CONTENTS

Preface xvii

Acknowledgments xix

List of Figures xxv

List of Tables xxvii

List of Contributors xxix

PART I INTRODUCTION 1

Chapter 1 Dynamical Recurrent Networks 3

John F. Kolen and Stefan C. Kroner

1.1 Introduction 3

1.2 Dynamical Recurrent Networks 4

1.3 Overview 6

1.3.1 Architectures 6

1.3.2 Capabilities 7

1.3.3 Algorithms 8

1.3.4 Limitations 9

1.3.5 Applications 10

1.4 Conclusion 11

PART II ARCHITECTURES 13

Chapter 2 Networks with Adaptive State Transitions 15

David Calvert and Stefan C. Kremer

2.1 Introduction 15

2.2 The Search for Context 15

2.2.1 Context as Input History 16

2.3 Recurrent Approaches to Context 17

2.4 Representing Context 18

2.5 Training 19

2.6 Architectures 19

2.6.1 Jordan 19

2.6.2 Elman 21
## Contents

2.6.3 Williams and Zipser 23  
2.6.4 Giles et al. 23  
2.6.5 Locally Recurrent Connections 24

### 2.7 Conclusion 25

### Chapter 3 Delay Networks: Buffers to the Rescue 27

*Tsung-Nan Lin and C. Lee Giles*

3.1 Introduction to Delay Networks 27  
3.2 Back-Propagation Through Time Learning Algorithm 28  
3.3 Delay Networks with Feedback: NARX Networks 31  
3.4 Long-Term Dependencies in NARX Networks 33  
3.4.1 Vanishing Gradients and Long-Term Dependencies 33  
3.4.2 The Effect of Delay Buffer on Vanishing Gradients 35  
3.5 Experimental Results: The Latching Problem 36  
3.6 Conclusion 38

### Chapter 4 Memory Kernels 39

*Ah Chung Tsoi, Andrew Back, José Principe, and Mike Mozer*

4.1 Introduction 39  
4.2 Different Types of Memory Kernels 40  
4.2.1 The Parameters in Each Modular Component Are Different 41  
4.2.2 Identical Parameters for Each Modular Component 43  
4.3 Generic Representation of a Memory Kernel 44  
4.3.1 State Observable Situation 45  
4.3.2 Output Observable Situation 45  
4.4 Basis Issues 45  
4.4.1 Completeness 46  
4.4.2 Gamma Filter 46  
4.4.3 Laguerre Filters 46  
4.4.4 General Conditions 46  
4.5 Universal Approximation Theorem 47  
4.6 Training Algorithms 48  
4.6.1 Training Algorithm for the Generic Predictor Weights 48  
4.6.2 State Observable Form 49  
4.6.3 Output Observable Form 50  
4.7 Illustrative Example 51  
4.8 Conclusion 54

### PART III CAPABILITIES 55

### Chapter 5 Dynamical Systems and Iterated Function Systems 57

*John F. Kolen*

5.1 Introduction 57  
5.2 Dynamical Systems 57  
5.2.1 Definition 58  
5.2.2 Trajectories, Transients, and Attractors 61
5.2.3 Attractor Dimensionality 67
5.2.4 Bifurcations and Phase Transitions 68
5.2.5 Summary 71

5.3 Iterated Function Systems 72
5.3.1 Basic Iterated Function Systems Theory 72
5.3.2 Random Iteration 75
5.3.3 Summary 76

5.4 Symbolic Dynamics 78

5.5 The DRN Connection 80
5.6 Conclusion 81

Chapter 6 Representation of Discrete States 83
C. Lee Giles and Christian Omlin
6.1 Introduction 83
6.2 Finite-State Automata 83
6.3 Neural Network Representations of DFA 85
6.3.1 Preliminaries 85
6.3.2 Feedforward Neural Networks 85
6.3.3 First- Versus Second-Order Networks 87
6.3.4 Locally Recurrent Neural Networks 90
6.3.5 Recurrent Cascade Correlation Networks 92
6.3.6 Elman Networks 94
6.3.7 NARX Recurrent Neural Networks 97
6.4 Pushdown Automata 99
6.5 Turing Machines 101
6.6 Conclusion 102

Chapter 7 Simple Stable Encodings of Finite-State Machines in Dynamic
Recurrent Networks 103
Mikel L. Forcada and Raphael C. Carrasco
7.1 Introduction 103
7.1.1 A Survey of Previous Work 103
7.1.2 Organization of the Chapter 106
7.2 Definitions 106
7.2.1 Mealy and Moore Machines and Deterministic Finite-State
Automaton 106
7.2.2 Dynamic Recurrent Networks 107
7.3 Encoding 109
7.3.1 Conditions for a DRN to Behave as a FSM 109
7.3.2 A General Encoding Scheme 111
7.4 Encoding of Mealy Machines in DRN 114
7.4.1 Mealy Machines in Second-Order DRN 114
7.4.2 Mealy Machines in First-Order DRN 120
7.5 Encoding of Moore Machines in DRN 123
7.5.1 Moore Machines in First-Order DRN 123
7.5.2 Moore Machines in Second-Order DRN 124
7.6 Encoding of Deterministic Finite-State Automata in DRN 125
  7.6.1 Encoding DFA in Elman Nets  125
  7.6.2 Encoding DFA in Second-Order DRN  125
7.7 Conclusion  126
7.8 Acknowledgments  127

Chapter 8  Representation Beyond Finite States: Alternatives to Pushdown Automata  129
Janet Wiles, Alan D. Blair, and Mikael Bodén
8.1 Introduction  129
  8.1.1 Finding Structure in Continuous-State Space  129
8.2 Hierarchies of Languages and Machines  130
  8.2.1 The Chomsky Hierarchy  130
  8.2.2 Regular and Context-Free Languages  131
  8.2.3 Context-Sensitive and Recursively Enumerable  132
  8.2.4 Learning and Induction  132
  8.2.5 Alternative Hierarchies  133
  8.2.6 Performance Limitations  133
8.3 DRNs and Nonregular Languages  134
  8.3.1 Augmenting DRNs with an External Stack  134
  8.3.2 DRNs without an External Stack  135
  8.3.3 Representing Counters in Recurrent Hidden Units  136
  8.3.4 Learning, Stability, and Generalization for  a^n b^n  138
  8.3.5 Learning the Context-Sensitive Language  a^n b^n c^n  139
8.4 Generalization and Inductive Bias  141
8.5 Conclusion  142

Chapter 9  Universal Computation and Super-Turing Capabilities  143
Hava T. Siegelmann
9.1 Introduction  143
9.2 The Model  144
9.3 Preliminary: Computational Complexity  145
9.4 Summary of Results  146
  9.4.1 Networks That Are Turing Equivalent  147
  9.4.2 Networks That Are Beyond Turing  148
9.5 Pondering Real Weights  149
9.6 Analog Computation  149
9.7 Conclusion  150
9.7 Acknowledgments  151

PART IV  ALGORITHMS  153

Chapter 10  Insertion of Prior Knowledge  155
Paolo Frasconi, C. Lee Giles, Marco Gori, and Christian Omlin
10.1 Introduction  155
10.2 Constrained Nondeterministic Insertion in First-Order Networks  156
  10.2.1 Motivations  156
Contents

10.2.2 Time-Warping Automata 156
10.2.3 Mapping Network Dynamics into Symbolic Dynamics 157
10.2.4 State Memorization 157
10.2.5 Neural Logic Operators 158
10.2.6 The Injection Algorithm 159

10.3 Second-Order Networks 160
10.3.1 DFA Encoding Algorithm 160
10.3.2 Empirical Results 162
10.3.3 Stability of the DFA Representation 164
10.3.4 Scaling Issues 166
10.3.5 DFA States with Large Indegree 166
10.3.6 Extension to Fuzzy Domains 167
10.3.7 Fuzzy Representation 170

10.4 Other Related Techniques 175
10.4.1 Recurrent Radial Basis Function Networks 175
10.4.2 Injecting “Automata Knowledge” in RBF Networks 176

10.5 Conclusion 177

Chapter 11 Gradient Calculations for Dynamic Recurrent Neural Networks 179
Barak A. Pearlmutter
11.1 Introduction 179
11.1.1 Why Recurrent Networks? 179
11.1.2 Why Hidden Units? 180
11.1.3 Continuous Versus Discrete Time 181

11.2 Learning in Networks with Fixed Points 182
11.2.1 Will a Fixed Point Exist? 182
11.2.2 Problems with Fixed Points 183
11.2.3 Recurrent Back Propagation 183
11.2.4 Deterministic Boltzmann Machines 186

11.3 Computing the Gradient Without Assuming a Fixed Point 188
11.3.1 Back Propagation Through Time 188
11.3.2 Real-Time Recurrent Learning 191
11.3.3 More Efficient Online Techniques 192
11.3.4 Time Constants 193
11.3.5 Time Delays 193
11.3.6 Extended RTRL to Time Constants and Time Delays 194
11.3.7 Long Short-Term Memory 195

11.4 Some Simulations 196
11.4.1 A Rotated Figure Eight 197
11.4.2 Computational Neuroscience: A Simulated Leech 197

11.5 Stability and Perturbation Experiments 198

11.6 Other Non-Fixed Point-Techniques 199
11.6.1 “Elman Nets” 199
11.6.2 The Moving Targets Method 199
11.6.3 Feedforward Networks with State 200
11.6.4 Teacher Forcing in Continuous Time 200
11.6.5 Jordan Nets 201
11.6.6 Teacher Forcing, RTRL, and the Kalman Filter 202

11.7 Learning with Scale Parameters 203

11.8 Conclusion 203
11.8.1 Complexity Comparison 203
11.8.2 Speeding the Optimization 204
11.8.3 Prospects and Future Work 205
11.8.4 Conclusion 205

Chapter 12 Understanding and Explaining DRN Behavior 207
Christian Omlin
12.1 Introduction 207
12.2 Performance Deterioration 208
12.3 Dynamic Space Exploration 209
  12.3.1 Extraction Algorithm 209
  12.3.2 Example 211
  12.3.3 Model Selection 213
12.4 DFA Extraction: Fool’s Gold? 215
12.5 Theoretical Foundations 216
12.6 How Can DFA Outperform Networks? 218
  12.6.1 Definitions 218
  12.6.2 Example 218
12.7 Alternative Extraction Methods 220
  12.7.1 Self-Clustering Recurrent Networks 221
  12.7.2 Explicit Cluster Modeling 221
  12.7.3 Learning Discrete Representations 222
  12.7.4 DFA Extraction as a Learning Problem 223
12.8 Extension to Fuzzy Automata 225
12.9 Application to Financial Forecasting 226
  12.9.1 A Hybrid Neural Network Architecture 226
12.10 Conclusion 227

PART V LIMITATIONS 229

Chapter 13 Evaluating Benchmark Problems by Random Guessing 231
Jurgen Schmidhuber, Sepp Hochreiter, and Yoshua Bengio
13.1 Introduction 231
13.2 Random Guessing (RG) 231
13.3 Experiments 232
  13.3.1 Latch and 2-Sequence Problems 232
  13.3.2 Parity Problem 233
  13.3.3 Tomita Grammars 234
13.4 Final Remarks 234
13.5 Conclusion 235
13.6 Acknowledgments 235

Chapter 14 Gradient Flow in Recurrent Nets: The Difficulty of Learning
Long-Term Dependencies 237
Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi,
and Jurgen Schmidhuber
14.1 Introduction 237
14.2 Exponential Error Decay 237
14.3 Dilemma: Avoiding Aradient Decay Prevents Long-Term Latching 240
14.4 Remedies 241
14.5 Conclusion 243

Chapter 15 Limiting the Computational Power of Recurrent Neural Networks: VC Dimension and Noise 245
Christopher Moore
15.1 Introduction 245
15.2 Time-Bounded Networks and VC Dimension 246
15.2.1 Real-Time Dynamical Recognizers 246
15.2.2 VC Dimension 247
15.2.3 A Memory Task That Needs a Large VC Dimension 248
15.2.4 Sinusoidal Transfer Functions and Others 249
15.3 Robustness to Noise 250
15.3.1 Robustness with Perfect Reliability 250
15.3.2 Performance as a Function of Noise 251
15.3.3 Robustness with Limited Reliability 253
15.4 Conclusion 254
15.5 Acknowledgments 254

PART VI APPLICATIONS 255

Chapter 16 Dynamical Recurrent Networks in Control 257
Danil V. Prokhorov, Gintaras V. Puskorius, and Lee A. Feldkamp
16.1 Introduction 257
16.2 Description and Execution of TLRNN 258
16.3 Elements of Training 260
16.3.1 Derivative Calculations 260
16.3.2 Weight Update Methods 262
16.4 Basic Approach to Controller Synthesis 266
16.4.1 Specifications of Controller Training 266
16.4.2 Modular Approach 268
16.4.3 Multi-Stream Training 270
16.5 Example 1 272
16.5.1 MIMO Control Problem 272
16.5.2 State Variables Accessible and Plant Equations Known 273
16.5.3 State Variables Inaccessible and Plant Equations Unknown 275
16.5.4 Further Testing, Improvements, and Summary of the Results 277
16.6 Example 2 282
16.6.1 Financial Portfolio Optimization 282
16.6.2 General Training Considerations 284
16.6.3 Empirical Simulations 284
16.6.4 Possible Extensions 288
16.7 Conclusion 288

Chapter 17 Sentence Processing and Linguistic Structure 291
Whitney Tabor
17.1 Introduction 291
17.1.1 The Role of Recurrent Connectionist Networks 292
17.1.2 The Role of Sentence Processing Studies 294
17.1.3 Overview 295

17.2 Case Studies: Dynamical Networks for Sentence Processing 295
17.2.1 Case 1: Frequency Sensitivity 297
17.2.2 Case 2: Phrase Structure 301

17.3 Conclusion 308

Chapter 18 Neural Network Architectures for the Modeling of Dynamic Systems 311

Hans-Georg Zimmermann and Ralph Neuneier

18.1 Introduction and Overview 311
18.2 Modeling Dynamic Systems by Feedforward Neural Networks 312
18.2.1 Preprocessing the Data 312
18.2.2 Suppressing Outliers by Net Internal Preprocessing 313
18.2.3 Merging MLP and RBF Networks 314
18.2.4 Modeling Dynamics by an Interaction Layer 316
18.2.5 Statistical Averaging 317
18.2.6 An Integrated Feedforward Architecture for Forecasting 318
18.3 Modeling Dynamic Systems by Recurrent Neural Networks 321
18.3.1 Representing Dynamic Systems by Recurrent Networks 322
18.3.2 Finite Unfolding in Time 323
18.3.3 Overshooting 326
18.3.4 Analysis of Partially Externally Driven Systems 327
18.3.5 Principle of Uniform Causality 328
18.3.6 Undershooting 330
18.3.7 Special Dynamics 332
18.4 Combining State-Space Reconstruction and Forecasting 334
18.4.1 Finite Unfolding in Space and Time 335
18.4.2 Smoothness 338
18.4.3 Separation of Variants and Invariants 341
18.4.4 Nonlinearity Versus Noise 342
18.4.5 Intrinsic Time 344
18.4.6 Measuring the Observed Dynamics 344
18.4.7 Extended State-Space Transformations and Periodic Orbits 346
18.4.8 Overshooting in Space and Time 347
18.4.9 Undershooting in Space and Time 348

18.5 Conclusion 350

Chapter 19 From Sequences to Data Structures: Theory and Applications 351

Paolo Frasconi, Marco Gori, Andreas Küchler, and Alessandro Sperduti

19.1 Introduction 351
19.2 Historical Remarks 352
19.3 Adaptive Processing of Structured Information 354
19.3.1 Structured Domains 354
19.3.2 The Computational Model 355
19.3.3 Graphical Transduction 357
19.3.4 On the Representational Power of Recursive Models 359
19.3.5 Learning Algorithms 359
LIST OF FIGURES

2.1 General recurrent architecture. 16
2.2 TDNN architecture. 17
2.3 Equivalent view of context units. 18
2.4 A simple recurrent network, (a), is unfolded in time, (b). 20
2.5 Jordan’s architecture. 21
2.6 Elman’s architecture. 22
2.7 Williams and Zipser’s architecture. 23
2.8 Gile’s architecture. 24
2.9 Mozer’s architecture. 24
3.1 A two-neuron recurrent network architecture. 29
3.2 The unfolded network of the two-neuron recurrent network architecture. 29
3.3 A schematic representation of back-propagation through time learning algorithm. 30
3.4 A NARX network with input memory of order 2 and output memory of order 2. 32
3.5 A variation of the NARX network. 32
3.6 A TDNN network with input memory of order 3. 33
3.7 Plots of $J(t, n)$ as a function of $n$ for different numbers of output delays. 36
3.8 Plots of the ratio $\frac{J(t, n)}{\sum_{r=1}^{n} J(t, r)}$ as a function of $n$ for different numbers of output delays. 36
3.9 The network used for the latching problem. 37
3.10 Plots of percentage of successful simulations as a function of $T$, the length of the input strings, for different number of output delays ($D = 1$, $D = 3$, and $D = 6$). 37
4.1 A diagram showing the class of recurrent neural network under study in this chapter. 40
4.2 A generic form of the short-term memory. 40
4.3 The neural network architecture considered in this chapter. 41
4.4 A tapped delay line memory model. 41
4.5 A Laguerre filter module. 42
4.6 A general modular form which encompasses the filter form as introduced in the previous section. 44
4.7 Illustration of how a Laguerre filter may be expressed in the general modular form. 44
4.8 The TINSTMA used in the worked example.
5.1 Attracting and repelling fixed points.
5.2 A demonstration comparing periodic and quasiperiodic trajectories.
5.3 Stretching and folding of state space in the logistic mapping.
5.4 From logistic map to tent map to the Bakers’ map.
5.5 Bifurcation diagrams of the logistic function and a neural network unimodal function.
5.6 A pitchfork bifurcation.
5.7 Several fractal sets generated by iterative function systems.
5.8 The difference between the limit of a single transformation (a point) and the limit of a collection of transformations (a Sierpinski triangle).
5.9 The individual transformations make three reduced copies of the attractor.
5.10 Examples of attractors generated with different transformation probabilities.
5.11 A superstable period-four orbit (CRLR) in the logistic mapping.

6.1 DFA for odd parity.
6.2 Network architectures for definite memory machines.
6.3 Example directed DeBruijn graph.
6.4 Definite memory machine.
6.5 First-order single-layer recurrent neural network.
6.6 Locally recurrent networks.
6.7 Recurrent cascade correlation networks.
6.8 Elman recurrent neural network.
6.9 NARX recurrent neural network.
6.10 Neural network pushdown automata.

7.1 The function $\eta$ (see text).
7.2 The function $\varepsilon$ (see text).

8.1 A trajectory in the state space of a first-order SRN with two recurrent hidden units, processing $a^b b^s$.
8.2 The oscillation performance for a self-recurrent unit with a bias and the instability of learning.
8.3 The trajectory in the state space of an SCN with three recurrent hidden units, processing $a^b b^s c^8$.
8.4 (a) The trajectory in the state space of an SCN with two recurrent hidden units processing $a^b b^8 c^8$ and (b) its linearizations around the fixed points.

10.1 Architecture of a first-order network.
10.2 Second-order recurrent neural network.
10.3 Performance of 10-state DFA.
10.4 Performance of 100-state DFA.
10.5 Performance of 1000-state DFA.
10.6 Fixed points of the sigmoidal discriminant function.
10.7 Existence of fixed points.
10.8 Scaling weight strength.
10.9 Transformation of a FFA into its corresponding DFA.
10.10 Recurrent network architecture for crisp representation of fuzzy finite-state automata.
<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.11</td>
<td>Network performance.</td>
</tr>
<tr>
<td>10.12</td>
<td>Fuzzy discriminant function for state representation.</td>
</tr>
<tr>
<td>10.13</td>
<td>Example of FFA transformation.</td>
</tr>
<tr>
<td>10.14</td>
<td>Stability of FFA state encoding.</td>
</tr>
<tr>
<td>10.15</td>
<td>Architecture of an $R^2BF$ network.</td>
</tr>
<tr>
<td>11.1</td>
<td>Energy landscape.</td>
</tr>
<tr>
<td>11.2</td>
<td>The architecture of a network to solve an associative version of the four-bit rotation problem.</td>
</tr>
<tr>
<td>11.3</td>
<td>A Hinton diagram of weights learned by the network of Figure 11.2.</td>
</tr>
<tr>
<td>11.4</td>
<td>Network state for all the cases in the four-bit rotation problem.</td>
</tr>
<tr>
<td>11.5</td>
<td>A recurrent network is shown on the left, and a representation of that network unfolded in time through four time steps is shown on the right.</td>
</tr>
<tr>
<td>11.6</td>
<td>The infinitesimal changes to $y$ considered in $e_i(t)$.</td>
</tr>
<tr>
<td>11.7</td>
<td>The infinitesimal changes to $y$ considered in $z_i(t)$.</td>
</tr>
<tr>
<td>11.8</td>
<td>The output of the rotated figure eight network at all the trained angles (left) and some untrained angles (right).</td>
</tr>
<tr>
<td>11.9</td>
<td>The output states $y_1$ and $y_2$ plotted against each other for a 1000 time unit run.</td>
</tr>
<tr>
<td>12.1</td>
<td>State clustering.</td>
</tr>
<tr>
<td>12.2</td>
<td>Extraction example.</td>
</tr>
<tr>
<td>12.3</td>
<td>DFA extraction algorithm.</td>
</tr>
<tr>
<td>12.4</td>
<td>Minimization of DFA.</td>
</tr>
<tr>
<td>12.5</td>
<td>Extracted DFA.</td>
</tr>
<tr>
<td>12.6</td>
<td>A simple DFA.</td>
</tr>
<tr>
<td>12.7</td>
<td>Dynamics of a recurrent network.</td>
</tr>
<tr>
<td>12.8</td>
<td>Prefix tree.</td>
</tr>
<tr>
<td>12.9</td>
<td>DFA extracted.</td>
</tr>
<tr>
<td>14.1</td>
<td>Robust latching.</td>
</tr>
<tr>
<td>15.1</td>
<td>Five ellipses can shatter the plane into $2^5$ components.</td>
</tr>
<tr>
<td>15.2</td>
<td>The family of sets $f^{-1}<em>y \left(H^*</em>{x_0}\right)$ over all finite words $v$ is independent.</td>
</tr>
<tr>
<td>15.3</td>
<td>The family of sets defined by sinusoidal inequalities has infinite VC dimension.</td>
</tr>
<tr>
<td>15.4</td>
<td>The definition of the sets $U_i$ in Lemma 15.3.1.</td>
</tr>
<tr>
<td>16.1</td>
<td>Schematic illustration of an RMLP, denoted as 1-2R-3-1R.</td>
</tr>
<tr>
<td>16.2</td>
<td>Block diagram of model reference control.</td>
</tr>
<tr>
<td>16.3</td>
<td>Distribution of the RMS error values for the feedforward controller (white) and the recurrent controller (black) during testing on 10,000 plants.</td>
</tr>
<tr>
<td>16.4</td>
<td>Performance of the recurrent controller during testing in the full state feedback case.</td>
</tr>
<tr>
<td>16.5</td>
<td>Performance of the feedforward controller during testing in the full state feedback case.</td>
</tr>
<tr>
<td>16.6</td>
<td>Test performance of the recurrent controller trained using the ID network.</td>
</tr>
<tr>
<td>16.7</td>
<td>Test performance of the recurrent controller trained with random reference signals using the ID network.</td>
</tr>
<tr>
<td>16.8</td>
<td>Test performance of the 5-20-10-2L feedforward controller trained with skylines on the nominal plant.</td>
</tr>
</tbody>
</table>
16.9 A block diagram representation of a simple, two-asset portfolio optimization system. 283
16.10 Typical results of training on the simulated price series. 286
16.11 A section of 1000 points from the simulated price series shown in Figure 16.10. 287

17.1 The bramble network (BRN). 296
17.2 Reduced-dimension plot of the manifolds associated with distinct lexical classes. 300
17.3 (a) Sierpinski triangle with stack states labeled. (b) A sample trajectory. 302
17.4 The performance of the network described above degrades in a realistic way with growth in the number of center embeddings. 304
17.5 Comparison of Gibson and Ko’s human reading-time data with the predictions of Gibson’s integration cost model. 306
17.6 Comparison of Gibson and Ko’s human reading-time data with the predictions of DA 2. 308

18.1 Net internal preprocessing to limit the influence of outliers and to eliminate unimportant inputs. 314
18.2 Net internal preprocessing cluster and the square cluster, which produces the input signals for following clusters. 315
18.3 Point predictions followed by the interaction layer. 316
18.4 Geometric interpretation of the effect of the interaction layer on the cost function. 317
18.5 Averaging of several point predictions for the same forecast horizon. 318
18.6 The integrating eleven layer architecture. 319
18.7 The identification of dynamic system using a discrete time description: input $u_t \in \mathbb{R}^k$, hidden states $s_t \in \mathbb{R}^d$, output $y_t \in \mathbb{R}^n$. 321
18.8 A time delay recurrent neural network. 323
18.9 Finite unfolding realized by shared weights $A$, $B$, $C$. 324
18.10 Optimal finite unfolding assuming error-level saturation within three time steps of Fig. 18.9 and after disconnecting the error flows from intermediate outputs $y_{t-3}^1$, $y_{t-2}^2$, $y_{t-1}^3$. 325
18.11 Overshooting is the extension of the autonomous part of the dynamics. 326
18.12 The dynamics for recurrent neural networks at the beginning of learning when weights are still small. 327
18.13 Search for a continuous embedding of a discretely measured dynamic system. 328
18.14 Refinement of the time grid by recurrent neural networks. 330
18.15 Since the intermediate targets are not available, the upper network cannot be trained. 331
18.16 Input design for undershooting networks. 332
18.17 A network architecture for volume-conserving transformations. 333
18.18 The time series of the observed state description $x_t$ may follow a very complex trajectory. 335
18.19 The function $g$ maps the observed states $x_t$ to an inner state $s_t$; $h$ is a generally nonlinear coordinate transformation back to the observation $x_t$. 336
18.20 An architecture for phase 1. 336
18.21 First approach for training phase 2. 337
18.22 Second approach for training phase 2.
18.23 Training phase 3 for fine tuning of the weights found by previous phases.
18.24 Triangle inequality smoothness (compare Eq. 18.58).
18.25 The dynamics of a pendulum can be separated in only one variant, the angle \( \varphi \), and in infinite number of invariants.
18.26 Variant-invariant separation of a dynamics.
18.27 Variant-invariant separation by neural networks.
18.28 An inappropriate description of the state of a dynamic system can be misinterpreted as noise.
18.29 The unfolding of singularities.
18.30 The intrinsic time concept tries to construct a dynamics with a constant velocity in a phase space.
18.31 Following the state description \( x_{t+1} \) of Eq. 18.62, the \( 2m \) forecasts are combined to \( m \) predictions, which are subsequently averaged to suppress noise.
18.32 By this architecture, we are able to describe a partially autonomous system which is also driven by external inputs.
18.33 The one-dimensional state space of an oscillator is a folded slope that can be unfolded as a circle in two dimensions.
18.34 Comparison of the unfolding in space for two different auto associators used as subnetworks in the recurrent network of Fig. 18.31.
18.35 Combining overshooting and unfolding in space and time.
18.36 Integration of undershooting into the unfolding in space and time concept.

19.1 A sequence seen as a data structure built on 1-SDOAGs.
19.2 A drawing represented by a tree with local features forming node labels.
19.3 Graphical model of the computation carried out by a dynamic neural network on sequences.
19.4 A two-layered MLP implementing the state transition function of a recursive neural network for \( k \)-DOAG data structures.
19.5 Graphical model of the computation carried out by a recursive neural network on an example of 2-DOAG.
19.6 Illustration of the “never-ending movie” input skeleton.
19.7 Example of how a logo can be represented as a structure.
19.8 The tree representation is particularly convenient when considering global transformations of the logo, like scaling and rotations.
19.9 A finite initial segment of a search-tree generated by an automated deduction system for one specific proof problem.
# LIST OF TABLES

5.1 A taxonomy of dynamical systems. .......................... 59
5.2 Attractor symbol sequences for the logistic map. ......... 79
6.1 Network architectures and computational models. .......... 102
7.1 Minimum values of weights and limiting values for encoding a Mealy machine. 117
7.2 Values of weights and limiting values for encoding a DFA on a second-order DRN without biases. 120
7.3 A summary of the encodings proposed in the chapter. .... 126
8.1 Chomsky hierarchy. ......................................... 130
9.1 The computational power of recurrent neural networks. ... 147
10.1 Comparison of different DFA encoding methods. ........ 177
11.1 A summary of the complexity of some learning procedures for recurrent networks. 204
12.1 Network generalization performance. ...................... 208
12.2 DFA generalization performance. ........................ 215
12.3 Extracted DFA for sparse training sets. .................. 225
16.1 Nominal values of parameters $\alpha$ for the MIMO plant of Narendra and Mukhopadhyay (1994). 273
17.1 Grammar 2. (implemented in dynamical automaton 1). .... 303
17.2 Dynamical automaton 1. .................................. 303
17.3 Dynamical automaton 2: transitions. ..................... 307
17.4 Dynamical automaton 2: compartment definitions. ....... 307