: Oxford University Press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., et al. (2012). A large-scale model of the functioning brain. *Science*, *338*(6111), 1202–1205.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, *11*, 625–660.

Escalante-B, A. N., & Wiskott, L. (2013). How to solve classification and regression problems on high-dimensional data with a supervised extension of slow feature analysis. *Journal of Machine Learning Research*, *14*, 3683–3719.

Eubank, R. L. (1988). Spline smoothing and nonparametric regression. In S. Farlow (Ed.), *Self-organizing methods in modeling*. New York: Marcel Dekker.

Euler, L. (1744). Methodus inveniendi.

Eyben, F., Weninger, F., Squartini, S., & Schuller, B. (2013). Real-life voice activity detection with LSTM recurrent neural networks and an application to Hollywood movies. In *Proc. 38th IEEE international conference on acoustics, speech, and signal processing* (pp. 483–487).

Faggin, F. (1992). Neural network hardware. In *International joint conference on neural networks, vol. 1* (p. 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 R. P. Lippmann, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems (NIPS), vol. 3* (pp. 190–196). Morgan Kaufmann.

Falconbridge, M. S., Stamps, R. L., & Badcock, D. R. (2006). A simple Hebbian/anti-Hebbian network learns the sparse, independent components of natural images. *Neural Computation*, *18*(2), 415–429.

Fan, Y., Qian, Y., Xie, F., & Soong, F. K. (2014). TTS synthesis with bidirectional LSTM based recurrent neural networks. In *Proc. Interspeech*.

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

Farlow, S. J. (1984). *Self-organizing methods in modeling: GMDH type algorithms, vol. 54*. CRC Press.

Feldkamp, L. A., Prokhorov, D. V., Eagen, C. F., & Yuan, F. (1998). Enhanced multi-stream Kalman filter training for recurrent networks. In *Nonlinear modeling* (pp. 29–53). Springer.

Feldkamp, L. A., Prokhorov, D. V., & Feldkamp, T. M. (2003). Simple and conditioned adaptive behavior from Kalman filter trained recurrent networks. *Neural Networks*, *16*(5), 683–689.

Feldkamp, L. A., & Puskorius, G. V. (1998). A signal processing framework based on dynamic neural networks with application to problems in adaptation, filtering, and classification. *Proceedings of the IEEE*, *86*(11), 2259–2277.

Felleman, D. J., & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, *1*(1), 1–47.

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

Fernandez, S., Graves, A., & Schmidhuber, J. (2007b). Sequence labelling in structured domains with hierarchical recurrent neural networks. In *Proceedings of the 20th international joint conference on artificial intelligence*.

Fernandez, R., Rendel, A., Ramabhadran, B., & Hoory, R. (2014). Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks. In *Proc. Interspeech*.

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.

Fieres, J., Schemmel, J., & Meier, K. (2008). Realizing biological spiking network models in a configurable wafer-scale hardware system. In *IEEE International joint conference on neural networks* (pp. 969–976).

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden Markov model: analysis and applications. *Machine Learning*, *32*(1), 41–62.

Fischer, A., & Igel, C. (2014). Training restricted Boltzmann machines: an introduction. *Pattern Recognition*, *47*, 25–39.

FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, *1*(6), 445–466.

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

Floreano, D., & Mattiussi, C. (2001). Evolution of spiking neural controllers for autonomous vision-based robots. In *Evolutionary robotics. From intelligent robotics to artificial life* (pp. 38–61). Springer.

Fogel, D. B., Fogel, L. J., & Porto, V. (1990). Evolving neural networks. *Biological Cybernetics*, *63*(6), 487–493.

Fogel, L., Owens, A., & Walsh, M. (1966). *Artificial intelligence through simulated evolution*. New York: Wiley.

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

Földiák, P., & Young, M. P. (1995). Sparse coding in the primate cortex. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 895–898). The MIT Press.

Förster, A., Graves, A., & Schmidhuber, J. (2007). RNN-based learning of compact maps for efficient robot localization. In *15th European symposium on artificial neural networks* (pp. 537–542).

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

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *Springer series in statistics: Vol. 1. The elements of statistical learning*. New York.

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

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

Fu, K. S. (1977). *Syntactic pattern recognition and applications*. Berlin: Springer.

Fukada, T., Schuster, M., & 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. *Transactions of the 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.

Gabor, D. (1946). Theory of communication. Part 1: the analysis of information. *Electrical Engineers-Part III: Journal of the Institution of Radio and Communication Engineering*, *93*(26), 429–441.

Gallant, S. I. (1988). Connectionist expert systems. *Communications of the ACM*, *31*(2), 152–169.

Gauss, C. F. (1809). Theoria motus corporum coelestium in sectionibus conicis solem ambientium.

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

Ge, S., Hang, C. C., Lee, T. H., & Zhang, T. (2010). *Stable adaptive neural network control*. Springer.

Geiger, J. T., Zhang, Z., Weninger, F., Schuller, B., & Rigoll, G. (2014). Robust speech recognition using long short-term memory recurrent neural networks for hybrid acoustic modelling. In *Proc. interspeech*.

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

Gers, F. A., & Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS international joint conference on neural networks, 2000, vol. 3* (pp. 189–194). IEEE.

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

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

Gerstner, W., & Kistler, W. K. (2002). *Spiking neuron models*. Cambridge University Press.

Gerstner, W., & van Hemmen, J. L. (1992). Associative memory in a network of spiking neurons. *Network: Computation in Neural Systems*, *3*(2), 139–164.

Ghavamzadeh, M., & Mahadevan, S. (2003). Hierarchical policy gradient algorithms. In *Proceedings of the twentieth conference on machine learning* (pp. 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, vol. 1* (pp. 643–644).

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

Gisslen, L., Luciw, M., Graziano, V., & Schmidhuber, J. (2011). Sequential constant size compressor for reinforcement learning. In *Proc. fourth conference on artificial general intelligence* (pp. 31–40). Springer.

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

Glackin, B., McGinnity, T. M., Maguire, L. P., Wu, Q., & Belatreche, A. (2005). A novel approach for the implementation of large scale spiking neural networks on FPGA hardware. In *Computational intelligence and bioinspired systems* (pp. 552–563). Springer.

Glasmachers, T., Schaul, T., Sun, Y., Wierstra, D., & Schmidhuber, J. (2010). Exponential natural evolution strategies. In *Proceedings of the genetic and evolutionary computation conference* (pp. 393–400). ACM.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier networks. In *AISTATS, vol. 15* (pp. 315–323).

Gloye, A., Wiesel, F., Tenchio, O., & 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*. Reading, MA: Addison-Wesley.

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

Golub, G., Heath, H., & 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* (Ph.D. thesis), Department of Computer Sciences, University of Texas at Austin.

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

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

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

Gomi, H., & Kawato, M. (1993). Neural network control for a closed-loop system using feedback-error-learning. *Neural Networks*, *6*(7), 933–946.

Gonzalez-Dominguez, J., Lopez-Moreno, I., Sak, H., Gonzalez-Rodriguez, J., & Moreno, P. J. (2014). Automatic language identification using long short-term memory recurrent neural networks. In *Proc. Interspeech*.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2014). Multi-digit number recognition from street view imagery using deep convolutional neural networks. ArXiv Preprint arXiv:1312.6082v4.

Goodfellow, I. J., Courville, A., & 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., & 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., Mirza, M., Da, X., Courville, A., & Bengio, Y. (2014). *An empirical investigation of catastrophic forgetting in gradient-based neural networks. TR*. arXiv:1312.6211v2.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. In *International conference on machine learning*.

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems (NIPS)* (pp. 2348–2356).

Graves, A., Eck, D., Beringer, N., & Schmidhuber, J. (2003). Isolated digit recognition with LSTM recurrent networks. In *First international workshop on biologically inspired approaches to advanced information technology*.

Graves, A., Fernandez, S., Gomez, F. J., & 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* (pp. 369–376).

Graves, A., Fernandez, S., Liwicki, M., Bunke, H., & Schmidhuber, J. (2008). Unconstrained on-line handwriting recognition with recurrent neural networks. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems (NIPS), vol. 20* (pp. 577–584). Cambridge, MA: MIT Press.

Graves, A., & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proc. 31st International conference on machine learning* (pp. 1764–1772).

Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., & 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., & Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International conference on acoustics, speech and signal processing* (pp. 6645–6649). IEEE.

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

Graves, A., & Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems (NIPS), vol. 21* (pp. 545–552). Cambridge, MA: MIT Press.

Graziano, M. (2009). *The intelligent movement machine: an ethological perspective on the primate motor system*. USA: Oxford University Press.

Griewank, A. (2012). Documenta Mathematica—Extra Volume ISMP, (pp. 389–400).

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

Grünwald, P. D., Myung, I. J., & Pitt, M. A. (2005). *Advances in minimum description length: theory and applications*. MIT Press.

Grüttner, M., Sehnke, F., Schaul, T., & 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* (pp. 114–123). Springer.

Guo, X., Singh, S., Lee, H., Lewis, R., & Wang, X. (2014). Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. In *Advances in neural information processing systems, vol. 27 (NIPS)*.

Guyon, I., Vapnik, V., Boser, B., Bottou, L., & Solla, S. A. (1992). Structural risk minimization for character recognition. In D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems (NIPS), vol. 4* (pp. 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.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Proc. computer vision and pattern recognition conference*. IEEE Press.

Hagras, H., Pounds-Cornish, A., Colley, M., Callaghan, V., & Clarke, G. (2004). Evolving spiking neural network controllers for autonomous robots. In *IEEE International conference on robotics and automation, vol. 5* (pp. 4620–4626).

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

Hanson, S. J. (1990). A stochastic version of the delta rule. *Physica D: Nonlinear Phenomena*, *42*(1), 265–272.

Hanson, S. J., & Pratt, L. Y. (1989). Comparing biases for minimal network construction with back-propagation. In D. S. Touretzky (Ed.), *Advances in neural information processing systems (NIPS), vol. 1* (pp. 177–185). San Mateo, CA: Morgan Kaufmann.

Happel, B. L., & Murre, J. M. (1994). Design and evolution of modular neural network architectures. *Neural Networks*, *7*(6), 985–1004.

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

Hassibi, B., & Stork, D. G. (1993). Second order derivatives for network pruning: optimal brain surgeon. In D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems, vol. 5* (pp. 164–171). Morgan Kaufmann.

Hastie, T. J., & Tibshirani, R. J. (1990). *Monographs on statisics and applied probability*: *Vol. 43. Generalized additive models*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *Springer series in statistics. The elements of statistical learning*.

Hawkins, J., & George, D. (2006). *Hierarchical temporal memory—concepts, theory, and terminology*. Numenta Inc.

Haykin, S. S. (2001). *Kalman filtering and neural networks*. Wiley Online Library.

Hebb, D. O. (1949). *The organization of behavior*. New York: Wiley.

Hecht-Nielsen, R. (1989). Theory of the backpropagation neural network. In *International joint conference on neural networks* (pp. 593–605). IEEE.

Heemskerk, J. N. (1995). Overview of neural hardware. In *Neurocomputers for brain-style processing. Design, implementation and application*.

Heess, N., Silver, D., & Teh, Y. W. (2012). Actor-critic reinforcement learning with energy-based policies. In *Proc. European workshop on reinforcement learning* (pp. 43–57).

Heidrich-Meisner, V., & Igel, C. (2009). Neuroevolution strategies for episodic reinforcement learning. *Journal of Algorithms*, *64*(4), 152–168.

Herrero, J., Valencia, A., & Dopazo, J. (2001). A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics*, *17*(2), 126–136.

Hertz, J., Krogh, A., & Palmer, R. (1991). *Introduction to the theory of neural computation*. Redwood City: Addison-Wesley.

Hestenes, M. R., & 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., & Bengio, Y. (1996). Hierarchical recurrent neural networks for long-term dependencies. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems, vol. 8* (pp. 493–499). MIT Press.

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

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1771–1800.

Hinton, G. E., Dayan, P., Frey, B. J., & 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., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.

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

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

Hinton, G. E., & Sejnowski, T. E. (1986). Learning and relearning in Boltzmann machines. In *Parallel distributed processing, vol. 1* (pp. 282–317). MIT Press.

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

Hinton, G. E., & van Camp, D. (1993). Keeping neural networks simple. In *Proceedings of the international conference on artificial neural networks, Amsterdam* (pp. 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., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer, & J. F. Kolen (Eds.), *A field guide to dynamical recurrent neural networks*. IEEE Press.

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

Hochreiter, S., & Schmidhuber, J. (1996). Bridging long time lags by weight guessing and Long Short-Term Memory. In F. L. Silva, J. C. Principe, & L. B. Almeida (Eds.), *Frontiers in artificial intelligence and applications*: *Vol. 37. Spatiotemporal models in biological and artificial systems* (pp. 65–72). Amsterdam, Netherlands: IOS Press.

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

Hochreiter, S., & Schmidhuber, J. (1997b). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. Based on TR FKI-207-95, TUM (1995).

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

Hochreiter, S., Younger, A. S., & Conwell, P. R. (2001). Learning to learn using gradient descent. In *Lecture notes on comp. sci.: Vol. 2130. Proc. intl. conf. on artificial neural networks* (pp. 87–94). Berlin, Heidelberg: Springer.

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, *117*(4), 500.

Hoerzer, G. M., Legenstein, R., & Maass, W. (2014). Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. *Cerebral Cortex*, *24*, 677–690.

Holden, S. B. (1994). *On the theory of generalization and self-structuring in linearly weighted connectionist networks* (Ph.D. thesis), Cambridge University, Engineering Department.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Honavar, V., & Uhr, L. M. (1988). A network of neuron-like units that learns to perceive by generation as well as reweighting of its links. In D. Touretzky, G. E. Hinton, & T. Sejnowski (Eds.), *Proc. of the 1988 connectionist models summer school* (pp. 472–484). San Mateo: Morgan Kaufman.

Honavar, V., & Uhr, L. (1993). Generative learning structures and processes for generalized connectionist networks. *Information Sciences*, *70*(1), 75–108.

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

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

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

Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, *195*(1), 215–243.

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

Hung, C. P., Kreiman, G., Poggio, T., & DiCarlo, J. J. (2005). Fast readout of object identity from macaque inferior temporal cortex. *Science*, *310*(5749), 863–866.

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*. Berlin: Springer, (On J. Schmidhuber's SNF grant 20-61847).

Hyvärinen, A., Hoyer, P., & Oja, E. (1999). Sparse code shrinkage: denoising by maximum likelihood estimation. In M. Kearns, S. A. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems (NIPS), vol. 12*. MIT Press.

Hyvärinen, A., Karhunen, J., & Oja, E. (2001). *Independent component analysis*. John Wiley & Sons.

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 R. Reynolds, H. Abbass, K. C. Tan, B. Mckay, D. Essam, & T. Gedeon (Eds.), *Congress on evolutionary computation, vol. 4* (pp. 2588–2595). IEEE.

Igel, C., & Hüsken, M. (2003). Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, *50*(C), 105–123.

Ikeda, S., Ochiai, M., & Sawaragi, Y. (1976). Sequential GMDH algorithm and its application to river flow prediction. *IEEE Transactions on Systems, Man and Cybernetics*, (7), 473–479.

Indermuhle, E., Frinken, V., & Bunke, H. (2012). Mode detection in online handwritten documents using BLSTM neural networks. In *Frontiers in handwriting recognition (ICFHR), 2012 international conference on* (pp. 302–307). IEEE.

Indermuhle, E., Frinken, V., Fischer, A., & 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* (pp. 73–77). IEEE.

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, *5*(73).

Ivakhnenko, A. G. (1968). The group method of data handling—a rival of the method of stochastic approximation. *Soviet Automatic Control*, *13*(3), 43–55.

Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems, Man and Cybernetics*, (4), 364–378.

Ivakhnenko, A. G. (1995). The review of problems solvable by algorithms of the group method of data handling (GMDH). *Pattern Recognition and Image Analysis/Raspoznavaniye Obrazov I Analiz Izobrazhenii*, *5*, 527–535.

Ivakhnenko, A. G., & Lapa, V. G. (1965). *Cybernetic predicting devices*. CCM Information Corporation.

Ivakhnenko, A. G., Lapa, V. G., & McDonough, R. N. (1967). *Cybernetics and forecasting techniques*. NY: American Elsevier.

Izhikevich, E. M., et al. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572.

Jaakkola, T., Singh, S. P., & Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), *Advances in neural information processing systems, vol. 7* (pp. 345–352). MIT Press.

Jackel, L., Boser, B., Graf, H.-P., Denker, J., LeCun, Y., & Henderson, D., et al. (1990). VLSI implementation of electronic neural networks: and example in character recognition. In IEEE (Ed.), *IEEE international conference on systems, man, and cybernetics* (pp. 320–322).

Jacob, C., Lindenmayer, A., & Rozenberg, G. (1994). Genetic L-system programming. In *Lecture notes in computer science. Parallel problem solving from nature III*.

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

Jain, V., & Seung, S. (2009). Natural image denoising with convolutional networks. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems (NIPS), vol. 21* (pp. 769–776). Curran Associates, Inc.

Jameson, J. (1991). Delayed reinforcement learning with multiple time scale hierarchical backpropagated adaptive critics. In *Neural networks for control*.

Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(1), 221–231.

Jim, K., Giles, C. L., & Horne, B. G. (1995). Effects of noise on convergence and generalization in recurrent networks. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems (NIPS), vol. 7* (p. 649). San Mateo, CA: Morgan Kaufmann.

Jin, X., Lujan, M., Plana, L. A., Davies, S., Temple, S., & Furber, S. B. (2010). Modeling spiking neural networks on SpiNNaker. *Computing in Science and Engineering*, *12*(5), 91–97.

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

Jones, J. P., & Palmer, L. A. (1987). An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, *58*(6), 1233–1258.

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

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. (1997). Serial order: a parallel distributed processing approach. *Advances in Psychology*, *121*, 471–495.

Jordan, M. I., & 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., & Sejnowski, T. J. (2001). *Graphical models: foundations of neural computation*. MIT Press.

Joseph, R. D. (1961). *Contributions to perceptron theory* (Ph.D. thesis), Cornell Univ.

Juang, C.-F. (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *34*(2), 997–1006.

Judd, J. S. (1990). *Neural network modeling and connectionism. Neural network design and the complexity of learning*. MIT Press.

Jutten, C., & 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., & Cassandra, A. R. (1995). *Planning and acting in partially observable stochastic domains. Technical report*. Providence RI: Brown University.

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

Kak, S., Chen, Y., & Wang, L. (2010). Data mining using surface and deep agents based on neural networks. In *AMCIS 2010 proceedings*.

Kalinke, Y., & Lehmann, H. (1998). Computation in recurrent neural networks: from counters to iterated function systems. In G. Antoniou, & J. Slaney (Eds.), *LNAI: Vol. 1502. Advanced topics in artificial intelligence, Proceedings of the 11th Australian joint conference on artificial intelligence*. Berlin, Heidelberg: Springer.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45.

Karhunen, J., & Joutsensalo, J. (1995). Generalizations of principal component analysis, optimization problems, and neural networks. *Neural Networks*, *8*(4), 549–562.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *IEEE conference on computer vision and pattern recognition*.

Kasabov, N. K. (2014). Neucube: a spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. *Neural Networks*.

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

Kempter, R., Gerstner, W., & Van Hemmen, J. L. (1999). Hebbian learning and spiking neurons. *Physical Review E*, *59*(4), 4498.

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

Khan, S. H., Bennamoun, M., Sohel, F., & Togneri, R. (2014). Automatic feature learning for robust shadow detection. In *IEEE conference on computer vision and pattern recognition*.

Khan, M. M., Khan, G. M., & Miller, J. F. (2010). Evolution of neural networks using Cartesian Genetic Programming. In *IEEE congress on evolutionary computation* (pp. 1–8).

Khan, M. M., Lester, D. R., Plana, L. A., Rast, A., Jin, X., Painkras, E., et al. (2008). SpiNNaker: mapping neural networks onto a massively-parallel chip multiprocessor. In *International joint conference on neural networks* (pp. 2849–2856). IEEE.

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

Kistler, W. M., Gerstner, W., & van Hemmen, J. L. (1997). Reduction of the Hodgkin–Huxley equations to a single-variable threshold model. *Neural Computation*, *9*(5), 1015–1045.

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

Klampfl, S., & Maass, W. (2013). Emergence of dynamic memory traces in cortical microcircuit models through STDP. *The Journal of Neuroscience*, *33*(28), 11515–11529.

Klapper-Rybicka, M., Schraudolph, N. N., & Schmidhuber, J. (2001). Unsupervised learning in LSTM recurrent neural networks. In *Lecture Notes on Comp. Sci.: Vol. 2130. Proc. intl. conf. on artificial neural networks* (pp. 684–691). Berlin, Heidelberg: Springer.

Kobatake, E., & Tanaka, K. (1994). Neuronal selectivities to complex object features in the ventral visual pathway of the macaque cerebral cortex. *Journal of Neurophysiology*, *71*, 856–867.

Kohl, N., & Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and automation, 2004. Proceedings. ICRA'04. 2004 IEEE international conference on, vol. 3* (pp. 2619–2624). IEEE.

Kohonen, T. (1972). Correlation matrix memories. *IEEE Transactions on Computers*, *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* (2nd ed). Springer.

Koikkalainen, P., & Oja, E. (1990). Self-organizing hierarchical feature maps. In *International joint conference on neural networks* (pp. 279–284). IEEE.

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 SSSR*, *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., & Schmidhuber, J. (2012). Incremental slow feature analysis: Adaptive low-complexity slow feature updating from high-dimensional input streams. *Neural Computation*, *24*(11), 2994–3024.

Kondo, T. (1998). GMDH neural network algorithm using the heuristic self-organization method and its application to the pattern identification problem. In *Proceedings of the 37th SICE annual conference* (pp. 1143–1148). IEEE.

Kondo, T., & Ueno, J. (2008). Multi-layered GMDH-type neural network self-selecting optimum neural network architecture and its application to 3-dimensional medical image recognition of blood vessels. *International Journal of Innovative Computing, Information and Control*, *4*(1), 175–187.

Kordík, P., Náplava, P., Snorek, M., & Genyk-Berezovskyj, M. (2003). Modified GMDH method and models quality evaluation by visualization. *Control Systems and Computers*, *2*, 68–75.

Korkin, M., de Garis, H., Gers, F., & 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.

Kosko, B. (1990). Unsupervised learning in noise. *IEEE Transactions on Neural Networks*, *1*(1), 44–57.

Koutník, J., Cuccu, G., Schmidhuber, J., & Gomez, F. (2013). Evolving large-scale neural networks for vision-based reinforcement learning. In *Proceedings of the genetic and evolutionary computation conference* (pp. 1061–1068). Amsterdam: ACM.

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

Koutník, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014). A clockwork RNN. In *Proceedings of the 31th international conference on machine learning, vol. 32* (pp. 1845–1853). arXiv:1402.3511 [cs.NE].

Koza, J. R. (1992). *Genetic programming—on the programming of computers by means of natural selection*. MIT Press.

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

Kremer, S. C., & Kolen, J. F. (2001). *Field guide to dynamical recurrent networks*. Wiley-IEEE Press.

Kriegeskorte, N., Mur, M., Ruff, D. A., Kiani, R., Bodurka, J., Esteky, H., et al. (2008). Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, *60*(6), 1126–1141.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (p. 4).

Krogh, A., & Hertz, J. A. (1992). A simple weight decay can improve generalization. In D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems, vol. 4* (pp. 950–957). Morgan Kaufmann.

Kruger, N., Janssen, P., Kalkan, S., Lappe, M., Leonardis, A., Piater, J., et al. (2013). Deep hierarchies in the primate visual cortex: what can we learn for computer vision? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1847–1871.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 79–86.

Kurzweil, R. (2012). *How to create a mind: the secret of human thought revealed*.

Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4*, 1107–1149.

Lampinen, J., & Oja, E. (1992). Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, *2*(2–3), 261–272.

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

Lange, S., & Riedmiller, M. (2010). Deep auto-encoder neural networks in reinforcement learning. In *Neural networks, The 2010 international joint conference on* (pp. 1–8).

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

Laplace, P. (1774). Mémoire sur la probabilité des causes par les évènements. *Mémoires de l'Academie Royale des Sciences Presentés par Divers Savan*, *6*, 621–656.

Larraanaga, P., & Lozano, J. A. (2001). *Estimation of distribution algorithms: a new tool for evolutionary computation*. Norwell, MA, USA: Kluwer Academic Publishers.

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Corrado, G., & Chen, K., et al. (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. In *Proceedings of cognitiva 85* (pp. 599–604).

LeCun, Y. (1988). A theoretical framework for back-propagation. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), *Proceedings of the 1988 connectionist models summer school* (pp. 21–28). CMU, Pittsburgh, Pa: Morgan Kaufmann.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et al. (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., et al. (1990). Handwritten digit recognition with a back-propagation network. In D. S. Touretzky (Ed.), *Advances in neural information processing systems, vol. 2* (pp. 396–404). Morgan Kaufmann.

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

LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In D. S. Touretzky (Ed.), *Advances in neural information processing systems, vol. 2* (pp. 598–605). Morgan Kaufmann.

LeCun, Y., Muller, U., Cosatto, E., & 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., & Pearlmutter, B. (1993). Automatic learning rate maximization by on-line estimation of the Hessian's eigenvectors. In S. Hanson, J. Cowan, & L. Giles (Eds.), *Advances in neural information processing systems, vol. 5 (NIPS 1992)*. San Mateo, CA: Morgan Kaufmann Publishers.

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

Lee, H., Battle, A., Raina, R., & Ng, A. Y. (2007). Efficient sparse coding algorithms. In *Advances in neural information processing systems (NIPS), vol. 19* (pp. 801–808).

Lee, H., Ekanadham, C., & Ng, A. Y. (2007). Sparse deep belief net model for visual area V2. In *Advances in neural information processing systems (NIPS), vol. 7* (pp. 873–880).

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th international conference on machine learning* (pp. 609–616).

Lee, S., & Kil, R. M. (1991). A Gaussian potential function network with hierarchically self-organizing learning. *Neural Networks*, *4*(2), 207–224.

Lee, H., Pham, P. T., Largman, Y., & Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Proc. NIPS, vol. 9* (pp. 1096–1104).

Legendre, A. M. (1805). *Nouvelles méthodes pour la détermination des orbites des cometes*. F. Didot.

Legenstein, R. A., & Maass, W. (2002). Neural circuits for pattern recognition with small total wire length. *Theoretical Computer Science*, *287*(1), 239–249.

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

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

Leibniz, G. W. (1684). Nova methodus pro maximis et minimis, itemque tangentibus, quae nec fractas, nec irrationales quantitates moratur, et singulare pro illis calculi genus. *Acta Eruditorum*, 467–473.

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

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

Lennie, P., & Movshon, J. A. (2005). Coding of color and form in the geniculostriate visual pathway. *Journal of the Optical Society of America A*, *22*(10), 2013–2033.

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

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

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

Levin, A. U., Leen, T. K., & Moody, J. E. (1994). Fast pruning using principal components. In *Advances in neural information processing systems (NIPS), vol. 6* (p. 35). Morgan Kaufmann.

Levin, A. U., & Narendra, K. S. (1995). Control of nonlinear dynamical systems using neural networks. II. Observability, identification, and control. *IEEE Transactions on Neural Networks*, *7*(1), 30–42.

Lewicki, M. S., & Olshausen, B. A. (1998). Inferring sparse, overcomplete image codes using an efficient coding framework. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in neural information processing systems (NIPS), vol. 10* (pp. 815–821).

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

Li, M., & Vitányi, P. M. B. (1997). *An introduction to Kolmogorov complexity and its applications* (2nd ed.). Springer.

Li, R., Zhang, W., Suk, H.-I., Wang, L., Li, J., Shen, D., et al. (2014). Deep learning based imaging data completion for improved brain disease diagnosis. In *Proc. MICCAI*. Springer.

Lin, L. (1993). *Reinforcement learning for robots using neural networks* (Ph.D. thesis), Pittsburgh: Carnegie Mellon University.

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

Lindenmayer, A. (1968). Mathematical models for cellular interaction in development. *Journal of Theoretical Biology*, *18*, 280–315.

Lindstädt, S. (1993). Comparison of two unsupervised neural network models for redundancy reduction. In M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, & A. S. Weigend (Eds.), *Proc. of the 1993 connectionist models summer school* (pp. 308–315). Hillsdale, NJ: Erlbaum Associates.

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., Cassandra, A. R., & Kaelbling, L. P. (1995). Learning policies for partially observable environments: scaling up. In A. Prieditis, & S. Russell (Eds.), *Machine learning: proceedings of the twelfth international conference* (pp. 362–370). San Francisco, CA: Morgan Kaufmann Publishers.

Liu, S.-C., Kramer, J., Indiveri, G., Delbrück, T., Burg, T., Douglas, R., et al. (2001). Orientation-selective aVLSI spiking neurons. *Neural Networks*, *14*(6–7), 629–643.

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

Logothetis, N. K., Pauls, J., & Poggio, T. (1995). Shape representation in the inferior temporal cortex of monkeys. *Current Biology*, *5*(5), 552–563.

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

Loiacono, D., Lanzi, P. L., Togelius, J., Onieva, E., Pelta, D. A., & Butz, M. V., et al. (2009). The 2009 simulated car racing championship.

Lowe, D. (1999). Object recognition from local scale-invariant features. In *The Proceedings of the seventh IEEE international conference on computer vision, vol. 2* (pp. 1150–1157).

Lowe, D. (2004). Distinctive image features from scale-invariant key-points. *International Journal of Computer Vision*, *60*, 91–110.

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

Lusci, A., Pollastri, G., & Baldi, P. (2013). Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *Journal of Chemical Information and Modeling*, *53*(7), 1563–1575.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *International conference on machine learning*.

Maass, W. (1996). Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, *8*(1), 1–40.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, *10*(9), 1659–1671.

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

Maass, W., Natschläger, T., & 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., & Miller, K. D. (1990). Analysis of Linsker's simulation of Hebbian rules. *Neural Computation*, *2*, 173–187.

Maclin, R., & Shavlik, J. W. (1993). Using knowledge-based neural networks to improve algorithms: Refining the Chou–Fasman algorithm for protein folding. *Machine Learning*, *11*(2–3), 195–215.

Maclin, R., & Shavlik, J. W. (1995). Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks. In *Proc. IJCAI* (pp. 524–531).

Madala, H. R., & Ivakhnenko, A. G. (1994). *Inductive learning algorithms for complex systems modeling*. Boca Raton: CRC Press.

Madani, O., Hanks, S., & Condon, A. (2003). On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, *147*(1), 5–34.

Maei, H. R., & Sutton, R. S. (2010). GQ($\lambda$): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the third conference on artificial general intelligence, vol. 1* (pp. 91–96).

Maex, R., & Orban, G. (1996). Model circuit of spiking neurons generating directional selectivity in simple cells. *Journal of Neurophysiology*, *75*(4), 1515–1545.

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

Malik, J., & Perona, P. (1990). Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America A*, *7*(5), 923–932.

Maniezzo, V. (1994). Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks*, *5*(1), 39–53.

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

Marchi, E., Ferroni, G., Eyben, F., Gabrielli, L., Squartini, S., & Schuller, B. (2014). Multi-resolution linear prediction based features for audio onset detection with bidirectional LSTM neural networks. In *Proc. 39th IEEE international conference on acoustics, speech, and signal processing* (pp. 2183–2187).

Markram, H. (2012). The human brain project. *Scientific American*, *306*(6), 50–55.

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 J. Fürnkranz, & T. Joachims (Eds.), *Proceedings of the 27th international conference on machine learning* (pp. 735–742). Haifa, Israel: OmniPress.

Martens, J., & Sutskever, I. (2011). Learning recurrent neural networks with Hessian-free optimization. In *Proceedings of the 28th international conference on machine learning* (pp. 1033–1040).

Martinetz, T. M., Ritter, H. J., & 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., Giusti, A., Ciresan, D. C., Fricout, G., & Schmidhuber, J. (2013). A fast learning algorithm for image segmentation with max-pooling convolutional networks. In *International conference on image processing (pp. 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., & 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 P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, & S. W. Wilson (Eds.), *From animals to animats 4: proceedings of the fourth international conference on simulation of adaptive behavior* (pp. 315–324). MIT Press, Bradford Books.

McCulloch, W., & 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., & Pollack, J. B. (2000). RAAM for infinite context-free languages. In *Proc. IJCNN (5)* (pp. 585–590).

Memisevic, R., & Hinton, G. E. (2010). Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation*, *22*(6), 1473–1492.

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

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, *345*(6197), 668–673.

Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., & Goodfellow, I., et al. (2011). Unsupervised and transfer learning challenge: a deep learning approach. In *JMLR W&CP: proc. unsupervised and transfer learning, vol. 7*.

Meuleau, N., Peshkin, L., Kim, K. E., & Kaelbling, L. P. (1999). Learning finite state controllers for partially observable environments. In *15th international conference of uncertainty in AI* (pp. 427–436).

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

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.

Miller, J. F., & Harding, S. L. (2009). Cartesian genetic programming. In *Proceedings of the 11th annual conference companion on genetic and evolutionary computation conference: late breaking papers* (pp. 3489–3512). ACM.

Miller, J. F., & Thomson, P. (2000). Cartesian genetic programming. In *Genetic programming* (pp. 121–132). Springer.

Miller, G., Todd, P., & Hedge, S. (1989). Designing neural networks using genetic algorithms. In *Proceedings of the 3rd international conference on genetic algorithms* (pp. 379–384). Morgan Kauffman.

Miller, W. T., Werbos, P. J., & Sutton, R. S. (1995). *Neural networks for control*. MIT Press.

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

Minsky, M. (1963). Steps toward artificial intelligence. In E. Feigenbaum, & J. Feldman (Eds.), *Computers and thought* (pp. 406–450). New York: McGraw-Hill.

Minsky, M., & Papert, S. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., & Gil, Y. (1989). Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, *40*(1), 63–118.

Mitchell, T. (1997). *Machine learning*. McGraw Hill.

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, *1*(1), 47–80.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). *Playing Atari with deep reinforcement learning. Technical report*. Deepmind Technologies, arXiv:1312.5602 [cs.LG].

Mohamed, A., & Hinton, G. E. (2010). Phone recognition using restricted Boltzmann machines. In *IEEE international conference on acoustics, speech and signal processing* (pp. 4354–4357).

Molgedey, L., & Schuster, H. G. (1994). Separation of independent signals using time-delayed correlations. *Physical Review 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*. Denmark: Computer Science Department, Aarhus University.

Montana, D. J., & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th international joint conference on artificial intelligence—vol. 1* (pp. 762–767). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Montavon, G., Orr, G., & Müller, K. (2012). *Lecture Notes in Computer Science Series. LNCS: Vol. 7700. Neural networks: tricks of the trade*. Springer Verlag.

Moody, J. E. (1989). Fast learning in multi-resolution hierarchies. In D. S. Touretzky (Ed.), *Advances in neural information processing systems (NIPS), vol. 1* (pp. 29–39). Morgan Kaufmann.

Moody, J. E. (1992). The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems (NIPS), vol. 4* (pp. 847–854). Morgan Kaufmann.

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

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

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

Moriarty, D. E. (1997). *Symbiotic evolution of neural networks in sequential decision tasks* (Ph.D. thesis), Department of Computer Sciences, The University of Texas at Austin.

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

Morimoto, J., & Doya, K. (2000). Robust reinforcement learning. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems (NIPS), vol. 13* (pp. 1061–1067). MIT Press.

Mosteller, F., & Tukey, J. W. (1968). Data analysis, including statistics. In G. Lindzey, & E. Aronson (Eds.), *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 R. P. Lippmann, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems, vol. 3* (pp. 627–634). Morgan Kaufmann.

Mozer, M. C. (1992). Induction of multiscale temporal structure. In D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems (NIPS), vol. 4* (pp. 275–282). Morgan Kaufmann.

Mozer, M. C., & Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. S. Touretzky (Ed.), *Advances in neural information processing systems (NIPS), vol. 1* (pp. 107–115). Morgan Kaufmann.

Muller, U. A., Gunzinger, A., & Guggenbühl, W. (1995). Fast neural net simulation with a DSP processor array. *IEEE Transactions on Neural Networks*, *6*(1), 203–213.

Munro, P. W. (1987). A dual back-propagation scheme for scalar reinforcement learning. In *Proceedings of the ninth annual conference of the cognitive science society* (pp. 165–176).

Murray, A. F., & Edwards, P. J. (1993). Synaptic weight noise during MLP learning enhances fault-tolerance, generalisation and learning trajectory. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems (NIPS), vol. 5* (pp. 491–498). San Mateo, CA: Morgan Kaufmann.

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

Nagumo, J., Arimoto, S., & Yoshizawa, S. (1962). An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, *50*(10), 2061–2070.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *International conference on machine learning*.

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

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

Neal, R. M. (1995). *Bayesian learning for neural networks* (Ph.D. thesis), University of Toronto.

Neal, R. M. (2006). Classification with Bayesian neural networks. In J. Quinonero-Candela, B. Magnini, I. Dagan, & F. D'Alche-Buc (Eds.), *Lecture notes in computer science: Vol. 3944. Machine learning challenges. Evaluating predictive uncertainty, visual object classification, and recognising textual entailment* (pp. 28–32). Springer.

Neal, R. M., & Zhang, J. (2006). High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees. In I. Guyon, S. Gunn, M. Nikravesh, & L. A. Zadeh (Eds.), *Studies in fuzziness and soft computing, Feature extraction: foundations and applications* (pp. 265–295). Springer.

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., & Cauwenberghs, G. (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, *7*(272).

Neil, D., & Liu, S.-C. (2014). Minitaur, an event-driven FPGA-based spiking network accelerator. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *PP*(99), 1–8.

Nessler, B., Pfeiffer, M., Buesing, L., & Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Computational Biology*, *9*(4), e1003037.

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

Neuneier, R., & Zimmermann, H.-G. (1996). How to train neural networks. In G. B. Orr, & K.-R. Müller (Eds.), *Lecture notes in computer science: Vol. 1524. Neural networks: tricks of the trade* (pp. 373–423). Springer.

Newton, I. (1687). *Philosophiae naturalis principia mathematica*. London: William Dawson & Sons Ltd.

Nguyen, N., & 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* (pp. 357–363). IEEE Press.

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

Nolfi, S., Floreano, D., Miglino, O., & Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. In R. A. Brooks, & P. Maes (Eds.), *Fourth international workshop on the synthesis and simulation of living systems (artificial life IV)* (pp. 190–197). MIT.

Nolfi, S., Parisi, D., & Elman, J. L. (1994). Learning and evolution in neural networks. *Adaptive Behavior*, *3*(1), 5–28.

Nowak, E., Jurie, F., & Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *Proc. ECCV 2006* (pp. 490–503). Springer.

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

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., & Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, *7*(178).

Oh, K.-S., & 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 T. Kohonen, K. Mäkisara, O. Simula, & J. Kangas (Eds.), *Artificial neural networks, vol. 1* (pp. 737–745). North-Holland: Elsevier Science Publishers BV.

Olshausen, B. A., & 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., & Giles, C. L. (1996). Extraction of rules from discrete-time recurrent neural networks. *Neural Networks*, *9*(1), 41–52.

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2013). *Learning and transferring mid-level image representations using convolutional neural networks. Technical report hal-00911179*.

O'Reilly, R. C. (1996). Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, *8*(5), 895–938.

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.

O'Reilly, R. C., Wyatte, D., Herd, S., Mingus, B., & Jilk, D. J. (2013). Recurrent processing during object recognition. *Frontiers in Psychology*, *4*, 124.

Orr, G., & Müller, K. (1998). *Lecture Notes in Computer Science Series. LNCS: Vol. 1524. Neural networks: tricks of the trade*. Springer Verlag.

Ostrovskii, G. M., Volin, Y. M., & Borisov, W. W. (1971). Über die Berechnung von Ableitungen. *Wissenschaftliche Zeitschrift der Technischen Hochschule für Chemie*, *13*, 382–384.

Otsuka, M. (2010). *Goal-oriented representation of the external world: a free-energy-based approach* (Ph.D. thesis), Nara Institute of Science and Technology.

Otsuka, M., Yoshimoto, J., & Doya, K. (2010). Free-energy-based reinforcement learning in a partially observable environment. In *Proc. ESANN*.

Otte, S., Krechel, D., Liwicki, M., & 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* (pp. 533–537). IEEE Computer Society.

Oudeyer, P.-Y., Baranes, A., & Kaplan, F. (2013). Intrinsically motivated learning of real world sensorimotor skills with developmental constraints. In G. Baldassarre, & M. Mirolli (Eds.), *Intrinsically motivated learning in natural and artificial systems*. Springer.

Pachitariu, M., & 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.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *The IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Parekh, R., Yang, J., & Honavar, V. (2000). Constructive neural network learning algorithms for multi-category pattern classification. *IEEE Transactions on Neural Networks*, *11*(2), 436–451.

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., & Bengio, Y. (2013). How to construct deep recurrent neural networks. arXiv Preprint arXiv:1312.6026.

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

Pasemann, F., Steinmetz, U., & Dieckman, U. (1999). Evolving structure and function of neurocontrollers. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, & A. Zalzala (Eds.), *Proceedings of the congress on evolutionary computation, vol. 3* (pp. 1973–1978). Mayflower Hotel, Washington, DC, 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., & Hinton, G. E. (1986). G-maximization: An unsupervised learning procedure for discovering regularities. In Denker, J.S., (Ed.), *Neural networks for computing: American institute of physics conference proceedings 151, vol. 2* (pp. 333–338).

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

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

Perrett, D., Hietanen, J., Oram, M., Benson, P., & Rolls, E. (1992). Organization and functions of cells responsive to faces in the temporal cortex [and discussion]. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, *335*(1273), 23–30.

Perrett, D., Rolls, E., & Caan, W. (1982). Visual neurones responsive to faces in the monkey temporal cortex. *Experimental Brain Research*, *47*(3), 329–342.

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

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

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

Pham, V., Kermorvant, C., & 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. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems (NIPS), vol. 5* (pp. 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* (pp. 527–536).

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

Pontryagin, L. S., Boltyanskii, V. G., Gamrelidze, R. V., & Mishchenko, E. F. (1961). *The mathematical theory of optimal processes*.

Poon, H., & Domingos, P. (2011). Sum–product networks: A new deep architecture. In *IEEE International conference on computer vision workshops* (pp. 689–690). IEEE.

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

Prasoon, A., Petersen, K., Igel, C., Lauze, F., Dam, E., & Nielsen, M. (2013). Voxel classification based on triplanar convolutional neural networks applied to cartilage segmentation in knee MRI. In *LNCS*: *Vol. 8150. Medical image computing and computer assisted intervention (MICCAI)* (pp. 246–253). Springer.

Precup, D., Sutton, R. S., & Singh, S. (1998). Multi-time models for temporally abstract planning. In *Advances in neural information processing systems (NIPS)* (pp. 1050–1056). Morgan Kaufmann.

Prokhorov, D. (2010). A convolutional learning system for object classification in 3-D LIDAR data. *IEEE Transactions on Neural Networks*, *21*(5), 858–863.

Prokhorov, D. V., Feldkamp, L. A., & Tyukin, I. Y. (2002). Adaptive behavior with fixed weights in RNN: an overview. In *Proceedings of the IEEE international joint conference on neural networks* (pp. 2018–2023).

Prokhorov, D., Puskorius, G., & Feldkamp, L. (2001). Dynamical neural networks for control. In J. Kolen, & S. Kremer (Eds.), *A field guide to dynamical recurrent networks* (pp. 23–78). IEEE Press.

Prokhorov, D., & Wunsch, D. (1997). Adaptive critic design. *IEEE Transactions on Neural Networks*, *8*(5), 997–1007.

Puskorius, G. V., & Feldkamp, L. A. (1994). Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. *IEEE Transactions on Neural Networks*, *5*(2), 279–297.

Raiko, T., Valpola, H., & LeCun, Y. (2012). Deep learning made easier by linear transformations in perceptrons. In *International conference on artificial intelligence and statistics* (pp. 924–932).

Raina, R., Madhavan, A., & Ng, A. (2009). Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual International conference on machine learning* (pp. 873–880). ACM.

Ramacher, U., Raab, W., Anlauf, J., Hachmann, U., Beichter, J., Bruels, N., et al. (1993). Multiprocessor and memory architecture of the neurocomputer SYNAPSE-1. *International Journal of Neural Systems*, *4*(4), 333–336.

Ranzato, M. A., Huang, F., Boureau, Y., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. computer vision and pattern recognition conference* (pp. 1–8). IEEE Press.

Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In J. Platt, et al. (Eds.), *Advances in neural information processing systems (NIPS 2006)*. MIT Press.

Rauber, A., Merkl, D., & Dittenbach, M. (2002). The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, *13*(6), 1331–1341.

Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. ArXiv Preprint arXiv:1403.6382.

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.

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

Rezende, D. J., & Gerstner, W. (2014). Stochastic variational learning in recurrent spiking networks. *Frontiers in Computational Neuroscience*, *8*, 38.

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

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

Riedmiller, M., Lange, S., & Voigtlaender, A. (2012). Autonomous reinforcement learning on raw visual input data in a real world application. In *International joint conference on neural networks* (pp. 1–8).

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*(11), 1019–1025.

Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning* (pp. 833–840).

Ring, M. B. (1991). Incremental development of complex behaviors through automatic construction of sensory-motor hierarchies. In L. Birnbaum, & G. Collins (Eds.), *Machine learning: proceedings of the eighth international workshop* (pp. 343–347). Morgan Kaufmann.

Ring, M. B. (1993). Learning sequential tasks by incrementally adding higher orders. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems, vol. 5* (pp. 115–122). Morgan Kaufmann.

Ring, M. B. (1994). *Continual learning in reinforcement environments* (Ph.D. thesis), Austin, Texas 78712: University of Texas at Austin.

Ring, M., Schaul, T., & Schmidhuber, J. (2011). The two-dimensional organization of behavior. In *Proceedings of the first joint conference on development learning and on epigenetic robotics*.

Risi, S., & Stanley, K. O. (2012). A unified approach to evolving plasticity and neural geometry. In *International joint conference on neural networks* (pp. 1–8). IEEE.

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

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

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

Robinson, T., & 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* (pp. 836–843).

Rodriguez, P., & Wiles, J. (1998). Recurrent neural networks can learn to implement symbol-sensitive counting. In *Advances in neural information processing systems (NIPS), vol. 10* (pp. 87–93). The MIT Press.

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

Roggen, D., Hofmann, S., Thoma, Y., & Floreano, D. (2003). Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot. In *Proc. NASA/DoD conference on evolvable hardware* (pp. 189–198). IEEE.

Rohwer, R. (1989). The 'moving targets' training method. In J. Kindermann, & A. Linden (Eds.), *Proceedings of 'distributed adaptive neural information processing'*. 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*. New York: Spartan.

Roux, L., Racoceanu, D., Lomenie, N., Kulikova, M., Irshad, H., Klossa, J., et al. (2013). Mitosis detection in breast cancer histological images—an ICPR 2012 contest. *Journal of Pathology Informatics*, *4*, 8.

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

Rückstieß, T., Felder, M., & Schmidhuber, J. (2008). State-dependent exploration for policy gradient methods. In W. Daelemans, et al. (Eds.), *LNAI*: *Vol. 5212. European conference on machine learning (ECML) and principles and practice of knowledge discovery in databases 2008, part II* (pp. 234–249).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing, vol. 1* (pp. 318–362). MIT Press.

Rumelhart, D. E., & Zipser, D. (1986). Feature discovery by competitive learning. In *Parallel distributed processing* (pp. 151–193). MIT Press.

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

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., & Edwards, D. D. (1995). *Artificial intelligence: a modern approach, vol. 2*. Englewood Cliffs: Prentice Hall.

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

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proc. interspeech*.

Sak, H., Vinyals, O., Heigold, G., Senior, A., McDermott, E., & Monga, R., et al. (2014). Sequence discriminative distributed training of long short-term memory recurrent neural networks. In *Proc. Interspeech*.

Salakhutdinov, R., & Hinton, G. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, *50*(7), 969–978.

Sallans, B., & Hinton, G. (2004). Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, *5*, 1063–1088.

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

Samejima, K., Doya, K., & 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 of Research and Development*, *3*, 210–229.

Sanger, T. D. (1989). An optimality principle for unsupervised learning. In D. S. Touretzky (Ed.), *Advances in neural information processing systems (NIPS), vol. 1* (pp. 11–19). Morgan Kaufmann.

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

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

Saund, E. (1994). Unsupervised learning of mixtures of multiple causes in binary data. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems (NIPS), vol. 6* (pp. 27–34). Morgan Kaufmann.

Schaback, R., & Werner, H. (1992). *Numerische mathematik, vol. 4*. Springer.

Schäfer, A. M., Udluft, S., & Zimmermann, H.-G. (2006). Learning long term dependencies with recurrent neural networks. In S. D. Kollias, A. Stafylopatis, W. Duch, & E. Oja (Eds.), *Lecture notes in computer science*: *Vol. 4131. ICANN (1)* (pp. 71–80). Springer.

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

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

Schaul, T., Zhang, S., & LeCun, Y. (2013). No more pesky learning rates. In *Proc. 30th International conference on machine learning*.

Schemmel, J., Grubl, A., Meier, K., & Mueller, E. (2006). Implementing synaptic plasticity in a VLSI spiking neural network model. In *International joint conference on neural networks* (pp. 1–6). IEEE.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Proc. International conference on artificial neural networks* (pp. 92–101).

Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook* (Diploma thesis), Inst. f. Inf., Tech. Univ. Munich, http://www.idsia.ch/~juergen/diploma.html.

Schmidhuber, J. (1989a). Accelerated learning in back-propagation nets. In R. Pfeifer, Z. Schreter, Z. Fogelman, & L. Steels (Eds.), *Connectionism in perspective* (pp. 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. (Dynamic neural nets and the fundamental spatio-temporal credit assignment problem.)* (Dissertation), Inst. f. Inf., Tech. Univ. Munich.

Schmidhuber, J. (1990b). Learning algorithms for networks with internal and external feedback. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, & G. E. Hinton (Eds.), *Proc. of the 1990 connectionist models summer school* (pp. 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*, vol. 1 (pp. 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, vol. 2* (pp. 253–258).

Schmidhuber, J. (1991a). Curious model-building control systems. In *Proceedings of the international joint conference on neural networks, vol. 2* (pp. 1458–1463). IEEE Press.

Schmidhuber, J. (1991b). Learning to generate sub-goals for action sequences. In T. Kohonen, K. Mäkisara, O. Simula, & J. Kangas (Eds.), *Artificial neural networks* (pp. 967–972). North-Holland: Elsevier Science Publishers BV.

Schmidhuber, J. (1991c). Reinforcement learning in Markovian and non-Markovian environments. In D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems, vol. 3 (NIPS 3)* (pp. 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* (pp. 191–195). IEE.

Schmidhuber, J. (1993b). *Netzwerkarchitekturen, Zielfunktionen und Kettenregel. (Network architectures, objective functions, and chain rule.)* (Habilitation thesis), Inst. f. Inf., Tech. Univ. Munich.

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 J. Kivinen, & R. H. Sloan (Eds.), *Lecture notes in artificial intelligence, Proceedings of the 15th annual conference on computational learning theory* (pp. 216–228). Sydney, Australia: Springer.

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 B. Goertzel, & C. Pennachin (Eds.), *Artificial general intelligence* (pp. 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*. The Swiss AI Lab IDSIA, arXiv:1210.0118v1 [cs.NE].

Schmidhuber, J. (2013a). *My first deep learning system of 1991 + deep learning timeline 1962–2013. Technical report*. The Swiss AI Lab IDSIA, arXiv:1312.5548v1 [cs.NE].

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., & Graves, A. (2011). On fast deep nets for AGI vision. In *Proc. fourth conference on artificial general intelligence* (pp. 243–246).

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

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

Schmidhuber, J., Mozer, M. C., & Prelinger, D. (1993). Continuous history compression. In H. Hüning, S. Neuhauser, M. Raus, & W. Ritschel (Eds.), *Proc. of intl. workshop on neural networks* (pp. 87–95). Augustinus: RWTH Aachen.

Schmidhuber, J., & Prelinger, D. (1992). *Discovering predictable classifications. Technical report CU-CS-626-92*. Dept. of Comp. Sci., University of Colorado at Boulder. Published in Neural Computation 5 (4) (1993) 625–635.

Schmidhuber, J., & Wahnsiedler, R. (1992). Planning simple trajectories using neural subgoal generators. In J. A. Meyer, H. L. Roitblat, & S. W. Wilson (Eds.), *Proc. of the 2nd international conference on simulation of adaptive behavior* (pp. 196–202). MIT Press.

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

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

Schmidhuber, J., Zhao, J., & Wiering, M. (1997). 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., & Smola, A. J. (Eds.) (1998). *Advances in kernel methods—support vector learning*. Cambridge, MA: MIT Press.

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

Schraudolph, N., & Sejnowski, T. J. (1993). Unsupervised discrimination of clustered data via optimization of binary information gain. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems, vol. 5* (pp. 499–506). San Mateo: Morgan Kaufmann.

Schraudolph, N. N., & Sejnowski, T. J. (1996). Tempering backpropagation networks: not all weights are created equal. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems (NIPS), vol. 8* (pp. 563–569). Cambridge, MA: The MIT Press.

Schrauwen, B., Verstraeten, D., & Van Campenhout, J. (2007). An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European symposium on artificial neural networks* (pp. 471–482).

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

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

Schuster, M., & 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* (pp. 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*. http://tinyurl.com/d2fgh7g.

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

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). OverFeat: integrated recognition, localization and detection using convolutional networks. ArXiv Preprint arXiv:1312.6229.

Sermanet, P., & LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of international joint conference on neural networks* (pp. 2809–2813).

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gómez-Rodríguez, F., et al. (2009). Caviar: A 45 k neuron, 5 m synapse, 12 g connects/s AER hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking. *IEEE Transactions on Neural Networks*, *20*(9), 1417–1438.

Serre, T., Riesenhuber, M., Louie, J., & Poggio, T. (2002). On the role of object-specific features for real world object recognition in biological vision. In *Biologically motivated computer vision* (pp. 387–397).

Seung, H. S. (2003). Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, *40*(6), 1063–1073.

Shan, H., & Cottrell, G. (2014). Efficient visual coding: From retina to V2. In *Proc. international conference on learning representations*. ArXiv Preprint arXiv:1312.6077.

Shan, H., Zhang, L., & Cottrell, G. W. (2007). Recursive ICA. In *Advances in neural information processing systems (NIPS), vol. 19* (p. 1273).

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.

Shao, L., Wu, D., & Li, X. (2014). Learning deep and wide: A spectral method for learning deep networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Shavlik, J. W. (1994). Combining symbolic and neural learning. *Machine Learning*, *14*(3), 321–331.

Shavlik, J. W., & Towell, G. G. (1989). Combining explanation-based and neural learning: An algorithm and empirical results. *Connection Science*, *1*(3), 233–255.

Siegelmann, H. (1992). *Theoretical foundations of recurrent neural networks* (Ph.D. thesis), New Brunswick Rutgers, The State of New Jersey: Rutgers.

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

Silva, F. M., & Almeida, L. B. (1990). Speeding up back-propagation. In R. Eckmiller (Ed.), *Advanced neural computers* (pp. 151–158). Amsterdam: Elsevier.

Síma, J. (1994). Loading deep networks is hard. *Neural Computation*, *6*(5), 842–850.

Síma, J. (2002). Training a single sigmoidal neuron is hard. *Neural Computation*, *14*(11), 2709–2728.

Simard, P., Steinkraus, D., & Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Seventh international conference on document analysis and recognition* (pp. 958–963).

Sims, K. (1994). Evolving virtual creatures. In A. Glassner (Ed.), *ACM SIGGRAPH, Proceedings of SIGGRAPH '94, computer graphics proceedings, annual conference* (pp. 15–22). ACM Press, ISBN: 0-89791-667-0.

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

Singh, S. P. (1994). Reinforcement learning algorithms for average-payoff Markovian decision processes. In *National conference on artificial intelligence* (pp. 700–705).

Singh, S., Barto, A. G., & Chentanez, N. (2005). Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems, vol. 17 (NIPS)*. Cambridge, MA: MIT Press.

Smith, S. F. (1980). *A learning system based on genetic adaptive algorithms* (Ph.D. thesis), Univ. Pittsburgh.

Smolensky, P. (1986). Parallel distributed processing: Explorations in the microstructure of cognition. In *Information processing in dynamical systems: foundations of harmony theory, vol. 1* (pp. 194–281). Cambridge, MA, USA: MIT Press, (Chapter).

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.

Solomonoff, R. J. (1978). Complexity-based induction systems. *IEEE Transactions on Information Theory*, *IT-24*(5), 422–432.

Soloway, E. (1986). Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM*, *29*(9), 850–858.

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, *3*(9), 919–926.

Speelpenning, B. (1980). *Compiling fast partial derivatives of functions given by algorithms* (Ph.D. thesis), Urbana-Champaign: Department of Computer Science, University of Illinois.

Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., & Schmidhuber, J. (2013). Compete to compute. In *Advances in neural information processing systems (NIPS)* (pp. 2310–2318).

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011). The German traffic sign recognition benchmark: A multi-class classification competition. In *International joint conference on neural networks* (pp. 1453–1460). IEEE Press.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, *32*, 323–332.

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

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

Steijvers, M., & 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.

Steil, J. J. (2007). Online reservoir adaptation by intrinsic plasticity for backpropagation–decorrelation and echo state learning. *Neural Networks*, *20*(3), 353–364.

Stemmler, M. (1996). A single spike suffices: the simplest form of stochastic resonance in model neurons. *Network: Computation in Neural Systems*, *7*(4), 687–716.

Stoianov, I., & Zorzi, M. (2012). Emergence of a 'visual number sense' in hierarchical generative models. *Nature Neuroscience*, *15*(2), 194–196.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, *36*, 111–147.

Stoop, R., Schindler, K., & Bunimovich, L. (2000). When pyramidal neurons lock, when they respond chaotically, and when they like to synchronize. *Neuroscience Research*, *36*(1), 81–91.

Stratonovich, R. (1960). Conditional Markov processes. *Theory of Probability and Its Applications*, *5*(2), 156–178.

Sun, G., Chen, H., & Lee, Y. (1993). Time warping invariant neural networks. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems (NIPS), vol. 5* (pp. 180–187). Morgan Kaufmann.

Sun, G. Z., Giles, C. L., Chen, H. H., & Lee, Y. C. (1993). *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., & Schmidhuber, J. (2013). A linear time natural evolution strategy for non-separable functions. In *Proceedings of the genetic and evolutionary computation conference* (p. 61). Amsterdam, NL: ACM.

Sun, Y., Wierstra, D., Schaul, T., & Schmidhuber, J. (2009). Efficient natural evolution strategies. In *Proc. 11th genetic and evolutionary computation conference* (pp. 539–546).

Sutskever, I., Hinton, G. E., & Taylor, G. W. (2008). The recurrent temporal restricted Boltzmann machine. In *NIPS, vol. 21* (p. 2008).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to sequence learning with neural networks. Technical report*. arXiv:1409.3215 [cs.CL] Google. NIPS'2014.

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

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems (NIPS), vol. 12* (pp. 1057–1063).

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

Sutton, R. S., Szepesvári, C., & 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), vol. 21* (pp. 1609–1616).

Szabó, Z., Póczos, B., & Lőrincz, A. (2006). Cross-entropy optimization for independent process analysis. In *Independent component analysis and blind signal separation* (pp. 909–916). Springer.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2014). *Going deeper with convolutions. Technical report*. arXiv:1409.4842 [cs.CV], Google.

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

Taylor, G. W., Spiro, I., Bregler, C., & Fergus, R. (2011). Learning invariance through imitation. In *Conference on computer vision and pattern recognition* (pp. 2729–2736). IEEE.

Tegge, A. N., Wang, Z., Eickholt, J., & Cheng, J. (2009). NNcon: improved protein contact map prediction using 2D-recursive neural networks. *Nucleic Acids Research*, *37*(Suppl 2), W515–W518.

Teichmann, M., Wiltschut, J., & Hamker, F. (2012). Learning invariance from natural images inspired by observations in the primary visual cortex. *Neural Computation*, *24*(5), 1271–1296.

Teller, A. (1994). The evolution of mental models. In E. Kenneth, & J. Kinnear (Eds.), *Advances in genetic programming* (pp. 199–219). MIT Press.

Tenenberg, J., Karlsson, J., & Whitehead, S. (1993). Learning via task decomposition. In J. A. Meyer, H. Roitblat, & S. Wilson (Eds.), *From animals to animats 2: proceedings of the second international conference on simulation of adaptive behavior* (pp. 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., & Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*.

Tikhonov, A. N., Arsenin, V. I., & John, F. (1977). *Solutions of ill-posed problems*. Winston.

Ting, K. M., & Witten, I. H. (1997). Stacked generalization: when does it work? In *Proc. international joint conference on artificial intelligence*.

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

Tonkes, B., & 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*.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, *70*(1), 119–165.

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

Tsodyks, M., Pawelzik, K., & Markram, H. (1998). Neural networks with dynamic synapses. *Neural Computation*, *10*(4), 821–835.

Tsodyks, M. V., Skaggs, W. E., Sejnowski, T. J., & McNaughton, B. L. (1996). Population dynamics and theta rhythm phase precession of hippocampal place cell firing: a spiking neuron model. *Hippocampus*, *6*(3), 271–280.

Turaga, S. C., Murray, J. F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., et al. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, *22*(2), 511–538.

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

Turner, A. J., & Miller, J. F. (2013). Cartesian genetic programming encoded artificial neural networks: A comparison using three benchmarks. In *Proceedings of the conference on genetic and evolutionary computation, GECCO* (pp. 1005–1012).

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* (Ph.D. thesis), Universidad del Valle.

Utgoff, P. E., & Stracuzzi, D. J. (2002). Many-layered learning. *Neural Computation*, *14*(10), 2497–2529.

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

Vaillant, R., Monrocq, C., & LeCun, Y. (1994). Original approach for the localisation of objects in images. *IEE Proceedings Vision, Image, and Signal Processing*, *141*(4), 245–250.

van den Berg, T., & Whiteson, S. (2013). Critical factors in the performance of HyperNEAT. In *GECCO 2013: proceedings of the genetic and evolutionary computation conference* (pp. 759–766).

van Hasselt, H. (2012). Reinforcement learning in continuous state and action spaces. In M. Wiering, & M. van Otterlo (Eds.), *Reinforcement learning* (pp. 207–251). Springer.

Vapnik, V. (1992). Principles of risk minimization for learning theory. In D. S. Lippman, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems (NIPS), vol. 4* (pp. 831–838). Morgan Kaufmann.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer.

Versino, C., & Gambardella, L. M. (1996). Learning fine motion by using the hierarchical extended Kohonen map. In *Proc. intl. conf. on artificial neural networks* (pp. 221–226). Springer.

Veta, M., Viergever, M., Pluim, J., Stathonikos, N., & van Diest, P. J. (2013). MICCAI 2013 grand challenge on mitosis detection.

Vieira, A., & Barradas, N. (2003). A training algorithm for classification of high-dimensional data. *Neurocomputing*, *50*, 461–472.

Viglione, S. (1970). Applications of pattern recognition technology. In J. M. Mendel, & K. S. Fu (Eds.), *Adaptive, learning, and pattern recognition systems*. Academic Press.

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

Vlassis, N., Littman, M. L., & Barber, D. (2012). On the computational complexity of stochastic controller optimization in POMDPs. *ACM Transactions on Computation Theory*, *4*(4), 12.

Vogl, T., Mangis, J., Rigler, A., Zink, W., & 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.

Waldinger, R. J., & Lee, R. C. T. (1969). PROW: a step toward automatic program writing. In D. E. Walker, & L. M. Norton (Eds.), *Proceedings of the 1st international joint conference on artificial intelligence* (pp. 241–252). Morgan Kaufmann.

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

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

Wang, S., & Manning, C. (2013). Fast dropout training. In *Proceedings of the 30th international conference on machine learning* (pp. 118–126).

Wang, C., Venkatesh, S. S., & Judd, J. S. (1994). Optimal stopping and effective machine complexity in learning. In *Advances in neural information processing systems (NIPS'6)* (pp. 303–310). Morgan Kaufmann.

Watanabe, S. (1985). *Pattern recognition: human and mechanical*. New York: Wiley.

Watanabe, O. (1992). Kolmogorov complexity and computational complexity. In *EATCS monographs on theoretical computer science*. Springer.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* (Ph.D. thesis), Oxford: King's College.

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

Watrous, R. L., & Kuhn, G. M. (1992). Induction of finite-state automata using second-order recurrent networks. In J. E. Moody, S. J. Hanson, & R. P. Lippman (Eds.), *Advances in neural information processing systems, vol. 4* (pp. 309–316). Morgan Kaufmann.

Waydo, S., & Koch, C. (2008). Unsupervised learning of individuals and categories from images. *Neural Computation*, *20*(5), 1165–1178.

Weigend, A. S., & Gershenfeld, N. A. (1993). Results of the time series prediction competition at the Santa Fe Institute. In *Neural networks, 1993., IEEE international conference on* (pp. 1786–1793). IEEE.

Weigend, A. S., Rumelhart, D. E., & Huberman, B. A. (1991). Generalization by weight-elimination with application to forecasting. In R. P. Lippmann, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems (NIPS), vol. 3* (pp. 875–882). San Mateo, CA: Morgan Kaufmann.

Weiss, G. (1994). Hierarchical chunking in classifier systems. In *Proceedings of the 12th national conference on artificial intelligence, vol. 2* (pp. 1335–1340). AAAI Press/The MIT Press.

Weng, J., Ahuja, N., & Huang, T. S. (1992). Cresceptron: a self-organizing neural network which grows adaptively. In *International joint conference on neural networks, vol. 1* (pp. 576–581). IEEE.

Weng, J. J., Ahuja, N., & Huang, T. S. (1997). Learning recognition and segmentation using the cresceptron. *International Journal of Computer Vision*, *25*(2), 109–143.

Werbos, P. J. (1974). *Beyond regression: new tools for prediction and analysis in the behavioral sciences* (Ph.D. 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* (pp. 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*, vol. 1 (pp. 209–216).

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

Werbos, P. J. (1992). Neural networks, system identification, and control in the chemical industries. In D. A. White, & D. A. Sofge (Eds.), *Handbook of intelligent control: neural, fuzzy, and adaptive approaches* (pp. 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* (pp. 15–34). Springer.

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

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

Whitehead, S. (1992). *Reinforcement learning for the adaptive control of perception and action* (Ph.D. thesis), University of Rochester.

Whiteson, S. (2012). Evolutionary computation for reinforcement learning. In M. Wiering, & M. van Otterlo (Eds.), *Reinforcement learning* (pp. 325–355). Berlin, Germany: Springer.

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

Whiteson, S., & Stone, P. (2006). Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, *7*, 877–917.

Widrow, B., & 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., & Lehr, M. A. (1994). Neural networks: Applications in industry, business and science. *Communications of the ACM*, *37*(3), 93–105.

Wieland, A. P. (1991). Evolving neural network controllers for unstable systems. In *International joint conference on neural networks, vol. 2* (pp. 667–673). IEEE.

Wiering, M., & Schmidhuber, J. (1996). Solving POMDPs with Levin search and EIRA. In L. Saitta (Ed.), *Machine learning: proceedings of the thirteenth international conference* (pp. 534–542). San Francisco, CA: Morgan Kaufmann Publishers.

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

Wiering, M. A., & Schmidhuber, J. (1998b). Fast online Q($\lambda$). *Machine Learning*, *33*(1), 105–116.

Wiering, M., & van Otterlo, M. (2012). *Reinforcement learning*. Springer.

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

Wierstra, D., Schaul, T., Peters, J., & Schmidhuber, J. (2008). Natural evolution strategies. In *Congress of evolutionary computation*.

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

Wiles, J., & Elman, J. (1995). Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. In *Proceedings of the seventeenth annual conference of the cognitive science society* (pp. 482–487). MIT Press: Cambridge, MA.

Wilkinson, J. H. (Ed.) (1965). *The algebraic eigenvalue problem*. New York, NY, USA: Oxford University Press, Inc.

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

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

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

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

Williams, R. J. (1992b). Training recurrent networks using the extended Kalman filter. In *International joint conference on neural networks, vol. 4* (pp. 241–246). IEEE.

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

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

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

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

Willshaw, D. J., & von der Malsburg, C. (1976). How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society of London. Series B*, *194*, 431–445.

Windisch, D. (2005). Loading deep networks is hard: The pyramidal case. *Neural Computation*, *17*(2), 487–502.

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

Witczak, M., Korbicz, J., Mrugalski, M., & Patton, R. J. (2006). A GMDH neural network-based approach to robust fault diagnosis: Application to the DAMADICS benchmark problem. *Control Engineering Practice*, *14*(6), 671–683.

Wöllmer, M., Blaschke, C., Schindl, T., Schuller, B., Färber, B., Mayer, S., et al. (2011). On-line driver distraction detection using long short-term memory. *IEEE Transactions on Intelligent Transportation Systems (TITS)*, *12*(2), 574–582.

Wöllmer, M., Schuller, B., & Rigoll, G. (2013). Keyword spotting exploiting long short-term memory. *Speech Communication*, *55*(2), 252–265.

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 J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems (NIPS), vol. 6* (pp. 200–207). Morgan Kaufmann.

Wu, L., & Baldi, P. (2008). Learning to play go using recursive neural networks. *Neural Networks*, *21*(9), 1392–1400.

Wu, D., & Shao, L. (2014). Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In *Proc. conference on computer vision and pattern recognition*.

Wyatte, D., Curran, T., & O'Reilly, R. (2012). The limits of feedforward vision: Recurrent processing promotes robust object recognition when objects are degraded. *Journal of Cognitive Neuroscience*, *24*(11), 2248–2261.

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2010). Evolving spiking neural networks for audiovisual information processing. *Neural Networks*, *23*(7), 819–835.

Yamauchi, B. M., & 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., & DiCarlo, J. J. (2013). Hierarchical modular optimization of convolutional networks achieves representations similar to macaque IT and human ventral stream. In *Advances in neural information processing systems (NIPS)* (pp. 1–9).

Yang, M., Ji, S., Xu, W., Wang, J., Lv, F., & Yu, K., et al. (2009). Detecting human actions in surveillance videos. In *TREC video retrieval evaluation workshop*.

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

Yin, J., Meng, Y., & Jin, Y. (2012). A developmental approach to structural self-organization in reservoir computing. *IEEE Transactions on Autonomous Mental Development*, *4*(4), 273–289.

Yin, F., Wang, Q.-F., Zhang, X.-Y., & Liu, C.-L. (2013). ICDAR 2013 Chinese handwriting recognition competition. In *12th international conference on document analysis and recognition* (pp. 1464–1470).

Young, S., Davis, A., Mishtal, A., & Arel, I. (2014). Hierarchical spatiotemporal feature extraction using recurrent online clustering. *Pattern Recognition Letters*, *37*, 115–123.

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

Zamora-Martínez, F., Frinken, V., España-Boquera, S., Castro-Bleda, M., Fischer, A., & Bunke, H. (2014). Neural network language models for off-line handwriting recognition. *Pattern Recognition*, *47*(4), 1642–1652.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. CoRR, abs/1212.5701.

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

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

Zemel, R. S., & Hinton, G. E. (1994). Developing population codes by minimizing description length. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems vol. 6* (pp. 11–18). Morgan Kaufmann.

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

Zimmermann, H.-G., Tietz, C., & Grothmann, R. (2012). Forecasting with recurrent neural networks: 12 tricks. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Lecture notes in computer science*: *Vol. 7700*. *Neural networks: tricks of the trade* (2nd ed.) (pp. 687–707). Springer.

Zipser, D., Kehoe, B., Littlewort, G., & Fuster, J. (1993). A spiking network model of short-term active memory. *The Journal of Neuroscience*, *13*(8), 3406–3420.