Analysis, Design, and Optimization of Cellular Neural Networks

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
Doctor of Technical Sciences

presented by
MARTIN HÄNGGI
Dipl. El.-Ing. ETH
born May 11, 1969
citizen of Nunningen/SO and Niederglatt/ZH

accepted on the recommendation of
Prof. Dr. George S. Moschytz, examiner
Prof. Dr. Leon O. Chua, co-examiner

1999
Für meine Eltern
Acknowledgments

I wish to express my deep gratitude to my thesis advisor Prof. George S. Moschytz for providing an excellent research environment at the Signal and Information Processing Laboratory (ISI) and for the precious freedom he gave me to try out my own ideas. His engaged interest and hours of proofreading substantially improved my publications and this thesis. Also, I am greatly indebted to him for the many opportunities I had in the course of my thesis to present my work at various occasions.

I would like to thank Prof. Leon O. Chua for his stimulating lectures at ISI in the summers 1996 and 1998, for his immense enthusiasm for the cellular neural network, and for serving as co-examiner of this thesis. I also wish to thank Prof. Hari Reddy for introducing me to delta operators and for our fruitful discussions.

I was very fortunate to share the office with Bahram Mirzai when I started to work on my project; I profited from many enriching discussions on CNNs and other important subjects. I am also indebted to Drahošlav Lim for providing the necessary insight into the physical limitations of CNN hardware.

Dr. Armin Friedli and Michael Vollmer from the Swiss Center for Scientific Computing showed great interest in my simulation problems. I gratefully acknowledge their assistance with the Paragon Supercomputer.

I enjoyed and learned a lot from the collaboration with Pio Balmelli and Ruben Notari during the supervision of their diploma project on stochastic optimization of CNNs. In an outstanding semester project, Simon Moser and Eric Pfaffhauser developed a Java CNN simulator that is now being used by numerous CNN researchers and lecturers throughout the world.

Finally, I would like to extend my thanks to all colleagues at ISI for providing a very pleasant environment and a stimulating atmosphere.
Abstract

Cellular neural networks (CNNs) constitute a class of recurrent and locally coupled arrays of identical dynamical systems (cells). The underlying equation governing the dynamics of each cell is nonlinear and the cells are assumed to operate in parallel. The connectivity among the cells is determined by a set of parameters denoted as a template set. A specific task is implemented by determining the appropriate template set.

Signal processing via CNNs only becomes efficient if the network is implemented in analog hardware. In view of the physical limitations that analog implementations entail, robust operation of a CNN chip with respect to parameter variations has to be insured. By far not all mathematically possible CNN tasks can be carried out reliably on an analog chip, some of them are inherently too sensitive.

We define a robustness measure to quantify the degree of robustness and propose an exact and direct analytical design method for the synthesis of optimally robust network parameters. This method is restricted to the class of so-called locally regular templates, which is rigorously defined. It turns out that the complementary class, the locally irregular templates, constitute precisely the class of inherently sensitive templates, which makes the synthesis method generally applicable to all tasks that allow robust operation.

Processing speed is always crucial when discussing signal processing devices. In the case of the CNN, it is shown that the settling time of locally regular templates can be specified in closed analytical expressions, which permits, on the one hand, template optimization with respect to speed and, on the other hand, efficient numerical integration of CNNs. Interdependence between robustness and speed issues are also addressed.
Seite Leer / Blank leaf
Another goal pursued is the unification of the theory of continuous-time and discrete-time CNNs. By means of a delta-operator approach, it is proven that basically the same templates can be used for both of these classes, even if their nonlinear output functions differ. However, not all tasks are feasible on all types of implementation. The common subset of templates that run on virtually any CNN chip built so far is, once again, the set of locally regular templates — we conclude that local regularity in fact is a key concept in chip-oriented CNN theory.

More complex CNN optimization problems that cannot be solved analytically necessitate resorting to numerical methods. Among these, stochastic optimization techniques such as genetic algorithms prove their usefulness, for example in image classification problems.
Kurzfassung


In der Signalverarbeitung kommen die Vorzüge der CNNs bezüglich Geschwindigkeit und Energieverbrauch erst zur Geltung, wenn das Netzwerk als analoger Prozessor in Hardware integriert wird. Aufgrund der physikalisch begrenzten Genauigkeit analoger Schaltungen muss sichergestellt werden, dass das CNN robust ist gegen Abweichungen der Parameter vom Nominalwert. Bei weitem nicht alle mathematisch möglichen CNN Operationen können mit einem analogen Prozessor durchgeführt werden, bei vielen ist die Sensitivität inhärent zu hoch.

Basierend auf einem Robustheitsmass wird eine exakte und direkte analytische Methode vorgeschlagen, mit der optimal robuste Netzwerkparameter bestimmt werden können. Die Methode ist beschränkt auf die Klasse der lokal regulären Templates, die präzise definiert wird. Es stellt sich heraus, dass die lokal irregulären Templates genau diejenigen sind, welche zu sensitiv sind für ein analogen Netzwerk. Umgekehrt formuliert deckt die Synthesemethode alle Operationen ab, die mit genügender Robustheit realisiert werden können.

Beim Vergleich von Signalprozessoren ist die Verarbeitungsgeschwindigkeit von zentraler Bedeutung. Beim CNN wird sie durch die Einschwingzeit bestimmt, die für lokal reguläre Templates in analytisch geschlossener Form angegeben werden kann. Dies erlaubt einerseits, Templates in bezug auf Geschwindigkeit zu optimieren, andererseits, CNNs numerisch effizient zu simu-
Kurzfassung


Für kompliziertere Optimierungsprobleme, die nicht analytisch gelöst werden können, muss auf numerische Verfahren ausgewichen werden. Stochastische Optimierungsmethoden wie genetische Algorithmen sind universell einsetzbar, was am Beispiel eines Klassifizierungsproblems gezeigt wird.
Contents

1 Introduction ........................................... 1
  1.1 Motivation .......................................... 1
  1.2 CNN Basics ........................................ 2
  1.3 Planar and Bipolar CNNs ......................... 5
  1.4 Circuit Implementation of CNNs .................. 7
  1.5 Outline of the Thesis ............................ 9

2 Robust Template Design .............................. 11
  2.1 Introduction ...................................... 11
  2.2 The Robustness of a CNN Template Set ........ 13
    2.2.1 Absolute and Relative Robustness .......... 13
    2.2.2 Template Scaling ............................ 15
  2.3 Direct Design of Optimally Robust Templates ... 16
    2.3.1 Problem Statement .......................... 16
    2.3.2 Elimination of Redundant Inequalities ...... 17
    2.3.3 The Exact and Direct Analytical Solution .. 17
  2.4 Examples ........................................ 22
  2.5 Making Templates Optimally Robust ............. 25
  2.6 Optimum Templates for a Specific CNN Implementation ... 27
    2.6.1 A Modular $g_m$-C Programmable CNN Implementation 27
    2.6.2 Deriving Optimum Templates for a Specific Chip .... 29
  2.7 Conclusions ..................................... 31

3 CNN Settling Time .................................... 35
  3.1 Introduction ..................................... 35
    3.1.1 The Settling Time of a Bipolar CNN ........ 35
    3.1.2 Significance of the Settling Time .......... 36
  3.2 The Exact Approach ................................ 37
    3.2.1 Uncoupled Templates ......................... 37
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2</td>
<td>Coupled Templates without Propagation</td>
<td>42</td>
</tr>
<tr>
<td>3.3</td>
<td>Analysis of Propagation-type Templates</td>
<td>43</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Introduction</td>
<td>43</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Propagation Strings</td>
<td>43</td>
</tr>
<tr>
<td>3.3.3</td>
<td>The One-cell Propagation Time</td>
<td>47</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Estimation of the Settling Time</td>
<td>47</td>
</tr>
<tr>
<td>3.4</td>
<td>Examples of Propagation-type Templates</td>
<td>47</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Shadowing</td>
<td>47</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Global Connectivity Detection</td>
<td>49</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Connected Component Detection</td>
<td>55</td>
</tr>
<tr>
<td>3.5</td>
<td>Sensitivity to Parameter Deviations</td>
<td>58</td>
</tr>
<tr>
<td>3.5.1</td>
<td>The Perturbed Settling Time</td>
<td>58</td>
</tr>
<tr>
<td>3.6</td>
<td>Robustness vs. Processing Speed</td>
<td>63</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Uncoupled Templates</td>
<td>63</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Propagation-type Templates</td>
<td>65</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Pseudo-gradient Optimization for Speed</td>
<td>68</td>
</tr>
<tr>
<td>3.7</td>
<td>Conclusions</td>
<td>69</td>
</tr>
</tbody>
</table>

4 Unification of Continuous- and Discrete-time CNNs  71

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Discrete-time CNNs</td>
<td>71</td>
</tr>
<tr>
<td>4.2</td>
<td>Delta Operator Based CNNs</td>
<td>72</td>
</tr>
<tr>
<td>4.2.1</td>
<td>The Delta Operator</td>
<td>72</td>
</tr>
<tr>
<td>4.2.2</td>
<td>The ((\delta, c))-CNN</td>
<td>73</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Robust Template Design for ((\delta, c))-CNNs</td>
<td>75</td>
</tr>
<tr>
<td>4.3</td>
<td>Efficient Numerical Integration of CNNs</td>
<td>78</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Numerical Integration and Sampling</td>
<td>78</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Forward Euler Integration and the Delta Operator</td>
<td>80</td>
</tr>
<tr>
<td>4.3.3</td>
<td>The CNN Sampling Theorem</td>
<td>81</td>
</tr>
<tr>
<td>4.4</td>
<td>Summary</td>
<td>91</td>
</tr>
</tbody>
</table>

5 Stochastic Optimization 93

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>93</td>
</tr>
<tr>
<td>5.2</td>
<td>Genetic Optimization of CNNs</td>
<td>95</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Genetic Algorithms</td>
<td>95</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Problem Statement</td>
<td>99</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Coding</td>
<td>100</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Search for Solutions ((GA_1))</td>
<td>101</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Robustness Optimization ((GA_2))</td>
<td>101</td>
</tr>
<tr>
<td>5.2.6</td>
<td>Parallel Implementation on a Supercomputer</td>
<td>102</td>
</tr>
<tr>
<td>5.2.7</td>
<td>Results</td>
<td>103</td>
</tr>
<tr>
<td>5.3</td>
<td>Other and Combined Approaches</td>
<td>106</td>
</tr>
<tr>
<td>5.3.1</td>
<td>The Hill Climbing Approach</td>
<td>106</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.3.2 The Averaging Approach</td>
<td>107</td>
<td></td>
</tr>
<tr>
<td>5.3.3 The Hybrid Approach</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>5.4 Attacking the General Classification Problem</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>5.4.1 The General Classification Problem</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>5.4.2 Problem Statement</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>5.4.3 The Genetic Approach</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>5.4.4 Results</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>5.5 Summary</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>6 Conclusions</td>
<td>119</td>
<td></td>
</tr>
<tr>
<td>A CNN Task Library</td>
<td>123</td>
<td></td>
</tr>
<tr>
<td>B Simulation and Visualization of CNN Dynamics</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>B.1 Introduction</td>
<td>127</td>
<td></td>
</tr>
<tr>
<td>B.2 Brief Characterization</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>B.3 Technical Information</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>List of Symbols</td>
<td>131</td>
<td></td>
</tr>
<tr>
<td>Bibliography</td>
<td>135</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

While being easily solved by human beings and animals, practical engineering problems such as image and speech recognition pose formidable difficulties to today's conventional digital computing technology. In consequence, new methods are being developed by investigating the mechanisms of the human brain, which are particularly promising in the field of image and speech processing.

The human brain is an extremely complex nonlinear system, consisting of billions of simple processing elements, the so-called neurons. Inspired by this biological network of neurons and deeply impressed by its signal processing capabilities, scientists and engineers design simplified artificial models with the far aim of achieving a performance comparable to the biological ideal.

These artificial neural networks (ANNs) [1] belong, technically speaking, to the category of distributed systems, consisting of many processing elements, the neurons, that are interconnected according to some structure. A characteristic of ANNs is that the neurons are identical and perform a relatively simple but nonlinear function. Only the network as the ensemble of all neurons features an interesting behavior.

Various classes of ANNs have been proposed and investigated in the last two decades, including feedforward and recurrent, time-continuous and discrete-time, and synchronous and asynchronous systems of different topologies and with different types of neurons. The Hopfield network [2] represents an im-
portant and well-studied class of ANNs. This network is fully connected and recurrent in the sense that the output values of all neurons are fed back to the inputs of all neurons. The neuron computes a nonlinear, monotonic, bounded, and differentiable function (a sigmoid function) of the sum of its weighted inputs. Although attempts have been made towards implementing these networks, their high degree of connectivity impedes the integration in VLSI technology. Instead, they have to be simulated on digital computers, which requires high computational power and usually results in bulky systems with high power consumption.

In 1988, Chua and Yang proposed the cellular neural network (CNN) [3,4] which can be viewed as a special case of a continuous-time Hopfield network. It differs from the analog Hopfield network in its local connectivity property, its space-invariant weight patterns, and the piece-wise linear output function of its neurons or cells which are arranged in a regular grid of dimension one or two. These properties allow its realization in VLSI technology, resulting in CNN chips that are tailor-made for real time signal processing. In fact, many complex scientific problems can be formulated with regular grids, where direct interaction between the signals on various grid points is limited within a finite local neighborhood, which is also called the sphere of influence. Hence, the most fundamental ingredients of the CNN paradigm are: the use of analog processing cells with continuous signal values, and local interaction within a finite radius.

Applications of the CNN include image and video signal processing, nonlinear signal processing in general, modeling of biological systems and higher brain functions, pattern recognition and generation, and the solving of partial differential equations. The CNN has turned out to be a very general paradigm for complexity, based on nonlinear dynamics.

In the following sections we provide some basic definitions and theorems on the characteristics and properties of CNNs.

1.2 CNN Basics

A cellular neural network or, more generally, a cellular nonlinear network is an ensemble of spatially arranged cells, where each cell is itself a dynamical system that is locally coupled to its neighboring cells within some prescribed sphere of influence [3,5–7].

A CNN is characterized by its topology and its dynamics. The topology
1.2. CNN Basics

determines the arrangement of cells (e.g., a square or a hexagonal grid) and the dimensionality. The dynamics describes the temporal evolution of each cell which is assumed to be governed by the differential equation

$$\frac{dx_\mu(t)}{dt} = -x_\mu(t) + \sum_{v \in \mathcal{N}_\mu} a_{\mu,v} f(x_v(t)) + \sum_{v \in \mathcal{N}_\mu} b_{\mu,v} u_v + I_\mu, \quad (1.1)$$

where $\mu = (\mu_1, \mu_2, \ldots, \mu_q)$ and $v = (v_1, v_2, \ldots, v_q)$ are multi-indices. The use of multi-indices allows us to include the notion of multi-layer CNN introduced in [3] as a subcase of (1.1). We define $\text{dim}(\mu)$ to be the order of a CNN. We speak of a one-dimensional CNN and a planar CNN if $\text{dim}(\mu) = 1$ and $\text{dim}(\mu) = 2$, respectively. Fig. 1.1 depicts an example of the topology of a CNN of order two.

The summations in (1.1) are performed over all cells which are connected with the cell at position $\mu$, denoted by $C_\mu$. We denote the set of these cells by $\mathcal{N}_\mu$, the neighborhood of $C_\mu$. The cardinality of $\mathcal{N}_\mu$ is the size of the neighborhood and is denoted by $|\mathcal{N}_\mu|$. The state and input of a cell $C_\mu$ are defined to be $x_\mu(t)$ and $u_\mu$, respectively. The input is assumed to be time-independent. Furthermore, by definition, $y_\mu(t) = f(x_\mu(t))$ is the output of the cell $C_\mu$ at time $t$. In particular, we denote by $x^*_\mu$ and $y^*_\mu$ the state and the output of $C_\mu$ at equilibrium, respectively, i.e.,

$$x^*_\mu = \sum_{v \in \mathcal{N}_\mu} a_{\mu,v} y^*_v + \sum_{v \in \mathcal{N}_\mu} b_{\mu,v} u_v + I_\mu, \quad (1.2)$$

$$y^*_\mu = f(x^*_\mu).$$

Figure 1.1: Topology of a planar CNN
Chapter 1. Introduction

The parameters of the CNN at cell $C_\mu$ are assumed to be time-independent and are collected in a set denoted by $T_\mu$, the template set\footnote{The term of a “template set” or simply “template” originates in image processing, where the first applications of CNNs were considered [4].} at $C_\mu$, consisting of

$$T_\mu = \{a_{\mu v}, b_{\mu v}, I_\mu \mid v \in N_\mu \}.$$ More specifically, $a_{\mu v}$ are called the feedback parameters, $b_{\mu v}$ the control parameters and $I_\mu$ the bias term. By these definitions, we assume that the coupling between the cells is linear.

The nonlinearity $f(\cdot)$ is the following piece-wise linear function (Fig. 1.2):

$$f(x) = \frac{1}{2}(|x + 1| - |x - 1|).$$ (1.3)

We will also denote it by sat(\cdot), the saturation function. Other non-linearities, such as the hard-limiting (sign) function

$$\text{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases},$$ (1.4)

may also be considered.

In the case of the saturation non-linearity, we speak of a linear cell if the state of the cell is in the linear region of sat(\cdot), i.e., $|y(t)| = |x(t)| < 1$, and

Figure 1.2: The saturation output function.
otherwise of a saturated cell. A saturated cell can take on only the values \( \pm 1 \), i.e., it is bipolar valued.

So-called spatially invariant CNNs constitute an important class of CNNs. Their significance lies in the ease of their implementation and in their analytic tractability, as well as in a wide range of applications. Spatially invariant CNNs are defined by the requirement that \( T_{\mu} \) is independent of \( \mu \), i.e., \( a_{\mu \nu} = a_{\nu} \), etc. In other words, for each cell, the connectivity with neighboring cells and the template values are independent of the cell’s position on the grid. In the case of spatially invariant CNNs, we use the notions of \( A \)-template and \( B \)-template for the sets \( A = \{ a_{\mu} \} \) and \( B = \{ b_{\mu} \} \), respectively.

### 1.3 Planar and Bipolar CNNs

Spatially invariant CNNs of order one and two are the main subject of this thesis. In the case of planar CNNs, we assume the cells to be arranged as vertices of a rectangular grid of size \( M \times N \), with \( M \) rows and \( N \) columns. Furthermore, we use the indices \( (i,j) \) instead of \( (\mu_1, \mu_2) \), etc. Eq. (1.1) can then be written as

\[
\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{mn \in \mathcal{N}_{ij}^r} a_{ij,mn} \text{sat}(x_{mn}(i)) + \sum_{mn \in \mathcal{N}_{ij}^r} b_{ij,mn}u_{mn} + I, \tag{1.5}
\]

where \( \mathcal{N}_{ij}^r \) denotes the \( r \)-neighborhood of the cell \( C_{ij} \) defined as

\[
\mathcal{N}_{ij}^r = \{ (mn) : |m-i| \leq r, |n-j| \leq r \}. \tag{1.6}
\]

The number of connections in the \( r \)-neighborhood is given by

\[
|\mathcal{N}_{ij}^r| = (2r + 1)^2.
\]

For later convenience, we adopt the notation \( \mathcal{N}_{ij} \) for \( \mathcal{N}_{ij}^1 \).

In the case of nearest neighborhood CNNs, i.e., \( r = 1 \), the templates \( A \) and \( B \) are \( 3 \times 3 \) matrices and we employ the following notation to denote their entries:

\[
A = \begin{bmatrix}
a_1 & a_2 & a_3 \\
a_4 & a_5 & a_6 \\
a_7 & a_8 & a_9
\end{bmatrix} \quad B = \begin{bmatrix}
b_1 & b_2 & b_3 \\
b_4 & b_5 & b_6 \\
b_7 & b_8 & b_9
\end{bmatrix}.
\]
The center entry of the $A$-template, $a_5$, corresponds to the self-feedback of a cell and is often denoted by $a_c$. Similarly, we denote $b_5$ also by $b_c$. A template, say $A$, is defined to be symmetric if

$$a_1 = a_3, \quad a_2 = a_8, \quad a_4 = a_6, \quad a_7 = a_9,$$

and anti-symmetric if

$$a_1 = -a_3, \quad a_2 = -a_8, \quad a_4 = -a_6, \quad a_7 = -a_9.$$

If all elements in the $A$-template are zero except possibly the center element, $a_c$, the CNN is said to be uncoupled since there is no feedback from the outputs of the neighbors. If at least one of the off-center entries in $A$ is non-zero, the CNN is coupled. For notational convenience, the 19 parameters $(A, B, I)$ are often rearranged into a single one-dimensional vector $\mathbf{T} \in \mathbb{R}^{19}$, henceforth called a template vector or, biologically inspired, a CNN gene:

$$\mathbf{T} = [a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, I]^t.$$ 

So far we have considered the system of differential equations given in (1.5) as an initial value problem. In most applications the system (1.5) is also subjected to some boundary constraints. Two kinds of boundary conditions are generally considered: constant and periodic boundary values. In the case of constant boundary values, the weight of a connection leading to a boundary cell is multiplied with the constant value of the boundary. In the latter case, the corresponding weight is multiplied with the state of the cell on the opposite edge of the grid, i.e., the edges of the grid “wrap around”. In the presence of non-zero boundary conditions (1.5) has to be modified to include the contribution of the boundary, e.g., for constant boundary values (1.5) becomes

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{mn \in \mathcal{N}_{ij}^-} a_{mn} \text{sat}(x_{mn}(t)) + \sum_{mn \in \mathcal{N}_{ij}^+} b_{mn} u_{mn} + I + \partial_{ij}, \quad (1.7)$$

where $\partial_{ij}$ is the contribution of the boundary to the state of the cell $C_{ij}$.

For computational convenience, we rewrite (1.7) in vector notation. Let $x$ be the state vector comprising the states of all cells. The components of $x$ correspond to the state of cells in the order of upper left to lower right. Eq. (1.7) can then be rewritten as

$$\dot{x}(t) = -x(t) + A \text{sat}(x(t)) + Bu + I + \partial,$$ 

(1.8)
1.4. Circuit Implementation of CNNs

where the saturation function is understood component-wise. The entries of the system matrices $A$ and $B$ can be obtained from (1.7) and are denoted by $A_{ij}$ and $B_{ij}$, respectively. In most cases, these matrices are very sparse. In particular, for symmetric or anti-symmetric templates the system matrices $A, B$, and $A - a_c 1, B - b_c 1$ are symmetric or anti-symmetric in the usual sense, respectively, where $1$ denotes the identity matrix.\(^2\)

In many applications we are interested in bipolar ($\pm 1$) outputs $y_{ij}^*$. If the input is also bipolar, we speak of a bipolar CNN.

**Definition 1.1 (Bipolar CNN.)**
A CNN is bipolar, if $u, y^* \in \mathbb{B}^{MN}$ and $x(0) \in \mathbb{B}_0^{MN}$ for $\mathbb{B} = \{-1, 1\}$ and $\mathbb{B}_0 = \{-1, 0, 1\}$.

We conclude this section by stating a theorem which provides a sufficient condition for bipolar outputs at equilibrium.

**Theorem 1.1 (Condition for bipolar output.)**
Assuming that the system (1.7) admits an equilibrium and that the self-coupling term $a_c$ satisfies $a_c > 1$, then the outputs of the system (1.7) at equilibrium are bipolar.

For a proof we refer to [8]. The basic idea is to show that for $a_c > 1$ any equilibrium in the linear region becomes unstable, i.e., there is at least one eigenvalue $\lambda$ with $\text{Re}(\lambda) > 0$.

1.4 Circuit Implementation of CNNs

Since the numerical integration of CNNs is computationally expensive, applications of CNNs in image and signal processing only become efficient if the network equation can be implemented in analog hardware. A variety of approaches have been proposed and successfully implemented [9–14]. Consider the circuit depicted in Fig. 1.3. The differential equation describing this circuit

\(^2\)Note that the definition of an anti-symmetric template does not necessarily require a zero center entry.
corresponds to (1.7) and can be written as

\[ C \frac{d\tilde{x}_{ij}(\tau)}{d\tau} = -\frac{1}{R} \tilde{x}_{ij}(\tau) + \sum_{mn \in \mathcal{N}_{ij}'} \tilde{a}_{mn} f(\tilde{x}_{mn}(\tau)) + \sum_{mn \in \mathcal{N}_{ij}'} \tilde{b}_{mn} \tilde{u}_{mn} + \tilde{I}, \quad (1.9) \]

where the tilde signifies non-normalized parameters. The state of a cell is the voltage across the capacitor. The weights are realized as voltage controlled current sources.

In this thesis, we will be mainly concerned with the normalized equation (1.7), where all quantities are dimensionless. Its relation to (1.9) is established by the following transformations:

\[
\begin{align*}
    t & \rightarrow \frac{\tau}{RC} \\
    x_{mn} & \rightarrow \tilde{x}_{mn} \\
    u_{mn} & \rightarrow \tilde{u}_{mn} \\
    a_{mn} & \rightarrow R\tilde{a}_{mn} \\
    b_{mn} & \rightarrow R\tilde{b}_{mn} \\
    I & \rightarrow R\tilde{I}
\end{align*}
\]

Analog implementations imply a number of limitations that need to be taken into account in the theory in order to assure a correct operation of the system. The main limitations concern parameter accuracy which can only be accommodated with a precision of a few percent of the nominal values [9, 14–16], and the characteristic of the saturation function; in particular saturation at exactly ±1 cannot be guaranteed.
1.5 Outline of the Thesis

The nonlinearities inherent in neural networks are difficult to handle mathematically. In the case of the CNN, its property of being time-continuous and having a recurrent structure poses additional obstacles. Nonetheless, the behavior of some important classes of CNNs can be analyzed analytically, and exact design methods can be developed.

The operation of a CNN is determined by an appropriate choice of template sets, inputs, and the initial values of the states. Assuming that the parameters are chosen in a way that we end up with a stable equilibrium of the system (1.1), the operation of a CNN can be considered as a mapping of the initial configurations onto the corresponding desired outputs. The design of a template set consists of determining those values that perform a specific mapping. Robust design then takes into account physical limitations of analog implementations, such as input or initial data noise, or parameter inaccuracies. In Chapter 2, an exact and direct analytical method for the design of optimally robust templates is proposed. The template synthesis procedure either starts without any initial template value given an input-output mapping, or can be used to make existing (but sensitive) templates optimally robust.

For both simulations and applications of the CNN, the settling time is an important issue, which is discussed in Chapter 3. The settling time, or, in terms of a signal processing device, the processing speed, is shown to depend on the template parameters in a highly nonlinear manner. Furthermore, the interdependence between processing speed and robustness is addressed.

Chapter 4 deals with continuous-time and discrete-time CNNs and their unification by means of the delta-operator approach. It is proven that the same template parameters can be used for both classes of CNNs, and that discrete-time CNNs correspond to numerically integrated continuous-time CNNs when a Forward Euler algorithm is applied. This result permits a drastic acceleration of the simulation of a broad class of CNNs.

Finally, Chapter 5 introduces stochastic optimization techniques that are very generally applicable if analytical methods fail. Genetic algorithms and simulated annealing are powerful tools for multi-dimensional, multimodal, and highly nonlinear optimization tasks. In the context of CNNs, they are useful for complex and multi-objective template learning problems.

In the Appendix, CNN tasks that are used as examples throughout the thesis are described, and a graphical CNN simulator designed for teaching and research purposes is presented.
Chapter 2

Robust Template Design

2.1 Introduction

The problem of template design or template learning is a key topic in CNN research; the methods which have been investigated since the inception of the CNN may be classified as analytical methods [17–19], local learning algorithms [20–22], and global learning algorithms [23, 24]. The analytical approaches are based upon a set of local rules characterizing the dynamics of a cell, depending on its neighboring cells.

Definition 2.1 (Local rules.)
A local rule prescribes whether the state’s derivative of a cell $C_{ij}$, $\dot{x}_{ij}(t)$, is to be negative or positive for a particular bipolar configuration of the input and output values of the neighboring cells. If this configuration appears in a CNN run, $y_{ij}$ will toggle from $-1$ to $1$ or vice versa, according to $\text{sgn}(\dot{x}_{ij}(t))$. While $x_{ij}(t)$ travels through the linear region, the neighbors have constant output.
A set of local rules defines $\text{sgn}(\dot{x}_{ij}(t))$ for all possible configurations.

These rules are transformed into an affine set of inequalities that has to be solved to get correctly operating templates. Local learning algorithms are derived from training methods developed for other neural networks like multilayer perceptrons, and their global counterparts mostly use stochastic opti-
mization techniques\textsuperscript{1} like genetic algorithms \cite{23} or simulated annealing \cite{24}.

Analog VLSI implementations of the network equation (1.5) have a number of limitations that need to be taken into account in the theory of CNNs in order to guarantee correct and efficient operation of analog VLSI hardware. Template parameters can only be realized with a precision of typically 5\% to 10\% of the nominal values, and usually only a discrete set of possible values is available \cite{25,26}. Further sources of error are

- perturbations of the input and the initial state.
- the output nonlinearity. Both the slope in the linear region and the clipping level may deviate from (1.3).
- the mismatch of the cell's time constant (product of the resistor's and the capacitor's values in each cell). Different time constants of interacting cells may result in unexpected network behavior.
- the limited state swing. The cell's states saturate at some level $x_{\text{max}}$, determined by the bias voltage of transistors or operational amplifiers, leading to non-exponential transients. (The solutions of the ideal network equation (1.5) are piecewise exponential functions.)

The requirement that a template set fulfills a given task reliably under these circumstances poses additional obstacles to template design. A CNN operation that is carried out reliably despite all these imperfections, is a so-called \textit{robust} operation. For our analysis, we consider only perturbations of the template parameters, assuming that inaccuracies in input, initial state, and clipping level can be accounted for by large enough errors for templates. Accordingly, the \textit{robustness} issue reduces to the problem of finding template parameters that can tolerate deviations from their nominal values while still executing the correct operation.

Most methodologies for the design of robust parameters deal with the problem of solving a system of affine inequalities for the "best" or, at least, sufficiently robust template \cite{19,27-30}. The proposed algorithms include linear programming or relaxation methods which entail considerable effort, often without yielding guaranteed optimal solutions. For the class of CNN operations that can be characterized by a set of local rules\textsuperscript{2}, we show that after a

\textsuperscript{1}See Chapter 5.
\textsuperscript{2}In Sec. 4.3.3, it is shown that the class of templates satisfying a set of local rules is a subclass of the so-called \textit{locally regular} class of templates.
small translation of the template space, the set of inequalities becomes homogeneous, and we propose to apply matrix-vector notation to solve the CNN design problem very generally using simple matrix algebra. Furthermore, our approach provides insight into the interaction of the parameters.

Stability issues of CNNs [3, 31–35] are beyond the scope of this chapter. If the system of inequalities is consistent, does not contain any loops\(^3\), and encompasses all cell configurations at the desired equilibria, then its solution will be stable, since (at least) one the configurations is necessarily at the end of the chain of the configurations the CNN travels through.

2.2 The Robustness of a CNN Template Set

2.2.1 Absolute and Relative Robustness

The robustness of a CNN template set is a measure which quantifies the degree by which a template set can be altered while still producing the desired output. In programs for CNN VLSI chips, it is crucial that all templates have a certain degree of robustness, since their values cannot be guaranteed to be reproduced exactly by the analog circuit.

Various definitions of robustness [13, 29, 30, 36] exist. We define the vector \( p \) to contain all \( m \) non-zero\(^4\) entries in a template set \( T \), with the center element of the \( A \)-template as its first element \( (p_1 := a_c) \) and the other \( m - 1 \) elements in arbitrary order. We refer to the final output of a CNN programmed with \( p \) by \( y^*(p) \).

**Definition 2.2 (Absolute robustness.)**

The absolute robustness \( \epsilon \) of a template set is

\[
\epsilon(p) = \max_\alpha \{ \alpha \mid y^*(p) = y^*(p + \alpha \pm u) \forall \pm u \in \mathbb{R}^m \}.
\]

Hardware tolerance effects due to physical and manufacturing imperfections give rise to parameter errors roughly proportional to the absolute value of

\( ^3 \)A loop is when, e.g., some configuration of cells \( \varphi_1 \) is supposed to lead to a configuration \( \varphi_2 \) et vice versa.

\( ^4 \)A zero template entry is assumed to be realized by omitting some circuitry, or by switching or disabling some controlled source, not by nulling. Zero template entries are therefore "precise".

the respective parameter [15]. We therefore consider a relative robustness criterion:

**Definition 2.3 (Relative robustness.)**

The relative robustness $D$ of a template set is

$$D(p) = \max \{ \alpha \mid y^*(p \circ (1 + \alpha I^\pm)) = y^*(p) \ \forall I^\pm \in \mathbb{R}^m \}. \quad (2.1)$$

where

- $\circ$ denotes componentwise vector multiplication.

For the sake of clarity and mathematical tractability, we define a slightly modified template vector $\tilde{p}$ to be

$$\tilde{p}_1 := p_1 - 1 = \alpha - 1; \quad \tilde{p}_i := p_i \quad 2 \leq i \leq m,$$

or, alternatively, $\tilde{p} := p - e_1$, where $e_1$ is the unit vector in direction of increasing $\alpha$.

We assume the desired CNN task to be fully characterized by a set of $\hat{m}$ inequalities for $x(t)$. Utilizing the modified template vector $\tilde{p}$ and matrix-vector notation, the region $\mathcal{R} \subset \mathbb{R}^m$ where a template set operates correctly is then defined to be

$$\mathcal{R} = \{ \tilde{p} \in \mathbb{R}^m \mid (K\tilde{p})_i > 0 \ \forall 1 \leq i \leq \hat{m} \}, \quad (2.2)$$

for a coefficient matrix $K \in \mathbb{R}^{\hat{m} \times m}$ representing the different constellations of $u$ and $x(t)$. (If a negative derivative is prescribed, the sign of the corresponding row in $K$ is adapted.) The strict “greater than” inequality may also be a “greater than or equal to” inequality for at most $\hat{m} - 1$ of these inequalities. Note that a value of zero in $K$ is only possible in case of zero initialization or zero boundary values.

By means of the set

$$\mathcal{R}' = \{ \tilde{p} \in \mathbb{R}^m \mid (K\tilde{p})_i \geq 0 \ \forall 1 \leq i \leq \hat{m} \} \supset \mathcal{R}, \quad (2.3)$$

which includes the boundary of $\mathcal{R}$, we introduce the term safety margin; we will denote it by $\gamma(\mathcal{T})$ and formally define it to be

$$\gamma(\mathcal{T}) = \gamma(\tilde{p}) = \min_{1 \leq i \leq \hat{m}} \{ (K\tilde{p})_i \}, \quad \tilde{p} \in \mathcal{R}'. \quad (2.4)$$

---

5If none of the inequalities were strict, a template for which $x \equiv 0$ would be allowed. No operation would be performed, since the initial state would be the equilibrium.
2.2. The Robustness of a CNN Template Set

The *absolute* and the *relative robustness* may now be expressed as

\[
\epsilon(\mathcal{T}) = \frac{\gamma(\mathcal{T})}{m} \quad \text{and} \quad D(\mathcal{T}) = \frac{\gamma(\mathcal{T})}{\| \mathcal{T} \|_1},
\]

respectively. \( \| \mathcal{T} \|_1 \) is the \( L_1 \)-norm of the template vector,

\[
\| \mathcal{T} \|_1 = |I| + \sum_{i=1}^{9} |a_i| + |b_i|.
\]

Note that these definitions of robustness are completely deterministic – they are not based on any assumptions on the probability density of the perturbation of the template vector. Hence, no statistical techniques such as Monte Carlo simulations are involved. Another deterministic definition of relative template robustness was proposed in [37] (adapted to our notation):

\[
D(\mathbf{p}) = \max_{\alpha} \left\{ \alpha \mid \| \Delta \mathbf{p} \|_2 = \alpha \| \mathbf{p} \|_2 \quad \Rightarrow \quad \mathbf{p} + \Delta \mathbf{p} \in \mathcal{R} \right\}. \tag{2.7}
\]

In this definition, the perturbation of all parameters is assumed to be *relative to the Euclidean norm of the template vector*, whereas our definition (2.1) is based on an "individual" perturbation relative to the respective parameter. Accordingly, the neighborhood of \( \mathbf{p} \) that is examined to decide whether \( \mathbf{p} \) has a sufficient degree of robustness is a hypersphere in (2.7) and a hypercuboid in the case of (2.1). Our definition 2.3 models relative deviations more accurately — it is unnatural to assume that a (small) parameter changes its sign due to relative perturbations, which could occur in (2.7) in the case of a large norm \( \| \mathbf{p} \|_2 \).

It may seem tedious to establish a system (2.2) for a task with a high connectivity. To reduce the dimension \( m \), the system can be recast in a form where parameters known \textit{a priori} to be identical are represented by a single variable. Since most highly connected tasks exhibit such isotropic behavior, this results in a manageable system. The new matrix \( \tilde{K} \) will then be in \( \mathbb{Z}^n \times \tilde{m} \).

### 2.2.2 Template Scaling

From (2.4) and (2.5), it follows that \( \epsilon(q \tilde{\mathbf{p}}) = q \epsilon(\tilde{\mathbf{p}}) \), i.e., by scaling the template vector \( \tilde{\mathbf{p}} \) by a factor of \( q \), we achieve proportionally higher *absolute robustness* (cf. [38]).
For the relative robustness, we obtain
\[
D(\tilde{p}) = \frac{\gamma(\tilde{p})}{\|\tilde{p}\|_1 + 1} \quad \Rightarrow \quad D(q \tilde{p}) = \frac{q \gamma(\tilde{p})}{q \|\tilde{p}\|_1 + 1} = \frac{\gamma(\tilde{p})}{\|\tilde{p}\|_1 + \frac{1}{q}},
\] (2.8)
where we have made use of the fact that \(\|\mathcal{T}\|_1 = \|\tilde{p}\|_1 + 1\). Hence, the relative robustness is strictly monotonically increasing with increasing \(q\), but it is upperbounded by \(\frac{\gamma(\tilde{p})}{\|\tilde{p}\|_1}\).

### 2.3 Direct Design of Optimally Robust Templates

#### 2.3.1 Problem Statement

The template optimization problem can now be stated as follows:
\[
p_{\text{opt}} = \arg \max_{p} \{ D(p) \mid y^*(p) = y_d \},
\] (2.9)
where \(y_d\) is the desired output.

With the result from the previous section, it is now easily seen that optimization with respect to relative robustness implies increasing \(\gamma(\tilde{p})\), while keeping \(\|\tilde{p}\|_1\) small. Template scaling by large factors does not improve the robustness significantly, and has the disadvantage of resulting in larger template values which may not be realizable on the CNN chip.

The design of a template with maximum robustness is in fact a design centering problem, since \(\tilde{p}_{\text{opt}}\) is, in some sense, “centered” in \(\mathcal{R}\). Formulated more precisely, the problem is to find a template set \(\mathcal{T}_{\text{opt}}\) (or \(\tilde{p}_{\text{opt}}\)) having the same safety margin in all its inequalities,
\[
(K \cdot \tilde{p}_{\text{opt}}(\gamma))_i = \gamma, \quad 1 \leq i \leq \hat{m},
\] (2.10)
assuming that the system of inequalities is non-redundant in the sense of the following definition.

**Definition 2.4 (Non-redundant set of inequalities.)**

A system \((Kx)_i > 0\) is non-redundant, if every row in \(K\) contributes to a diminution of the solution space, or, equivalently, if no inequality in the system can be removed without affecting the solution space.

In the next subsection, we show how redundant inequalities are found and eliminated from a general set of linear and homogeneous inequalities.
2.3.2 Elimination of Redundant Inequalities

Definition 2.5 (Positive linear combination.)
A positive linear combination of a set of vectors $x_1, \ldots, x_m$ is a linear combination with solely non-negative coefficients. To denote the subspace spanned by positive linear combinations of $x_1, \ldots, x_m$, we use

$$\langle x_1, \ldots, x_m \rangle^+.$$ 

Lemma 2.1 (Determination of redundant inequalities.)
In a system $(Kk)_i > 0$, $K \in \mathbb{R}^{n \times m}$, the redundant inequalities are those which can be expressed as positive linear combinations of others. Hence all row vectors $k_i$ are redundant for which

$$\exists \lambda \in (\mathbb{R}^{m})^n \text{ such that } k_i = \sum_{j=1}^{n} \lambda_j k_j.$$ 

Proof: Using $k_1, k_2, \ldots, k_n$ to denote the rows of $K$, we note that the system is homogeneous in the sense that all hyperplanes $k_i x_i = 0$ intersect at the origin. If one of the inequalities, say $k_i^T \tilde{p} > 0$, can be expressed as the sum of positive multiples of some others, then $k_i^T \tilde{p} > 0$ is trivially satisfied and therefore redundant. This argument has a geometrical interpretation: The vectors $k_i$ normal to the hyperplanes $k_i^T \tilde{p} = 0$ point in the direction of the solution subspace. If one of them, say $k_i$, points into the positive subspace spanned by the others, $\langle k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_n \rangle^+$, this signifies that the inequality defined by $k_i$ is redundant since it is always satisfied when the others hold. Hence it can be eliminated without affecting the solution space. A possible constellation in $\mathbb{R}^2$ is depicted in Fig. 2.1, where $k_3$ is redundant.

Remark: Matrix algebra may be used to determine which row vectors are positive linear combinations of others. In practice, however, the redundant equations can often be eliminated simply by inspection or excluded a priori.

2.3.3 The Exact and Direct Analytical Solution

Using the concepts introduced above, it turns out that the optimally robust template set can be calculated analytically in a rather elegant manner. The method
of solving (2.10) depends on the number of non-zero template parameters \(m\), the number of inequalities \(\tilde{m}\), and the rank of \(K\). If \(K\) has full rank, i.e., \(\text{rank} K = \min\{m, \tilde{m}\}\), and \(\tilde{m} \geq m\) (there are at least as many inequalities as parameters), the system can be solved in the least squares sense. Nevertheless, the solution will always be exact as long as the system is consistent. For this case, the following theorem presents the solution for the robust template design problem.

**Theorem 2.1 (Optimally robust template design.)**

Assuming that

\[
(K \cdot \tilde{p})_i > 0, \quad 1 \leq i \leq m, \quad K \in \mathbb{R}^{\tilde{m} \times m}, \quad \tilde{p} \in \mathbb{R}^{\tilde{m}},
\]

is a set of non-redundant inequalities characterizing a CNN task, the optimally robust template vector \(\tilde{p}_{\text{opt}}\) as a function of a scaling parameter \(q\) is

\[
\tilde{p}_{\text{opt}}(q) = q (K'K)^{-1}K'1^{\tilde{m}}.
\]

(\(1^{\tilde{m}}\) denotes the vector in \(\mathbb{R}^{\tilde{m}}\) with all its components +1.) We define \(\Psi := \tilde{p}_{\text{opt}}(1)\), i.e., the template with a safety margin of 1, to be the mother template of a particular task.
The relative robustness can then be expressed as

\[ D(\hat{p}_{\text{opt}}(q)) = \frac{1}{\|\Psi\|_1 + \frac{1}{q}}, \quad (2.13) \]

where \( q \) equals the safety margin, and the theoretical maximum \( \hat{D} \) for the achievable robustness is

\[ \hat{D} = \lim_{q \to \infty} D(\hat{p}_{\text{opt}}(q)) = \frac{1}{\|\Psi\|_1}. \quad (2.14) \]

An optimally robust template set is a template set that has maximum robustness for a given norm \( \|T\|_1 \).

**Proof:** (2.12) is the solution of (2.10) in the least squares sense. We project \( \mathbf{K} \) into the subspace of dimension \( m \times m \) by multiplication with \( \mathbf{K}^t \) from the left and solve the new system \( \mathbf{K}^t \hat{\mathbf{K}}_{\text{opt}}(q) = q \mathbf{K}^t \mathbf{1}^{\hat{m}} \), yielding a solution in the least squares sense. If \( \mathbf{Kp} = q \mathbf{1}^{\hat{m}} \) has an exact solution, then it is \( \hat{\mathbf{p}}_{\text{opt}} \). If there is no exact solution, but the system (2.11) is not inconsistent, then there is always a point \( \hat{\mathbf{p}}_{\text{opt}} \) with the lowest mean square distance between \( \mathbf{Kp} \) and \( q \mathbf{1}^{\hat{m}} \).

However, it is not guaranteed that the template \( \hat{\mathbf{p}}_{\text{opt}}(q) \) we get from (2.12) in fact solves (2.11) — this has to be checked. If it does not, then the system is not consistent.

Note that by this solution, the case \( m = \hat{m} = \text{rank}(\mathbf{K}) \), where \( \mathbf{K} \) can be inverted directly, is covered as well. The robustness is derived from (2.5). \( \square \)

If \( \mathbf{K} \) has a rank less than \( m \), the set of solutions, which then has higher dimension, may be determined using \( QR \)-decomposition or by reduction of the number. Three different cases have to be considered:

I. \( \text{rank}(\mathbf{K}) < m = \hat{m} \).

   This system has a solution of dimension \( (m - \text{rank}(\mathbf{K})) \); it can also be solved in straightforward manner using \( QR \)-decomposition, for example.

II. \( \text{rank}(\mathbf{K}) < m < \hat{m} \).

   In this case, the number of parameters is to be reduced by matrix algebra to achieve \( m = \text{rank}(\mathbf{K}) \). The system can then be solved using the method proposed for the previous case.

III. \( \text{rank}(\mathbf{K}) < m < \hat{m} \).

   If the number of parameters exceeds the number of inequalities, there is always a solution which has the dimension \( (\hat{m} - \text{rank}(\mathbf{K})) \).
Definition 2.6 (The principal axis of a CNN task.)
Assuming that a CNN task is characterized by a set of inequalities (2.2), the principal axis points in the direction of the unit vector \( \Psi / \| \Psi \|_1 \) and contains all templates \( \tilde{p} = \tilde{p}_{opt}(q) \) with \( q > 0 \).

The principal axis of a task comprises all templates that are optimal with respect to robustness. Hence scaling such a template does not affect its property of being optimal.

Corollary 2.1 (Unboundedness of \( \mathcal{R} \).
\( \mathcal{R} \) is infinite. It is spanned by
\[
(K'K)^{-1}K' \lambda \quad \forall \lambda \in (\mathbb{R}^+)^\tilde{m}.
\] (2.15)

Proof: In every case, \( \mathcal{R} \) is not bounded along the principal axis. (2.11) is satisfied for any vector \( \lambda \) with solely positive components. \( \square \)

For all CNN chips, there is an upper bound for \( \| \mathcal{T} \|_1 \). The next corollary shows how to find the optimum template under this constraint.

Corollary 2.2 (Optimum template for \( \| \mathcal{T} \|_1 \leq \beta \).
Under the constraint \( \| \mathcal{T} \|_1 \leq \beta \), the optimum template is
\[
\tilde{p}_{opt} \big|_{\| \mathcal{T} \|_1 \leq \beta} = (\beta - 1) \frac{\Psi}{\| \Psi \|_1}
\] (2.16)
with a relative robustness of
\[
D(\tilde{p}_{opt} \big|_{\| \mathcal{T} \|_1 \leq \beta}) = \frac{1}{\| \Psi \|_1} \left( 1 - \frac{1}{\beta} \right).
\] (2.17)

Proof: Expressed by \( \tilde{p} \), the constraint is \( \gamma \| \tilde{p} \| + 1 \leq \beta \). Solving for \( \gamma \) and inserting in (2.12) and (2.5) yields the above results. \( \square \)

Given the constraint \( \beta \), (2.17) tells us directly how far we are from the theoretical optimum. For \( \beta = 10 \), for example, we achieve 90% of this optimum.

Corollary 2.3 (Initialization for uncoupled tasks.)
If, for an uncoupled bipolar CNN task, the initial state is chosen to be \( x(0) = \pm 1 \), \( x(0) = 0 \), or \( x(0) = u \), then the space of optimally robust templates according to (2.10) is not necessarily a single point, but may have dimension
2.3. Direct Design of Optimally Robust Templates

\( \geq 1. \) Depending on the initialization, there is a degree of freedom in the \((\tilde{p}, b_c)\) or in the \((\tilde{p}, I)\) plane, subject to the constraint \( \tilde{p} \geq 0 \) in order to guarantee bipolar output \( (\tilde{p} := a_c - 1) \).

\[ x(0) = 0 : \quad \tilde{p} = 0 \text{ is the optimum choice.} \]

Proof: \( \tilde{p} \) does not appear in the parameter vector \( \tilde{p} \). Any \( \tilde{p} \geq 0 \) does not influence the functionality of the template. Thus we set \( \tilde{p} = 0 \) for maximum relative robustness.

\[ x(0) = -1 : \quad \text{Let } c := -\tilde{p} + I. \text{ From (2.10), we get only a value for } c, \text{ but not for the individual parameters } \tilde{p} \text{ and } I. \text{ If } c < 0 \text{ we may choose } \tilde{p} \in [0,-c] \text{ and } I = \tilde{p} + c \text{ without affecting the robustness. If } c \geq 0, \text{ then } \tilde{p} = 0 \text{ and } I = c \text{ is the only optimum solution.} \]

Proof: In the matrix \( K \), \( -\tilde{p} \) and \( I \) have the same coefficient in every row, since for initialization with \(-1\), \(( -1)(-\tilde{p})\) plays the role of an additional bias at \( t = 0 \). Hence the matrix \( K \) is singular, and \( \tilde{p} \) and \( I \) are only determined by the value of \( c \) and \( \tilde{p} > 0 \). For maximum relative robustness, \( \tilde{p} + |I| = \tilde{p} + |\tilde{p} + c| \) has to be minimal, which results in the above solution space.

\[ x(0) = 1 : \quad \text{Let } c := \tilde{p} + I. \text{ As in the previous case, only } c \text{ is determined by (2.10). If } c > 0, \text{ all } \tilde{p} \in [0,c] \text{, } I = c - \tilde{p} \text{ are optimum values, whereas for } c \leq 0 \text{, only one optimum solution exists, namely } \tilde{p} = 0, \text{ } I = c. \]

Proof: For positive initialization, \( \tilde{p} \) can be considered as additional bias at \( t = 0 \), and \( \tilde{p} \) and \( I \) have the same coefficients in \( K \). Thus only \( c = \tilde{p} + I \) is fixed by (2.10) — depending on \( c \), the minimization of \(|\tilde{p}| + |I|\) yields the above solution(s).

\[ x(0) = u : \quad \text{Let } c := \tilde{p} + b_c. \text{ The optimum solutions for } \tilde{p} \text{ and } b_c \text{ are } \tilde{p} \in [0,c], \text{ } b_c = c - \tilde{p} \text{ if } c > 0, \text{ and } \tilde{p} = 0, \text{ } b_c = c \text{ if } c < 0. \]

Proof: Similar reasoning as in the previous case applies. Simply replace \( I \) by \( b_c \), since, \( \tilde{p} \) and \( b_c \) have the same coefficients in \( K \) for input initialization.

Remark: When applying theorem 2.1 to uncoupled tasks, it is advantageous not to include \( \tilde{p} \) (and \( b_c \) or \( I \), respectively) in the matrix \( K \), but just \( c \) (as defined in corollary 2.3) in order to get a regular matrix, and then to determine \( \tilde{p} \) and \( b_c \) or \( I \), respectively, after having calculated the optimum value for \( c \). However, \( QR \)-decomposition would of course lead to the same result with a parameter vector including \( \tilde{p} \), \( b_c \), and \( I \).
2.4 Examples

Intentionally, we make only minimal use of *a priori* knowledge, and merely assume symmetry of isotropic tasks. For those examples, we use the reduced coefficient matrix $\tilde{K} \in \mathbb{Z}^{n \times m}$ instead of $K \in \mathbb{R}^{n \times m}$. We consider the “actual” image to be in black (+1) on a white (−1) “background”. The boundary condition for both state and input is assumed to be −1 throughout this section. The tasks are described in detail in Appendix A.

The functionality of the templates proposed in this section may be verified using the simulator available on World Wide Web under http://www.isi.ee.ethz.ch/~haenggi/CNN_web/CNNsim_adv.html [39]. The simulator is briefly characterized in Appendix B.

Example 2.1 (Uncoupled horizontal line detection)

**Prototype.**

$$A = [\tilde{p} + 1]; \quad B = [s \ b_c \ s]; \quad I = z; \quad x(0) = u$$

**Definition of the Task.** With $c := \tilde{p} + b_c$, we get

<table>
<thead>
<tr>
<th></th>
<th>$u = y(0)$</th>
<th>$y^*$</th>
<th>$\dot{x}(0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>$\circ \bullet \circ$</td>
<td>$\circ \circ \circ$</td>
<td>$-c + 2s - z &gt; 0$</td>
</tr>
<tr>
<td>(2)</td>
<td>$\circ \bullet \bullet$</td>
<td>$\circ \bullet \bullet$</td>
<td>$c + z \geq 0$</td>
</tr>
<tr>
<td>(3)</td>
<td>$\bullet \circ \circ$</td>
<td>$\bullet \circ \circ$</td>
<td>$c - 2s - z \geq 0$</td>
</tr>
<tr>
<td>(4)</td>
<td>$\bullet \circ \circ$</td>
<td>$\bullet \circ \circ$</td>
<td>$c - z \geq 0$</td>
</tr>
</tbody>
</table>

**Removal of Redundant Inequalities.** The last row vector, $k_4$ is the sum of the other three and should be eliminated. No additional non-redundant inequalities (i.e., cell configurations) can be included.

**Solution.**

$$\tilde{p}_{opt} = \gamma \tilde{K}^{-1} 1^3 \quad \Rightarrow \quad c = 2\gamma, \quad s = \gamma, \quad z = -\gamma$$

$\tilde{p} \in [0, 2\gamma]$ leads to the extrema

$$A = [2\gamma + 1]; \quad B = [\gamma \ 0 \ \gamma]; \quad I = -\gamma \quad \text{and}$$

$$A = [1]; \quad B = [\gamma \ 2\gamma \ \gamma]; \quad I = -\gamma$$
From (2.14) we deduce that the maximum achievable robustness is $1/5 = 20\%$.

**Example 2.2 (Shadowing)**

**TEMPLATE PROTOTYPE.**

$$A = [s \; \tilde{p} + 1 \; q]; \; B = 0; \; I = z; \; x(0) = u$$

**DEFINITION OF THE TASK.**

<table>
<thead>
<tr>
<th>$y(t)$</th>
<th>$y(t + T)$</th>
<th>$\dot{x}(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>o o o o</td>
<td>-s - \tilde{p} + q + z &gt; 0</td>
</tr>
<tr>
<td>(2)</td>
<td>o o o o</td>
<td>s - \tilde{p} + q + z &gt; 0</td>
</tr>
<tr>
<td>(3)</td>
<td>o o o o</td>
<td>s + \tilde{p} + q + z \geq 0</td>
</tr>
<tr>
<td>(4)</td>
<td>o o o o</td>
<td>-s + \tilde{p} + q + z \geq 0</td>
</tr>
<tr>
<td>(5)</td>
<td>o o o o</td>
<td>s + \tilde{p} - q - z \geq 0</td>
</tr>
<tr>
<td>(6)</td>
<td>o o o o</td>
<td>-s + \tilde{p} - q - z \geq 0</td>
</tr>
<tr>
<td>(7)</td>
<td>o o o o</td>
<td>s + \tilde{p} - q + z \geq 0</td>
</tr>
<tr>
<td>(8)</td>
<td>o o o o</td>
<td>-s + \tilde{p} - q + z \geq 0</td>
</tr>
</tbody>
</table>

**REMOVAL OF REDUNDANT INEQUALITIES.** We find $k_3 = k_2 + k_6 + k_7$ and $k_4 = k_2 + k_6 + k_8$. The remaining 6 inequalities are non-redundant.

**SOLUTION.** Since we end up with a system with $\tilde{m} = 6 > m = 4$, we have to solve it in the least squares sense, which yields

$$\tilde{p}_{\text{opt}} = (K'K)^{-1}K'1^6 \implies \tilde{p} = q = z = \gamma, \ s = 0,$$

which is in fact the well-known shadowing template

$$A = [0 \; \gamma + 1 \; \gamma]; \; B = [0]; \; I = \gamma.$$

The maximum achievable robustness is $1/3 = 33\%$.

**REMARK:** This method may also be applied to check whether there exists a shadowing template with a symmetrical $A$-template, and, if so, to determine the most robust solution. With the prototype

$$A = [s \; \tilde{p} + 1 \; s]; \; B = [q \; b_c \; r]; \; I = z; \; x(0) = -1,$$
and $c := \tilde{p} - z$, we find the non-redundant set

<table>
<thead>
<tr>
<th>$u$</th>
<th>$y(t)$</th>
<th>$\rightarrow$</th>
<th>$y(t+T)$</th>
<th>$\dot{x}(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>● ● ●</td>
<td>○ ○ ○</td>
<td>$\rightarrow$</td>
<td>○ ● ●</td>
<td>$-c - 2s + q + b_c + r &gt; 0$</td>
</tr>
<tr>
<td>● ● ○</td>
<td>○ ○ ○</td>
<td>$\rightarrow$</td>
<td>○ ● ○</td>
<td>$-c - 2s + q + b_c - r &gt; 0$</td>
</tr>
<tr>
<td>● ○ ○</td>
<td>○ ○ ○</td>
<td>$\rightarrow$</td>
<td>● ○ ○</td>
<td>$c - q + b_c + r \geq 0$</td>
</tr>
<tr>
<td>○ ○ ○</td>
<td>○ ○ ●</td>
<td>$\rightarrow$</td>
<td>○ ● ●</td>
<td>$-c - q - b_c - r &gt; 0$</td>
</tr>
<tr>
<td>● ○ ○</td>
<td>● ○ ●</td>
<td>$\rightarrow$</td>
<td>● ● ●</td>
<td>$-c + 2s + q - b_c - r &gt; 0$</td>
</tr>
</tbody>
</table>

Evaluating $\tilde{p}_{opt} = \gamma \tilde{K}^{-1} \mathbf{1}^5$ and minimizing $|\tilde{p}| + |I|$ yields

$$A = [\gamma 1 \gamma], \quad B = [-\gamma 2\gamma 0], \quad I = 2\gamma$$

with a maximum robustness of $1/7 \approx 14.3\%$.

**Example 2.3 (Connected component detection)**

**TEMPLATE PROTOTYPE.**

$$A = [s \tilde{p} + 1 q]; \quad B = 0; \quad I = z; \quad x(0) = u$$

**DEFINITION OF THE TASK.**

<table>
<thead>
<tr>
<th></th>
<th>$y(t)$</th>
<th>$\rightarrow$</th>
<th>$y(t+T)$</th>
<th>$\dot{x}(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>● ○ ○</td>
<td>$\rightarrow$</td>
<td>● ● ○</td>
<td>$-\tilde{p} + s - q + z &gt; 0$</td>
</tr>
<tr>
<td>(2)</td>
<td>○ ● ●</td>
<td>$\rightarrow$</td>
<td>○ ○ ●</td>
<td>$-\tilde{p} + s - q - z &gt; 0$</td>
</tr>
<tr>
<td>(3)</td>
<td>● ○ ●</td>
<td>$\rightarrow$</td>
<td>● ○ ●</td>
<td>$\tilde{p} - s - q - z \geq 0$</td>
</tr>
<tr>
<td>(4)</td>
<td>○ ○ ○</td>
<td>$\rightarrow$</td>
<td>○ ● ○</td>
<td>$\tilde{p} - s - q + z \geq 0$</td>
</tr>
<tr>
<td>(5)</td>
<td>● ● ●</td>
<td>$\rightarrow$</td>
<td>● ● ●</td>
<td>$\tilde{p} + s + q + z \geq 0$</td>
</tr>
<tr>
<td>(6)</td>
<td>○ ○ ○</td>
<td>$\rightarrow$</td>
<td>○ ○ ○</td>
<td>$\tilde{p} + s + q - z \geq 0$</td>
</tr>
<tr>
<td>(7)</td>
<td>● ○ ●</td>
<td>$\rightarrow$</td>
<td>● ● ○</td>
<td>$\tilde{p} + s - q + z \geq 0$</td>
</tr>
<tr>
<td>(8)</td>
<td>○ ○ ●</td>
<td>$\rightarrow$</td>
<td>○ ○ ●</td>
<td>$\tilde{p} + s - q - z \geq 0$</td>
</tr>
</tbody>
</table>

**REMOVAL OF REDUNDANT INEQUALITIES.** We eliminate the last two rows since $k_7 = k_1 + k_4 + k_6$ and $k_8 = k_2 + k_3 + k_5$.

**SOLUTION.** Solving this system yields

$$\tilde{p}_{opt} = (K^T K)^{-1} K^T \mathbf{1}^6 \implies \tilde{p} = s = \gamma, \quad q = -\gamma, \quad z = 0,$$

with a maximum achievable robustness of $1/3 = 33\%$.

**Example 2.4 (Global connectivity detection)**

**TEMPLATE PROTOTYPE.** This task does not distinguish between vertically or horizontally connected cells — it exhibits isotropic behavior. The off-center
2.5. Making Templates Optimally Robust

entries in $A$ and $B$ are therefore a priori assumed to be identical. To ensure that the template will operate on both white images on a black background and black images on a white background, the bias is set to zero.

$$A = \begin{bmatrix} 0 & s & 0 \\ s & \hat{p} + 1 & s \\ 0 & s & 0 \end{bmatrix}; \quad B = \begin{bmatrix} 0 & q & 0 \\ q & b_c & q \\ 0 & q & 0 \end{bmatrix};$$

$I = 0$; $x(0) =$ bipolar image

DEFINITION OF THE TASK.

<table>
<thead>
<tr>
<th>$u$</th>
<th>$y(t)$</th>
<th>$y(t+T)$</th>
<th>$\dot{x}(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>$\bullet \circ \circ \circ \circ$</td>
<td>$\circ \circ \circ \circ$</td>
<td>$\hat{p} + b_c + 4s + 4q \geq 0$</td>
</tr>
<tr>
<td>(2)</td>
<td>$\circ \circ \circ \circ \circ \circ$</td>
<td>$\circ \circ \circ \circ$</td>
<td>$\hat{p} + b_c - 4s - 4q \geq 0$</td>
</tr>
<tr>
<td>(3)</td>
<td>$\circ \circ \circ \circ \circ \circ$</td>
<td>$\circ \circ \circ \circ$</td>
<td>$-\hat{p} - b_c - 2s - 4q &gt; 0$</td>
</tr>
<tr>
<td>(4)</td>
<td>$\bullet \circ \circ \circ \circ \circ$</td>
<td>$\circ \circ \circ \circ \circ$</td>
<td>$\hat{p} - b_c - 4s - 4q \geq 0$</td>
</tr>
<tr>
<td>(5)</td>
<td>$\circ \circ \circ \circ \circ \circ$</td>
<td>$\circ \circ \circ \circ \circ$</td>
<td>$\hat{p} + b_c + 2s + 2q \geq 0$</td>
</tr>
<tr>
<td>(6)</td>
<td>$\bullet \circ \circ \circ \circ \circ$</td>
<td>$\circ \circ \circ \circ \circ \circ$</td>
<td>$-\hat{p} - b_c - 4q &gt; 0$</td>
</tr>
</tbody>
</table>

REMOVAL OF REDUNDANT INEQUALITIES. Inequalities (1)–(4) constitute a regular non-redundant matrix $\tilde{K}$. Other constellations like $k_5 = \frac{3}{4}k_1 + \frac{1}{4}k_2$ and $k_6 = k_1 + 2k_3$ are clearly redundant.

SOLUTION.

$$\hat{p}_{opt} = \gamma \tilde{K}^{-1}1^4 \implies \hat{p} = s = \gamma, \quad q = -\gamma, \quad b_c = 0,$$

in agreement with the template found by stochastic optimization [40], where $\gamma = 2$ with a robustness of $2/19 = 10.5\%$. The maximum robustness for this task is $\gamma/\|\hat{p}_{opt}\|_1 = 1/9 = 11.1\%$.

2.5 Making Templates Optimally Robust

In this section, we assume that we have, either by intuition, by means of any learning method, or in a template library, found a correctly operating template vector $T$, or the reduced version $p$, respectively. The problem now is to find the optimally robust template performing the same operation. We again
assume that the desired CNN task is fully characterized by defining the sign
of $\dot{x}$ for all possible bipolar configurations of input and output values of the
neighboring cells, including the center cell itself. Conversely, the behavior
of a CNN programmed with a given template vector $p$ may be analyzed by
calculating $\dot{x}$ for all configurations.
We code the value of the input and output of the neighbors in a bipolar vector
$v \in \mathbb{B}^m$. The derivative of a cell’s state, given any constellation of its neigh-
bors, may now be written as

$$\dot{x} \big|_{v} = v^T \cdot \tilde{p} \quad \forall v \in \mathbb{B}^m. \quad (2.18)$$

In the case of a non-zero bias $I$, the number of constellations is $n = 2^{m-1}$,
otherwise $n = 2^m$, since the bias is not multiplied by any cell-dependent value.
Introducing an index $i$, $1 \leq i \leq n$, for all possible $v$, we define a coefficient
matrix $K \in \mathbb{B}^{n \times m}$ to be

$$K = \begin{bmatrix}
\text{sgn}(v_1^T \tilde{p}) v_1^T \\
\text{sgn}(v_2^T \tilde{p}) v_2^T \\
\vdots \\
\text{sgn}(v_n^T \tilde{p}) v_n^T
\end{bmatrix}, \quad (2.19)$$

and we end up with the same linear and homogeneous system of inequalities
as in the previous section,

$$(K \tilde{p})_i > 0 \quad \forall 1 \leq i \leq n. \quad (2.20)$$

After this analysis step we proceed in exactly the same manner as in Sec. 2.3.3:
after the elimination of the redundant inequalities, theorem 2.1 is applicable.

**Example 2.5 (Shadowing)**
The template set

$$A = \begin{bmatrix} 0 & 2 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \end{bmatrix} \quad I = 2$$

is often used for the shadow projection task. Since the first entry in the
$A$-template is zero, only 4 configurations of neighboring cells (including
the center cell itself) are to be investigated, $\tilde{p} = [1 \ 2 \ 2]'$. The configurations may
occur at any time during the CNN transient, not necessarily at the beginning.
The relative robustness of this template is

$$D(\tilde{p}) = \min_i \frac{(K \tilde{p})_i}{\|T\|_1} = \frac{k_1^T \tilde{p}}{6} = \frac{1}{6} \approx 16.7\%.$$
2.6 Optimum Templates for a Specific CNN Implementation

The non-redundant matrix $\mathbf{K}$ is square and regular and may be inverted directly which yields

$$\mathbf{p}_{\text{opt}}(\gamma) = \mathbf{K}^{-1} \mathbf{1}^T = [1 \ 1 \ 1]^T$$

$$\implies A = [0 \ 1 + \gamma \ \gamma] \quad B = [0] \quad I = \gamma$$

with a robustness of $D(\mathbf{p}_{\text{opt}}(\gamma)) = (3 + 1/\gamma)^{-1}$. The theoretical upper bound is $\lim_{\gamma \to \infty} D(\mathbf{p}_{\text{opt}}(\gamma)) \approx 33.3\%$.

2.6 Optimum Templates for a Specific CNN Implementation

In this section, we briefly characterize a prototype of a modularly extendable $g_m$-C implementation of the CNN universal chip [41], focusing on its deviations from the mathematical model. As an application of the theory developed in the last two sections, templates are designed that optimally overcome the limitations of this particular chip.

2.6.1 A Modular $g_m$-C Programmable CNN Implementation

The chip is designed and fabricated in a 0.8$\mu$m double-metal CMOS process. Each chip containing $3 \times 2$ cells operates from $\pm 3$ V, the active area is $2.4 \times 2.5$ mm, and the power consumption per cell is 10mW. The configuration of $3 \times 2$ cells was chosen merely so that a 40 pin package would suffice – there is no inherent limit to the number of cells on a single die.

It is of course desirable to have as many cells in the network as possible. However, if the entire network must be on a single IC, cost, yield, and electrical considerations will limit the achievable network size. In the design of the chip, provisions have been made to have the ability to construct an arbitrarily large CNN in a modular way, requiring off-chip analog connections. The parasitic capacitance associated with the off-chip wiring requires that the state capacitor $C$ be increased so as to "swamp" the parasitic capacitances. The cell $RC$ time constant will increase in order to accommodate the delay caused by inter-chip connections. These connections cannot be implemented using circuits relying on differential operation, because that would make the number of inter-chip connecting wires impractical. Twenty analog pins are
Table 2.1: Implementation data of the chip.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCCS errors, intrachip</td>
<td>&lt; 5%</td>
</tr>
<tr>
<td>VCCS errors, interchip</td>
<td>8%</td>
</tr>
<tr>
<td>VCCS linearity (±400V input)</td>
<td>6%</td>
</tr>
<tr>
<td>Saturation level error</td>
<td>&lt; 5%</td>
</tr>
<tr>
<td>Mismatch in $u_{ij}$</td>
<td>&lt; 5%</td>
</tr>
<tr>
<td>Power consumption per cell</td>
<td>10mW</td>
</tr>
<tr>
<td>Power for tuning circuits</td>
<td>13mW</td>
</tr>
<tr>
<td>Cell $RC$ time constant</td>
<td>2.5μs</td>
</tr>
<tr>
<td>Interchip $RC$ mismatch</td>
<td>&lt; 50%</td>
</tr>
<tr>
<td>Available state swing</td>
<td>±3 units</td>
</tr>
<tr>
<td>Cell area</td>
<td>0.78mm²</td>
</tr>
</tbody>
</table>

required to be able to connect an arbitrary number of chips to form a large CNN array. Table 2.1 presents the implementation data of the chip.

The design of a CNN is dominated by the implementation of the connections between cells, represented by voltage controlled current sources (VCCSs). Each cell has 19 VCCSs, which must all be of variable strength in a programmable CNN.

The setting of each VCCS, i.e. the connection weight between the cells, is digitally represented by a binary code, as a small number of bits. In this way, step-wise programmable templates are implemented, and the VCCSs perform an implicit digital-to-analog conversion. Each VCCS (realizing a template value) comprises several unit and half-unit sized differential pairs, and a common set of current mirrors. Every VCCS is therefore a type of primitive (nonlinear) digital-to-analog converter. The center $A$ and $B$ sources have eight values available, the off-center $A$ and $B$ sources four, and the constant (bias) source $I$ has fifteen, each of them with positive and negative sign (Table 2.2). The set of all possible combinations of template parameters will be denoted by $T$. Step-wise programmability also allows each VCCS of each cell to be programmed differently, which would be virtually impossible to implement with analog control lines.

The state capacitor $C$ in every CNN cell must normally be charged to a certain value before the CNN transient takes place, i.e., the initial state has to be set. The current sources for the $A$, $B$, and $I$ weights are switched off for $t < 0$, and the resistor is disconnected. $C$ is charged to an initial value $x_{ij}(0)$. At $t = 0$, the current sources and $R$ are reconnected. Nearly any implemen-
2.6. Optimum Templates for a Specific CNN Implementation

Optimum Templates for a Specific CNN Implementation

Optimum Templates for a Specific CNN Implementation

The output nonlinearity cannot be obtained by making use of the limiting characteristics of the differential pairs in the VCCSs, because the point at which the transconductors saturate (and the saturation level itself) is not controlled – the single degree of freedom of electrical tuning is used to tune the VCCSs to unit strength. A separate circuit approximating the piecewise linear $\text{sat}(\cdot)$ function is therefore used, consisting of a differential pair with non-saturated MOS transistor loads.

2.6.2 Deriving Optimum Templates for a Specific Chip

Deriving Optimum Templates for a Specific Chip

The chip is intended for bipolar operation in both input and output, and the large (interchip) mismatch in the time constant of the cells further restricts the executable tasks to the locally regular ones. Any locally irregular task is very sensitive against the $RC$ mismatch and is very likely to fail. The robustness theory covers exactly this class of tasks or templates, and is therefore applicable in a straightforward manner.

For any locally regular task, the mother template $\Psi$ has to be matched to the set of available parameter values in Table 2.2. A good initial, chip-specific

<table>
<thead>
<tr>
<th>$A$ and $B$ off-center entries:</th>
<th>$\pm {0, 1, 2, 3, 4}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$ and $B$ center entries $a_c$ and $b_c$:</td>
<td>$\pm {0, 0.5, 1, 1.5, \ldots, 4}$</td>
</tr>
<tr>
<td>Bias $I$:</td>
<td>$\pm {0, 0.5, 1, 1.5, \ldots, 7.5}$</td>
</tr>
</tbody>
</table>

Table 2.2: The sets of programmable template values.

---

6This is the class of tasks that can be characterized by a set of local rules. See Sec. 4.3.3 on p. 81 for details.
guess $\hat{p}_S$ is

$$\hat{p}_S = \arg \min_{\hat{p} \in \mathbb{T}, q > 0} \{ \| q \Psi - \hat{p} \| \} \quad (2.21)$$

for some distance measure $\| . \|$. If this distance is zero for some positive $q$, $\hat{p}_S$ is guaranteed to operate correctly. In the case of a non-zero distance, it has to be checked whether $(K\hat{p}_S)_i > 0$ (cf. (2.11)) holds. If not, the template parameter set $\mathbb{T}$ has to be searched for any point satisfying the system of inequalities, which may turn out to be insufficiently robust. For certain tasks, one might even have to conclude that there is no solution in $\mathbb{T}$, which would signify that this particular operation cannot be carried out on this chip. In most cases, however, chances are good that at least one scaled mother template $q \Psi$ is an element of $\mathbb{T}$, since $\mathbb{T}$ includes multiples of $1/2$ and $\Psi \in \mathbb{Z}^m$.

Considering the specifications of the chip in Table 2.1, we conclude that a robustness of $10\%$ is sufficient. Although in a worst case, the interchip errors introduced by the VCCSs and the saturation error of the output nonlinearity eventually add up to $13\%$, such a constellation is unlikely to occur and, even if it does, will very rarely lead to incorrect behavior since the robustness analysis itself is a worst case analysis: a template with $D = 10\%$ will tolerate a simultaneous perturbation of $10\%$ of all its parameters in any direction.

Deviations from the unit slope in the linear region of the sat($\cdot$) function will not affect the final output $y^*$. In fact, locally regular tasks only require to be strictly monotonically increasing between the saturation levels. Neither does the limited state swing have any influence on $y^*$, since the state is not fed back to neighboring cells.

For the uncoupled horizontal line detector (see Example 2.1 in Section 3.4), we found that

$$A = [2\gamma + 1]; \quad B = [\gamma \ 0 \ \gamma]; \quad I = -\gamma \quad (2.22)$$

is optimal. For any $\gamma \in \{0.5, 1, 1.5\}$, the template can be programmed on the chip. Its robustness is $D(\gamma) = 1/(5 + 1/\gamma) \in \{\frac{1}{7}, \frac{1}{6}, \frac{3}{17}\} \subset [14.3\%, 17.6\%]$. From Corollary 2.3, we know that there is another set of optimum templates,

$$A = [1]; \quad B = [\gamma \ 2\gamma \ \gamma]; \quad I = -\gamma, \quad (2.23)$$

with precisely the same robustness. However, this version of the template is still realizable on the chip for $\gamma = 2$, resulting in $D(2) = 18.1\%$. There is another reason why to prefer (2.23): The initialization $x(0)$ is less precise
than the time-invariant input $u$; consequently, it is good advice to choose the $A$ parameters smaller than the $B$ parameters, if possible.

For the connected component detector (Example 2.3), we get $A = [3\ 4\ -3]$, $I = 0$ ($\gamma = 3$) as the optimum template. Due to the small connectivity or, equivalently, the large number of zero template parameters, its robustness of 30% is much higher than actually needed. As we will show in the next chapter, this degree of freedom can be exploited to design a template that operates faster than a template with maximum robustness.

**Example 2.6 (Hole filling)**
The hole filling template has zero diagonal elements in the $A$ template and zero off-center entries in the $B$ template.

$$A = \begin{bmatrix} 0 & s & 0 \\ s & \tilde{p} + 1 & s \\ 0 & s & 0 \end{bmatrix}, \quad B = [b_c]; \quad I = z; \quad x(0) = 1$$

The analysis of the task yields the optimum solution

$$\tilde{p} + z = \gamma \ (\tilde{p} \geq 0), \quad b_c = 4\gamma, \quad s = \gamma.$$  

with a robustness of 10% for $\gamma = 1$ — larger values for $\gamma$ are not programmable. Note that although this is a coupled task, Corollary 2.3 is applicable due to the spatially invariant initial condition. Any template for which $\tilde{p} + z = \gamma$ is equivalent in terms of robustness. In theory, the relative amount of perturbation of all parameters is assumed to be identical. On the chip, however, this assumption is not quite justified for the bias $I$. The bias is not multiplied by any cell value, it does not suffer from interchip matching errors and is therefore more accurately reproduced on the chip. This reasoning leads to the conclusion that $\tilde{p} = 0$ and $z = \gamma$ is the best solution for this chip.

### 2.7 Conclusions

Mainly due to physical and manufacturing inaccuracies, VLSI implementations of CNN chips may deviate significantly from the ideal mathematical model. The dominant sources of error are the programmable voltage controlled current sources representing the interconnection weights between the cells. To ensure correct operation of the chip, these template parameters have to be designed in such a way that they can tolerate a certain degree of perturbation while still producing the correct output.
In this chapter, we have proposed an exact and analytical approach for the design of robust templates for the class of locally regular CNN tasks. Absolute and relative robustness is defined in a deterministic and easily reproducible manner. The desired task is characterized by a set of inequalities defining the subspace within which all correctly operating templates lie. We have shown that this system of inequalities can be solved directly for the optimally robust template; there is no need for an iterative algorithm. Furthermore, this analytical method provides insight into the dynamics of the CNN and the interaction between different template parameters.

Matrix-vector notation for the coefficient matrix and the vector of non-zero template entries is applicable. With a translation by $-1$ in the direction of the $a_c$ axis, the system turns out to be homogeneous. Redundant rows (i.e., equations that are unnecessary since they do not impose any additional restrictions on the solution space) of the coefficient matrix are to be eliminated in a first step — these row vectors have the property of being positive linear combinations of others. Since the method includes the removal of these redundant inequalities, it is not important to specify exactly the right inequalities and the right number thereof — one may include inequalities for all possible constellations, or restrict oneself to a small set, which is in fact often sufficient to fully characterize a task.

The safety margin specifies by what amount an inequality is satisfied. Optimally robust templates $\mathbf{p}_{opt}$ are those with the same safety margin $\gamma$ in all non-redundant inequalities: $K\mathbf{p}_{opt} = \gamma \mathbf{1}$. The solution depends on the dimension and the rank of $K$. Often, $K$ is regular and we obtain simply $\mathbf{p}_{opt} = \gamma K^{-1} \mathbf{1}$. If the system is overdetermined, we solve it in a least squares sense, if it is underdetermined, we may apply a $QR$-decomposition or reduce the number of parameters. For every $\gamma > 0$, the solution is optimal in the sense that no other more robust template with a smaller or equal $L_1$-norm $\|\mathbf{p}\|_1$ exists.

With increasing $\gamma$, the absolute and relative robustness increase strictly monotonically. The relative robustness is upperbounded for $\gamma \to \infty$. This upper bound is a property of the underlying task, the initial state, and the boundary value; it can be very easily determined and permits the optimization of the initial and boundary conditions for robustness.

Scaling a correctly operating template $\mathbf{p}$ by a positive factor always yields another valid solution, with the absolute robustness being scaled by the same factor. This implies that the subspace of the solutions for a given task is not bounded.
2.7. Conclusions

For VLSI implementations of the CNNs, $\|T\|_1 = \|\hat{p}\|_1 + 1$ is constrained by some upper bound $\beta$. The proposed method directly yields the optimum solution for $\beta \leq \|T\|_1$ and specifies its degree of robustness.

For a given CNN chip prototype with step-wise programmable weights, it has been demonstrated how to apply the robustness theory to design template sets that optimally cope with the chip's nonidealities. Generalizing these results, we conclude that template sets with more than 10 or 11 (including the bias current $I$) non-zero parameters will hardly operate correctly on an analog VLSI chip, and that for templates with lower connectivity, no tweaking of parameters for individual chips is necessary, since optimum templates can be analytically derived for specific implementations of CNN chips.
Chapter 3

CNN Settling Time

3.1 Introduction

When discussing and comparing signal processing devices, their processing speed is always of particular interest. In the case of a nonlinear dynamical system, the processing speed is defined by its settling time, i.e., the time it takes the system to reach its equilibrium state. In this chapter, the settling time of bipolar CNNs is defined and investigated.

3.1.1 The Settling Time of a Bipolar CNN

Definition 3.1 (Settling time.)

For stable planar CNNs with bipolar output, we define the settling time \( T_{S_{ij}} \) of the cell \( C_{ij} \) to be the time it takes the cell to reach its final output value:

\[
T_{S_{ij}} = \min_{t \geq 0} (t \mid y_{ij}(\tau) = y_{ij}^* \quad \forall \tau \geq t)
\]

In a straightforward extension, the settling time \( T_{S} \) of an entire CNN is defined as

\[
T_{S} = \max_{1 \leq i \leq N} \min_{1 \leq j \leq M} (t \mid y(t) = y^*).
\]
This definition is not inherently restricted to planar CNNs. The indices $ij$ may be extended to multi-indices to include bipolar CNNs of higher order.

**Definition 3.2 (Slowest cells.)**

A slowest cell is a cell for which $T_{S_{ij}} = T_S$ holds. Let $\mathcal{I}$ be the set of all slowest cells, i.e., those cells that determine $T_S$,

$$\mathcal{I} = \{ c_{ij} : T_{S_{ij}} = T_S \}.$$

### 3.1.2 Significance of the Settling Time

$T_S$ depends on the input $u$, the initial state $x(0)$ and, in a complex and highly nonlinear manner, on the template set $\mathcal{I} = (A, B, I)$; this dependency is particularly interesting not only in most CNN research areas, but also when it comes to applications of a universally programmable CNN chip [10, 11, 42]. Typical issues related to settling time are:

- In a simulator, without a priori knowledge of $T_S$, it either has to be checked after each integration step whether the CNN has settled (which is not trivial), or the integration time $T_I$ has to be fixed beforehand to a "sufficiently large" value. In the latter case, $T_I$ is likely to be chosen much larger than necessary ($T_I \gg T_S$) to be on the safe side, which increases the computational effort drastically.

- In a program for the CNN universal chip [42], every processing step consists of either a logical operation or a CNN transient. Before starting the CNN transient, besides the templates and inputs, the processing time also has to be specified, since it is not possible to determine when the analog network has reached its equilibrium. Again, this run time can only be optimized if a tight upper bound $\hat{T}_S \gg T_S$ can be deduced from the template set.

- Having an estimate of $T_S$ at one’s disposal allows template optimization with respect to processing speed. Design rules for faster templates can be derived.

Although attempts have been made toward the design of templates with high processing speed [28], no rigorous analysis of the settling time has yet been reported. In this chapter, an exact approach for the classes of uncoupled and
3.2. The Exact Approach

non-propagating coupled templates (section 3.2) is presented, analytical esti­mates of $T_S$ for propagating-type templates (section 3.3) are derived, and several examples (section 3.4) are given. In section 3.5, the influence of parameter perturbations on the settling time is explored, and, since robustness aspects must not be neglected, the interdependence of robustness and speed is discussed in section 3.6.

3.2 The Exact Approach

In this section, for the sake of simplicity, input, state, and output of a particular cell are represented by one index, i.e., all cells are considered to form a vector.

3.2.1 Uncoupled Templates

Uncoupled templates have only one non-zero entry in the $A$-template, namely the center element $a_c$. Defining $w_i := B \ast u_i + I + \partial_i$, the CNN equation may be written as

$$\dot{x}_i(t) = -x_i(t) + a_c f(x_i(t)) + w_i, \quad 1 \leq i \leq n.$$  

(3.1)

In this case, all trajectories $x_i(t)$ increase or decrease monotonically, and $a_c > 1$ guarantees that the stable equilibria of all cells lie in the saturation region [3]. For the initial states, we assume $x(0) \in \mathbb{R}_0^n$.

Definition 3.3 (Initially linear cells.)
Let $\mathcal{C}$ be the set of cells that start within or enter the linear region at $t=0$,

$$\mathcal{C} = \left\{ C_i : (x_i(0) = 0 \cup x_i(0) \dot{x}_i(0) < 0) \right\}.$$  

Only cells in $\mathcal{C}$ influence the settling time. Their dynamics at the beginning of the trajectory is described by

$$\dot{x}_i(t) = (a_c - 1)x_i(t) + w_i, \quad C_i \in \mathcal{C}, \quad 0 \leq t \leq T_{\mathcal{C}_i}.$$  

(3.2)

Depending on the final output of a cell, we divide $\mathcal{C}$ into two subsets $\mathcal{C}^+$ ($y^* = +1$) and $\mathcal{C}^-$ ($y^* = -1$). In both subsets, we find the slowest cells
(\mathcal{C}_i^+ \text{ and } \mathcal{C}_i^-, \text{ respectively}) \text{ by evaluating}

\begin{align*}
\mathcal{C}^+ : \quad i^+ &= \arg \min_{i \in \mathcal{C}^+} \left\{ (a_c - 1) x_i(0) + w_i \right\} > 0 \\
\mathcal{C}^- : \quad i^- &= \arg \max_{i \in \mathcal{C}^-} \left\{ (a_c - 1) x_i(0) + w_i \right\} < 0.
\end{align*}

(3.3)

Solving the differential equation (3.2) yields

\begin{align*}
 x_{i^+}(t) &= \left( x(0) + \frac{w_{i^+}}{a_c - 1} \right) e^{(a_c - 1)t} - \frac{w_{i^+}}{a_c - 1}; \quad x(0) \in \{-1, 0\} \\
 x_{i^-}(t) &= \left( x(0) + \frac{w_{i^-}}{a_c - 1} \right) e^{(a_c - 1)t} - \frac{w_{i^-}}{a_c - 1}; \quad x(0) \in \{0, 1\}.
\end{align*}

(3.4) \quad (3.5)

The settling times are obtained by solving \( x_{i^+}(T_s^+) = +1 \) and \( x_{i^-}(T_s^-) = -1 \).

For \( a_c > 1 \), we get

\begin{align*}
\mathcal{C}^+ : \quad x_{i^+}(T_s^+) &= +1 \Rightarrow T_s^+ = \frac{1}{a_c - 1} \log \frac{1 + \frac{w_{i^+}}{a_c - 1}}{x_{i^+}(0) + \frac{w_{i^+}}{a_c - 1}} \\
\mathcal{C}^- : \quad x_{i^-}(T_s^-) &= -1 \Rightarrow T_s^- = \frac{-1 + \frac{w_{i^-}}{a_c - 1}}{a_c - 1} \log \frac{-1 + \frac{w_{i^-}}{a_c - 1}}{x_{i^-}(0) + \frac{w_{i^-}}{a_c - 1}},
\end{align*}

(3.6)

and, for \( a_c = 1 \),

\begin{align*}
T_s^+ &= \frac{1 - x_{i^+}(0)}{w_{i^+}}; \quad T_s^- = \frac{-1 - x_{i^-}(0)}{w_{i^-}}.
\end{align*}

(3.7)

Note that \( w_{i^+} > 0, w_{i^-} < 0, x_{i^+} \in \{-1, 0\}, \) and \( x_{i^-} \in \{0, 1\} \). The settling time can now be expressed as

\[ T_S = \max(T_s^+, T_s^-). \]

However, these analytical results depend on the input image and on the initial states; they do not allow the prediction of \( T_S \) if this information is not available. To provide an upper bound \( T_S^* \geq T_S \), we replace the expressions (3.3) by maximizations over all possible combinations of neighboring cells and initial conditions. In this way we find the worst possible cases for \( w \) and \( x(0) \), denoted by \( w_{wc} \) and \( x_{wc} \), respectively. With \( \xi(w, x) := (a_c - 1)x + w \), we
3.2. The Exact Approach

may write

\[(w_{wc}^+, x_{wc}^-) = \arg \min_{w = B \cdot 1 \pm 1,0 + I} \{ \xi(w, x) \mid \xi(w, x) > 0 \} \]

\[(w_{wc}^-, x_{wc}^+) = \arg \max_{w = B \cdot 1 \pm 1,0 + I} \{ \xi(w, x) \mid \xi(w, x) < 0 \}. \]  

(3.8)

\(1_{\pm 1,0}\) is any vector of length nine with elements \(\pm 1\) or zero, i.e., \(1_{\pm 1,0} \in \mathbb{R}_0^9\). (The zero entry is only necessary if the boundary condition of \(u\) is set to zero.) \(w_i\) and \(x_i\) in (3.6) can now be replaced by \(w_{wc}\) and \(x_{wc}\) to give the desired upper bound \(T_s\). The evaluation of (3.8) may seem tedious, as up to \(2 \cdot 3^9 = 39366\) combinations have to be checked, but in all practical examples, this number is drastically reduced due to symmetry, zero entries in the \(B\)-template, and/or spatially invariant initial conditions.

In template design problems, the parameter subspace within which the template performs the desired operation is determined by a set of inequalities [17,27]. Hence, in most cases, there are several degrees of freedom available for choosing the parameter values. Increasing the absolute value of some parameter(s) causes the settling time to tend towards zero, and the optimization problem is easily solved, at first sight. However, respecting implementation issues of the VLSI CNN chip, the realizable parameter space is bounded. Accordingly, the issue of speeding up the CNN processing has to be regarded under the additional condition \(\|\mathcal{T}\|_1 \leq \beta\).

**Example 3.1 (Logical NOT)**

The template set \(A = [a_c], B = [b_c], I = 0\) inverts the input image \((y^* = -u)\) if \(a_c > 1\) and \(b_c < 0\). For \(a_c = 1\) and \(x(0) = 0\), we get \(w_{wc}^+ = -w_{wc}^- = -b_c\) (3.8), and (3.7) yields \(T_s^* = -1/b_c\). Bipolar initialization doubles \(T_s^*\). For \(a_c > 1\) and zero initialization, (3.6) simplifies to

\[ T_s^* = \frac{1}{a_c - 1} \log \left(1 + \frac{a_c - 1}{-b_c} \right). \]  

(3.9)

Both increasing \(a_c\) and \(|b_c|\) results in faster convergence. However, if there is an upper bound for \(a_c + |b_c|\), it is better to focus on raising \(|b_c|\), since \(\left| \partial T_s^*(a_c, b_c) / \partial b_c \right| > \left| \partial T_s^*(a_c, b_c) / \partial a_c \right|\), i.e., the absolute sensitivity of the settling time with respect to \(b_c\) is larger (in absolute value) than that with respect to \(a_c\).
Example 3.2 (Edge extraction)
For any $q > 0$, the template set

$$a_c = 1 + q/4; \quad B = \begin{bmatrix} 0 & -q & 0 \\ -q & 4q & -q \\ 0 & -q & 0 \end{bmatrix}; \quad I = -q/2$$

extracts the edges from the input image [43]. Table 3.1 shows the settling time as a function of $q$, the boundary $\partial$, and the initial state. In all cases, it turns out that $T_S^* \sim 1/q$. For each case, the worst case constellation(s) of cells are listed as well. (* denotes a boundary cell with input zero.)

It follows that selecting $\partial = -1$ and $x(0) = -1$ is the best choice for this edge extractor.

Example 3.3 (Uncoupled horizontal line detection)
Due to the symmetry of the problem, the following template ansatz is appropriate:

$$A = [a_c]; \quad B = [s \quad b_c \quad s]; \quad I = z; \quad \partial_u = -1. \quad (3.10)$$

The desired behavior for different configurations of cells yields a set of inequalities which depends on the initial state $x(0)$.

Initialization with the input image$^1$ ($x(0) = u$).
Only isolated black cells (○●○) are slowest cells. Their dynamics in the linear

$^1$This implies bipolar initialization.
3.2. The Exact Approach

region is described by

\[ \dot{x} = x(a_c - 1) + b_c - 2s + z \]

\[ \Rightarrow x(t) = \left(1 - \frac{-b_c + 2s - z}{a_c - 1}\right)e^{(a_c - 1)t} + \frac{-b_c + 2s - z}{a_c - 1}. \]

(3.11) (3.12)

The settling time is defined by \( x(T^*_S) = -1 \), which evaluates to

\[ T^*_S = \frac{1}{a_c - 1} \log \frac{1 + \frac{-b_c + 2s - z}{a_c - 1}}{-1 + \frac{-b_c + 2s - z}{a_c - 1}}. \]

(3.13)

From example 2.1 we know that the template set

\[ A = [1 + q]; \quad B = [q \ q \ q]; \quad I = -q \quad (a_c = 1 + q, \ s = b_c = q, \ z = -q) \]

(3.14)

has a safety margin of \( q \) in all its local rules. Since only an isolated and initially black cell toggles (\( \bullet \circ \circ \Rightarrow \circ \circ \circ \)), \( \mathcal{C}^+ \) is empty and \( w_{wc} = -q \), resulting in

\[ T^*_s = T^*_S = \frac{1}{q} \log 3 \approx \frac{1.10}{q}. \]

(3.15)

The fastest template for uncoupled HLD with input initialization is, however,

\[ A = [1 + 2q]; \quad B = [q \ 0 \ q]; \quad I = -q \quad (a_c = 1 + 2q, \ s = q, \ b_c = 0, \ z = -q) \]

(3.16)

with the same safety margin and

\[ T^*_s = T^*_S = \frac{1}{2q} \log 5 \approx \frac{0.80}{q}. \]

(3.17)

Initialization with \( x(0) = 0 \). At time 0, \( a_c \) has no influence on \( \dot{x} \). The system

\[ \circ \bullet \circ : \quad 0 > -2s + b_c + z \]

\[ \circ \bullet \bullet : \quad 0 \leq b_c + z \]

\[ \bullet \circ \bullet : \quad 0 \geq -b_c + 2s \]

(3.18)

is solved by

\[ A = [a_c]; \quad B = [q \ 3q \ q]; \quad I = -2q. \]
Once again, the margin is \( q \cdot a_c \geq 1 \) is arbitrary. For the configurations \( \circ \bullet \circ \) and \( \circ \bullet \bullet \), we obtain \( w_{wc}^- = -q \) and \( u_{wc}^+ = q \), respectively. In both cases, the cell settles at

\[
T^*_S = \begin{cases} 
\frac{1}{a_c-1} \log \left( 1 + \frac{a_c-1}{q} \right) & \text{for } a_c > 1 \\
\frac{1}{q} & \text{for } a_c = 1.
\end{cases}
\] (3.19)

Clearly, by increasing \( a_c \) or \( q \), we achieve higher processing speed. Given the upper bound \( \beta \), we replace \( q \) by \((\beta - a_c)/7 \) in (3.19) and solve \( \partial T^*_S(a_c, \beta)/\partial a_c = 0 \), which yields optimum values \( a_c^\beta(\beta) \) and \( q^\beta(\beta) \). However, \( T^*_S \) turns out to be a few percent larger than in the case with input initialization. As a consequence, we recognize that it is good advice to use input initialization in the case of uncoupled HLD.

### 3.2.2 Coupled Templates without Propagation

As an example for this small template class, we examine the coupled horizontal fine detector,

\[
A = [s \ p \ s], \quad B = [0], \quad I = z, \quad \partial_x = -1.
\] (3.20)

Its parameter region in the \((s, p, z)\)-space is bounded by three planes:

\[
-1 + p - 2s + z < 0 \quad (3.21)
\]
\[
-1 + p + z \geq 0 \quad (3.22)
\]
\[
1 - p + 2s + z \leq 0. \quad (3.23)
\]

Only black cells with two white neighbors (\( \circ \bullet \circ \)) enter the linear region, where their state equation is

\[
\dot{x}(t) = x(t)(p - 1) - 2s + z \quad (3.24)
\]
\[
\Rightarrow \quad x(t) = \left( 1 - \frac{2s - z}{p - 1} \right) e^{(p-1)t} + \frac{2s - z}{p - 1}. \quad (3.25)
\]

The cell is again considered to be settled when the trajectory (3.25) leaves the linear region, i.e., \( x(T^*_S) = -1 \). We thus obtain

\[
T^*_S = \frac{1}{p - 1} \log \frac{1 + \frac{2s - z}{p - 1}}{-1 + \frac{2s - z}{p - 1}}. \quad (3.26)
\]
If there is at least one isolated black cell in the initial state, then \( T_S = T_S^* \), otherwise, \( T_S = 0 \).

Compared to the settling time of the *uncoupled* horizontal line detector with input initialization (3.13), the results differ only in the presence of the \( b_c \)-parameter in the uncoupled case. Note that in the coupled case, the center element of the \( A \)-template is denoted by \( p \) and not by \( a_c \).

### 3.3 Analysis of Propagation-type Templates

#### 3.3.1 Introduction

The basic idea behind the analysis of this class of templates is to estimate the time \( \Delta t \) that it takes the operation to progress from one cell to its neighbor(s). \( \Delta t \), multiplied by the maximum possible number of cells in a “propagation string” upperbounds the settling time. We start with a rigorous definition of the term “propagation string” and of \( \Delta t \), restricting ourselves to bipolar initialization, i.e., \( x(0) \in \mathbb{B}^n \).

#### 3.3.2 Propagation Strings

**Definition 3.4 (Affected cells.)**

A cell \( C_i \) is said to flip if \( \exists t > 0 \quad y_i(t) = -y_i(0) \).

A cell \( C_i \) is affected by a cell \( C_j \) if it flips and

\[
\int_{t=0}^{\infty} \left( y_i(t) \bigg|_{y_i(0)=1} - y_i(t) \bigg|_{y_i(0)=-1} \right)^2 \, dt > 0 \tag{3.27}
\]

holds. (Note that (3.27) may be an improper integral.)

In words, \( C_i \) is affected by \( C_j \) if it flips and depends on the initial state of \( C_j \). Any cell is by definition affected by itself.

**Definition 3.5 (Propagation strings.)**

A propagation string of length \( L \) is an ordered set of cells

\[
\langle C_{k(1)}, C_{k(2)}, \ldots, C_{k(L)} \rangle
\]

for which \( C_{k(i)} \) is affected by \( C_{k(j)} \) \( \forall 1 \leq j < i \leq L \).

A maximum propagation string is a propagation string for which \( \not\exists C_q \quad (1 \leq q < L) \)
such that \( \langle C_q, C_{k(1)}, C_{k(2)}, \ldots, C_{k(L)} \rangle \) or \( \langle C_{k(1)}, C_{k(2)}, \ldots, C_{k(L)}, C_q \rangle \) is a propagation string.

Hence, no cell can be added to a maximum propagation string either at the beginning nor at the end to build another propagation string.

A propagation string may be considered to be a "fuse" that is "lit" at the cell \( C_{k(1)} \) and then "burns" through all cells that are influenced by it, up to \( C_{k(L)} \).

Propagation can be likened to a type of domino effect, where the cells constituting a propagation string play the role of the dominoes.

**Theorem 3.1 (Properties of propagation strings.)**

Let \( \mathcal{C} \) and \( \mathcal{S} \) be the sets of initially linear and slowest cells, respectively (cf. Def. 3.3 and Def. 3.2). A propagation string contains at most one cell which belongs to \( \mathcal{C} \) and at most one cell which belongs to \( \mathcal{S} \). A maximum propagation string containing both \( (C_{k(c)} \in \mathcal{C} \) and \( C_{k(s)} \in \mathcal{S} \)) for some \( c \) and \( s \), is called a longest propagation string. \( c \in \{1, 2\} \) and \( s \in \{L - 1, L\} \), depending on the template set.

Proof: A propagation string is an ordered set. If two cells were initially linear or there were two slowest cells, this would contradict the property of being ordered. The property of being affected implies that no pair of cells must flip at the same time.

Since the (maximum) neighborhood radius of the templates under consideration is 1, \( c \) cannot be larger than 2 and \( s \) cannot be smaller than \( L - 1 \). In fact, for most templates, \( c = 1 \) and \( s = L \), except for CCD, where \( c = 2 \) and \( s = L - 1 \), i.e., the second cell in the string is affected by the first one, but is itself the first to become linear, and the penultimate cell is a slowest cell (which determines \( T_S \)), but affects the last one.

**Corollary 3.1 (Property of affected cells.)**

The states \( x(t) \) of all affected cells which do not belong to \( \mathcal{C} \) exhibit non-monotonic behavior if \( x(0) \neq 0 \).

Proof: Being affected implies flipping. If an affected cell is not an element of \( \mathcal{C} \), if will enter the linear region at some time \( \tau > 0 \). Hence, its state's derivative at \( t = 0 \) is either zero or causes the trajectory to withdraw from the linear region, resulting in non-monotonic behavior.

**Corollary 3.2 (Property of longest propagation strings.)**

Assuming isotropic propagation, all longest propagation strings have the same length \( L_p \), and there are no propagation strings which are longer than \( L_p \).
3.3. Analysis of Propagation-type Templates

Proof: No cell can start flipping earlier than at time zero, and no cell’s output can per definition settle later than a slowest cell. On the other hand, all longest propagation strings contain an initially linear and a slowest cell, so they must have equal length if the propagation speed is the same, which is guaranteed by the assumption.

Definition 3.6 (Input affected cells.)
Equivalently to Def. 3.4, we define the notion of being affected by the input of a cell:
A cell $C_i$ is affected by the input of a cell $C_j$ if it flips and

$$\int_{t=0}^{\infty} \left( y_i(t)\big|_{u_j=1} - y_i(t)\big|_{u_j=-1} \right)^2 dt > 0$$

holds. By straightforward extension, we define input-driven (maximum) propagation strings.

$L_p$ depends on the input image and/or the initial state. The maximization over all possible inputs and initial states yields an upper bound $L_p^*$ on $L_p$,

$$L_p^* = \max_{u,x(0)\in B^n} (L_p)$$

Example 3.4 (Connected component detection)
The detector is assumed to operate row-wise from left to right. The initial state representing the worst case in terms of settling time is any row containing a single black pixel in its leftmost column. This pixel then has to travel through the whole row.

```
1 2 3 4 5 6 N
```

Every such row forms a longest propagation string. The second-to-last cell, i.e., the one to the left of the $N^{th}$ cell in this string, is a slowest cell, and the right neighbor of the black cell is an element of $\mathcal{C}$. We find $L_p^* = N$.

Example 3.5 (Shadowing)
This is a row-wise operation, with a light source on the right. A single black cell in the rightmost column projects its shadow through the whole row. We find $L_p^* = N$, as in the previous example.
Example 3.6 (Global connectivity detection)
At first sight, two different schemes, a spiral-type scheme (Fig. 3.1(a)) and a meander-type scheme (Fig. 3.1(b)) seem to be good candidates for longest GCD propagation strings. $L^*_p$ depends not only on the dimensions $M$ and $N$, but also on whether they are even or odd. If only $M$ is even, both schemes result in the same length for $M = N - 1$, and the meander is longer if $M < N - 1$.

If both $M$ and $N$ are even, neither meanders nor spirals appear optimum. In fact, more elaborate schemes result in longer strings, e.g., Fig. 3.1(c) and (d). For any even $M$ and $N$, there is a string of length $\frac{1}{2}(MN+M+N-2)$.

We finally obtain

$$L^*_p(M, N) = \begin{cases} 
\frac{MN+M+N-2}{2} & \text{if } M \text{ and } N \text{ even} \\
\frac{MN+M+N-1}{2} & \text{otherwise.}
\end{cases} \tag{3.30}$$

Example 3.7 (Hole filling)
Although this task differs considerably from GCD, its $L^*_p$ is closely related to (3.30). The worst case is an input image with an almost completely closed margin of black cells (Fig. 3.1(e)). Only one cell of the margin is white; this cell is $C_{k(1)}$ of the longest propagation string. Inside, we have to solve exactly the same problem as in the case of GCD, except that we look for a string of white cells. Due to the margin, the dimension reduces to $(M - 2) \times (N - 2)$.

Using (3.30) and adding the white margin cell, we end up with

$$L^*_p(M, N) = \begin{cases} 
\frac{MN-M-N}{2} & \text{if } M \text{ and } N \text{ even} \\
\frac{MN-M-N+1}{2} & \text{otherwise.}
\end{cases} \tag{3.31}$$

For a $10 \times 10$ image, $L^*_p = 40$ is the worst case for a hole filler, for example (see Fig. 3.1(e)).
3.3.3 The One-cell Propagation Time

Definition 3.7 (The one-cell propagation time.)
Let $\xi_i$ be the time of the first zero-crossing of $y_i(t)$,

$$\xi_i = \min \{ t \mid y_i(t) = 0 \}.$$

The one-cell propagation time $\Delta t$ is defined as the average time between the first zero-crossings of two neighbor cells in a propagation string:

$$\Delta t = \frac{1}{L-2} \sum_{i=2}^{L-1} \left( \xi_{k(i+1)} - \xi_{k(i)} \right).$$

(The summation starts from 2 since the first cell $C_{k(1)}$ may not have a zero-crossing.)

3.3.4 Estimation of the Settling Time

$L_p$ and $\Delta t$ now enable us to give an estimate of the settling time.

$$T_S \approx L_p \cdot \Delta t. \quad (3.32)$$

An upper bound $L_p^*$ is found by maximization of (3.29). It remains to give an estimate $\hat{T}_S \geq \Delta t$ in order to get the desired $\hat{T}_S = L_p^* \cdot \hat{\Delta} t$. Note that $\hat{T}_S$ denotes an approximate upper bound for the settling time, whereas $T_s^*$ in section 3.2 denotes the exact settling time of a CNN containing the worst case configuration of cells for a particular task. In the next section, we investigate several examples.

3.4 Examples of Propagation-type Templates

3.4.1 Shadowing

The shadowing operation is defined by the template set

$$A = [0 \, p \, s]; \quad B = [0]; \quad I = s, \quad \text{with } 1 < p < 2s + 1$$
with a boundary of $-1$.

To estimate $\Delta t$, we investigate the dynamics of a CNN of dimension $1 \times N$. All cells are assumed to be initially white, except the rightmost cell $C_N$, which is the first cell in the longest propagation string $\langle C_{N}, C_{N-1}, \ldots, C_1 \rangle$:

$$
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & \ldots & N
\end{array}
$$

We focus on a cell $C_i$ that is “sufficiently far away” from $C_N$ (Fig. 3.2, where $i = 6$). Until its right neighbor becomes linear, the cell’s dynamics is governed by

$$
\dot{x}_i = -x_i - p \\
\implies x_i(t) = (-1 + p)e^{-t} - p.
$$

(3.33)  
(3.34)

To estimate the time $\Delta t = \zeta_{i+1} - \zeta_i$ (from the zero-crossing of $C_{i+1}$ to the zero-crossing of $C_i$), we assume that $x_i(\zeta_{i+1}) \approx -p$. Solving the new differential equation

$$
\dot{x}_i = -x_i - p + 2s
$$

yields

$$
x_i(t) = -2s e^{-t} - p + 2s \quad \text{for} \quad \zeta_{i+1} \leq t \leq \tau_1, \quad x_i(\zeta_{i+1}) \approx -p.
$$

(3.35)

where $\tau_1$ denotes the time when $C_i$ itself becomes linear, i.e., $x(\tau_1) = -1$. For $C_i$ linear, $\dot{x}_i = x_i(p - 1) + 2s$ holds. The trajectory is therefore

$$
x_i(t) = \left(-1 + \frac{2s}{p - 1}\right)e^{(p - 1)t} - \frac{2s}{p - 1} \\
= (-1 + \mu)e^{(p - 1)t} - \mu, \quad x_i(\tau_1) = -1
$$

(3.36)  
(3.37)

with $\mu = 2s/(p - 1)$, as in the previous section.

Since $\Delta t = \zeta_i - \zeta_{i+1}$, it remains to add $\tau_1 - \zeta_{i+1}$ and $\zeta_i - \tau_1$:

$$
\tau_1 - \zeta_{i+1} \approx -\log\left(1 - \frac{1}{\mu}\right) \quad \text{(from (3.35))}
$$

(3.38)

$$
\zeta_i - \tau_1 \approx \frac{1}{p - 1}\log\left(\frac{\mu}{\mu - 1}\right) \quad \text{(from (3.36))}
$$

(3.39)

$$
\implies \tilde{\Delta t} = \frac{p}{p - 1}\log\left(\frac{\mu}{\mu - 1}\right) = \frac{p}{p - 1}\log\left(\frac{2s}{2s - p + 1}\right).
$$

(3.40)

\footnote{This equation holds from the time when $C_{i+1}$ becomes linear. Here, we approximate this time by $\zeta_{i+1}$.}
Shadowing (SH) is closely related to CCD (slow case), as a comparison of this result with (3.62) below exhibits (Fig. 3.3).

\[
\hat{T}_{S_{SH}} = \frac{p}{p-1} (\hat{T}_{S_{CCD}} - \log 2)
\]  

(3.41)

The maximum error is 34\% (for \( p/s \ll 1 \)), the mean error is 11\% (Fig. 3.4), whereby large relative errors occur for fast templates. However, the absolute error (Fig. 3.5) is of practical interest, since this quantity equals the effective loss in processing time when relying on the derived upper bound \( \hat{T}_S \).

### 3.4.2 Global Connectivity Detection

To benefit from the results of the shadowing task, we use

\[
A = \begin{bmatrix} 0 & s & 0 \\ s & p & s \\ 0 & s & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & -s & 0 \\ -s & 0 & -s \\ 0 & -s & 0 \end{bmatrix}, \quad I = 0
\]  

(3.42)

with \( 1 < p < 2s + 1 \) as a parameterization of the GCD-template. Since \( I = 0 \), the propagation is symmetric in the sense that it affects white and black cells in the same way, i.e., it detects connected black cells on a white background as well as connected white cells on a black background. For our analysis, we restrict ourselves to the former case, without loss of generality, and examine a \( 1 \times N \) CNN with input \( \cdots \cdots \cdots \cdots \cdots \cdot \) and initial state \( \square \cdots \cdots \cdots \cdots \cdots \square \) (Fig. 3.6).

A black cell \( C_i \) whose neighboring cell’s inputs and outputs are equal tends exponentially towards \( p \) (from \( \dot{x_i} = -x_i + p \)). If one neighbor turns from black to white, the transient starts decreasing (\( \dot{x_i} = -x_i + p - 2s \)). Hence it exhibits exactly the same behavior as for shadowing, and the upper bound for the settling time is, using (3.30),

\[
\hat{T}_S = L_p \Delta t = \left\{ \begin{array}{ll}
\frac{MN+M+N-2}{2} & \frac{p}{p-1} - \log\left(\frac{2s}{2s-p+1}\right) \\
\frac{MN+M+N-1}{2} & \frac{p}{p-1} - \log\left(\frac{2s}{2s-p+1}\right)
\end{array} \right.
\]

(3.43)

\( ^3 \)If more than one neighbor turns white, the propagation is faster.
Figure 3.2: Transients for shadowing. $A = [0.2, 9, 1]$, $I = 1$.

Figure 3.3: Settling time $\hat{T}_S$ and upper bound $T_S$ for shadowing. ($N = 60$ cells.)
3.4. Examples of Propagation-type Templates

**Figure 3.4:** Relative error between $\hat{T}_S$ and $T_S$ for shadowing.

**Figure 3.5:** Absolute error between $\hat{T}_S$ and $T_S$ for shadowing.
Figure 3.6: Transients for GCD. $s = 1, p = 2.9$. 
For the robust template with $p = 3$ and $s = 2$ and an image size of $100 \times 100$, for example, the longest propagation string contains 5099 cells, and we obtain $t_5' \approx 5300$, which is remarkably long.

So far the center parameter of the $B$-template, $b_c$, and the bias $I$ were set to zero. Changing those parameters may speed up the processing, while preserving the GCD operation. Since for $I \neq 0$, we would have to give up the symmetric propagation property, we keep a zero bias and concentrate on $b_c$. If the detector is to operate on black connected components only, altering $I$ and $b_c$ would have exactly the same effect.

First, the bounds for $b_c$ within which correct operation is guaranteed must be established. We consider an initially black cell $C_i$ which belongs to the “tail” of a maximum propagation string. Until one of its neighbors enters the linear region, its transient obeys $\dot{x}_i = -x_i + p + b_c$, and the cell has to remain black, i.e., $x_i \geq 0$, leading to $b_c \geq 1 - p$. On the other hand, an initially white cell $C_j$ must remain white, even if it has a black input, which implies $b_c < p - 1$ (from $\dot{x}_j = -x_j - p + b_c > 0$). Furthermore, $C_i$ has to flip when one of its neighbors flips: $\dot{x}_i = -x_i - p + b_c > 0 \Rightarrow b_c < 2s - p + 1$. These three conditions may be summarized by

$$1 - p \leq b_c < \min(p - 1, 2s - p + 1).$$  \hspace{1cm} (3.44)

One intuitively expects that decreasing $b_c$ causes faster propagation, since it affects $\dot{x}$ “in the correct direction”, and since the intermediate equilibrium state $x_i = p + b_c$ of $C_i$ is closer to +1 for $b_c < 0$; consequently, the time to reach the linear region is reduced. Indeed, the settling time can be considerably shortened by choosing a negative $b_c$, as the following analysis shows.

As with the shadowing example, we divide the time from $\zeta_i$ to $\zeta_{i+1}$ into two phases, one from $\zeta_i$ to $\tau_1$ (the time when $x_i = 1$), and another from $\tau_1$ to $\zeta_{i+1}$. Solving the respective differential equations yields

$$\tau_1 - \zeta_i \approx \log \left( \frac{2s}{2s - p - b_c + 1} \right)$$ \hspace{1cm} (3.45)

$$\zeta_{i+1} - \tau_1 \approx \frac{1}{p - 1} \log \left( \frac{2s - b_c}{2s - p - b_c + 1} \right)$$ \hspace{1cm} (3.46)

$$\Rightarrow \ \hat{\Delta t} = \log 2s + \frac{1}{p - 1} \log \left( \frac{2s - b_c}{(2s - p - b_c + 1)^p} \right).$$ \hspace{1cm} (3.47)

This result has been verified for negative $b_c$; the relative difference between this upper bound and the effective settling time ranges from 5% to 17% with
Figure 3.7: GCD. Settling time as a function of $b_c$, $p$ and $s$ for $L_P = 60$.

an average of 11% for $p=\text{const.}$ (Fig. 3.7 (a)), and from 4% to 12% with an average of 8% for $s=\text{const.}$ (Fig. 3.7 (b)). The gain in processing speed is remarkable: setting $b_c = -2.75$ instead of $b_c = 0$ in Fig. 3.7 (a) saves 84% of processing time (averaged over all $s$). For the above example ($100 \times 100$ cells, same $s$ and $p$), introducing a $b_c = -1.5$ would reduce the settling time to a value below 1833. However, decreasing $b_c$ towards the lower limit $1 - p$ entails a loss in robustness — there is a trade-off between robustness and processing speed (see section 3.6).
3.4.3 Connected Component Detection

We analyze the CCD template

\[ A = [s \ p \ -s]; \quad B = [0]; \quad I = 0; \quad \partial_x = -1 \quad \text{with} \quad 1 < p < 2s + 1. \]

The corresponding CNN equation is given as

\[ \dot{x}_i = -x_i + p \ \text{sat}(x_i) + s \ \text{sat}(x_{i-1}) - s \ \text{sat}(x_{i+1}). \quad (3.48) \]

From example 3.4, we know that in this case \( L_p^* = N \). The smallest input image which contains this longest propagation string has the dimension \( 1 \times N \), with an initially black cell \( C_1 \):

\[
\begin{array}{cccccccc}
\blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & N
\end{array}
\]

To determine \( \Delta t \), the dynamics of this CNN have to be investigated. We divide the problem along the time axis into different phases within which no transitions to or from the saturation region occur. Hence, within one region, the system of differential equations is linear, but inhomogeneous. In the first phase, independent of \( p \) and \( s \), only \( C_2 \) enters the linear region.

\[
\begin{align*}
\dot{x}_1 &= -x_1 + p - s(1 + x_2), \quad x_1(0) = 1 + 1 \\
\dot{x}_2 &= (p - 1)x_2 + 2s, \quad x_2(0) = -1 \\
\dot{x}_3 &= -x_2 - p + s(1 + x_2), \quad x_3(0) = -1 \\
\dot{x}_i &= -x_i - p, \quad x_i(0) = -1 \quad 4 \leq i \leq N. 
\end{align*}
\]

\( x_2(t) \) is independent of its neighbors and evaluates to

\[ x_2(t) = (-1 + \mu)e^{(p-1)t} - \mu \quad \text{with} \quad \mu := \frac{2s}{p - 1}. \quad (3.53) \]

Inserting (3.53) into (3.49) yields

\[
\begin{align*}
\dot{x}_1 &= -x_1 + p + s(\mu - 1)(1 - e^{(p-1)t}) \\
\Rightarrow \quad x_1(t) &= \alpha(s, p)e^{-t} + \beta(s, p)e^{(p-1)t} + \gamma(s, p), \quad (3.55)
\end{align*}
\]

where \( \alpha(\cdot, \cdot) \), \( \beta(\cdot, \cdot) \), and \( \gamma(\cdot, \cdot) \) are rational functions of \( s \) and \( p \). (3.49) to (3.52) remain valid until either \( C_2 \) enters the positive saturation region or \( C_1 \) (and \( C_3 \)) enters the linear range, whichever occurs first. Fig. 3.8 shows both cases. In (a) \( s = p = 3 \), \( C_1 \) is “faster”, whereas in (b) \( s = 3 \), \( p = 6 \), it is “slower”. The time when \( x_2(t) = 1 \) is denoted by \( T_2 \); from (3.53), we get
Chapter 3. CNN Settling Time

Figure 3.8: CCD, beginning of the transients of cells 1 to 4.

\[ T_2 = \frac{1}{p - 1} \log \frac{\mu + 1}{\mu - 1}. \]  

(3.56)

However, \( x_1(T_1) = 1 \) cannot be solved for \( T_1 \) analytically (apart from the trivial solution \( T_1 = 0 \)) due to the superposition of two exponential functions in (3.55). A third-order Taylor series expansion at \( t = 0 \) yields an approximate analytic solution for \( T_1 \).

The state trajectories demonstrate fundamentally different behavior in the cases \( T_1 \leq T_2 \) and \( T_1 > T_2 \) since the number of linear cells is different. Approximately, the following regions of the parameter space \((s, p)\) lead to \( T_1 \leq T_2 \), which we will denote as case \( F \), or to \( T_1 > T_2 \) (case \( S \)), respectively:

\[
\begin{align*}
\text{case F:} & \quad T_1 \leq T_2 \iff p \leq 2s - 1 \\
\text{case S:} & \quad T_1 > T_2 \iff p > 2s - 1
\end{align*}
\]  

(3.57)

Note that in case \( F \), \( T_2 \) is not exact, since it is based on an equation which is not valid for \( t > T_1 \), while in case \( S \), the same argument applies to \( T_1 \). Both cases have to be studied separately. In this paper, we do not consider degenerate cases where \( p \) is considerably greater than \( s \); we restrict ourselves to \( p/s < 9/4 \), which includes all commonly used CCD templates.
In Fig. 3.9, typical examples of a CNN with eight cells for both cases, S and F, are presented. The time $\Delta t$ between the maxima of the states of two neighboring cells, multiplied by the number of cells $N$, determines the settling time: $T_s \approx N \Delta t$. The maxima coincide with the intersection of the trajectories of the neighbors, which is due to the antisymmetry of the $A$-template. Case F (Fig. 3.9a) is characterized by the fact that there is al-

Figure 3.9: CCD transients. The vertical lines indicate $T_s$.

most always only one cell $C_i$ in the positive saturation region. Its neighbors intersect at $x_{i-1}(t) = x_{i+1}(t) \approx 0$; this symmetry accounts for the fact that $\max(x_i(t)) \approx p$. Hence, $\Delta t$ equals the time it takes $x_i(t)$ to decrease from its approximate maximum $p$ to zero. The dynamics of $x_i(t)$ in this phase is governed by two differential equations,

$$
\dot{\xi}_i = -\xi_i + p + s(\xi_{i-1} - \xi_{i+1}) \quad |\xi_i| > 1, \quad (3.58)
$$
$$
\dot{\xi}_i = \xi_i(p - 1) + 2s \quad |\xi_i| \leq 1. \quad (3.59)
$$

Within the time span $\Delta t$, the trajectory starts from $x_i(t) \approx p$ and tends exponentially towards zero. Its behavior can be modeled by an exponential function $a e^{bt} + p - a$ with $1 - p < b < -1$ and $2s < a < 2s + p$, where the upper bounds originate from (3.58) and the lower bounds from (3.59). To determine a tight upper bound $\widetilde{\Delta t}$ for $\Delta t$, we choose $b := -1$ and $a := 2s + p$ and
solve $2s e^{-\Delta t} + p - 2s = 0$:

$$\Delta t = -\log\left(1 - \frac{p}{2s}\right) \geq \Delta t.$$

(3.60)

By multiplying (3.60) by the CNN length $N$, a tight upper bound for $T_S$ for case F can be deduced:

$$\hat{T}_S^F = -N \log\left(1 - \frac{p}{2s}\right).$$

(3.61)

In case S, $\Delta t$ is equal to the transition time of a cell $C_i$ from its maximum to the edge of the linear region, $x_i(t) = 1$ (Fig. 3.9b). Since the neighboring cells are saturated, $x_i = -x_i + p - 2s$ holds. To be on the safe side, we assume the maximum of $x_i(t)$ to be $p + 2s$; the upper bound for $T_S$ can now be readily obtained by solving $4s e^{-t} + p - 2s = 1$, which results in

$$\hat{T}_S^S = -N \log\left\{\frac{2s - p + 1}{4s}\right\} = -N \log\left(\frac{1}{2} - \frac{1}{2s}\right).$$

(3.62)

Both estimates $\hat{T}_S^F$ and $\hat{T}_S^S$ are always greater than the effective settling time. Therefore, their minimum can be taken as the tighter of the two bounds (note that $\hat{T}_S^F$ is not defined for $p \geq 2s$): $\hat{T}_S = \min(\hat{T}_S^F, \hat{T}_S^S)$. For $p = 2s - 1$, both estimates are equal, in agreement with (3.57). Templates of type F are fast, i.e., the CNN converges rapidly, whereas templates of type S are slow, hence the choice of letters.

These analytical results have been verified by highly accurate numerical simulations of a CNN with $N = 40$ cells, $p \in \{1.5, 2.5, 3.5 \ldots, 40.5\}$, and $s \in \{1, 1.5, 2, \ldots, 20\}$. Fig. 3.10 (a) shows the effective settling time $T_S$, while in (b), the upper bound $\hat{T}_S$ is presented. Both graphs are characterized by the fact that $T_S(s, p) \sim \frac{1}{s}$ and $T_S(s, p) \sim e^{vp}$ for a constant $v$. The relative difference $(\hat{T}_S - T_S)/T_S$ (Fig. 3.11) ranges between 4% and 42%, with an average of 12%; only for very fast ($p/s \ll 1$) and very slow templates ($p/s \approx 2$), this difference is greater than 15%.

### 3.5 Sensitivity to Parameter Deviations

#### 3.5.1 The Perturbed Settling Time

For simulations, the formulas presented in the previous sections apply in a straightforward manner; in a CNN program for a VLSI chip, however, where
3.5. Sensitivity to Parameter Deviations

Figure 3.10: Settling time $T_S$ and upper bound $\hat{T}_S$ for CCD.

Figure 3.11: Relative error between $\hat{T}_S$ and $T_S$ for CCD.
the run time of every program step has to be specified, the problem of parameter deviations related to the analog implementation has to be taken into account.

**Definition 3.8 (Perturbed settling time.)**

We define the perturbed settling time $\hat{T}(\alpha)$ to be the settling time under the condition that all template entries suffer from a relative perturbation of $\alpha$ in the direction in which the settling time is prolonged.

The relative sensitivity of the settling time to a single template entry $z$ is given by

$$S_z^{TS} = \frac{\partial T_S(z)}{\partial z} \cdot \frac{z}{T_S(z)},$$

(3.63)

The variation $V_z^{TS}$ can then be expressed as

$$V_z^{TS} = \frac{\Delta T_S}{T_S} = S_z^{TS} \cdot \Delta z / z,$$

(3.64)

Let $\alpha$ be the maximum relative deviation of all template entries. The resulting total variation is, in the worst case,

$$\frac{\Delta T_S}{T_S} = \alpha \sum_{i=1}^m |S_{z_i}^{TS}|,$$

(3.65)

where $m$ denotes the number of non-zero entries $z_i$ in the template set. With $\hat{T}_S(z_1, \ldots, z_m)$ as an upper bound for $T_S$ we derive an upper bound $\hat{T}_S$ for the perturbed settling time:

$$\hat{T}_S(\alpha) = \hat{T}_S \left(1 + \alpha \sum_{i=1}^m |S_{z_i}^{\hat{T}_S}| \right) = \hat{T}_S + \alpha \sum_{i=1}^m z_i \frac{\partial \hat{T}_S}{\partial z_i}.$$

(3.66)

Apart from providing information about the perturbed settling time, the sensitivity analysis is useful to optimize templates. If there is a constraint $\| \mathcal{T} \|_1 \leq \beta$, then the parameter $z_i$ with the highest absolute sensitivity $|\partial T_S / \partial z_i|$ should be increased to achieve the maximum gain in processing speed (see example 3.1 and subsection 3.6.3).
Example 3.8 (Connected component detection)
From (3.61) and (3.62) we have
\[ \hat{T}_S^F = -N \log \left( 1 - \frac{p}{2s} \right) \quad \text{and} \quad \hat{T}_S^S = -N \log \left( \frac{2s - p + 1}{4s} \right). \]

Focusing on the slow case \( (S) \), we get
\[ \frac{\partial \hat{T}_S^S}{\partial p} = \frac{1}{2s - p + 1}, \quad \frac{\partial \hat{T}_S^S}{\partial s} = -\frac{p - 1}{s(2s - p + 1)}. \]

For the commonly used CCD template \( s = 1 \) and \( p = 2 \), these absolute sensitivities evaluate to \( 1 \) and \( -1 \), respectively, leading to
\[ S_p \hat{T}_S^S = \frac{2}{\log 4} \approx 1.44 \quad \text{and} \quad S_s \hat{T}_S^S = -\frac{1}{\log 4} \approx -0.72. \]

This analysis is based on the fact that both off-center parameters in the \( A \)-template affect \( T_S \) equally. However, this is not the case, as the simulation in Fig. 3.12 demonstrates. For \( A = [s_1 \ p \ s_2] \) with \( s_1 = 1, \ p = 2 \) and \( s_2 = -1 \), the sensitivities differ substantially (\( s_1 : -0.37, \ s_2 : -1.04 \)). When deriving the formula (3.62), we assumed that \( s_1 = -s_2 \), which may not hold, of course, on an analog chip. To reveal the sensitivity information individually for both parameters, we would have to re-analyze the dynamics, allowing different \( s_1 \) and \( s_2 \).

Fortunately, this is not necessary to obtain a correct upper bound for the perturbed settling time, since our analysis for the sensitivity to a single \( s \) yields (as intuition suggests) rather precisely the average of the sensitivities to \( s_1 \) and \( s_2 \) (which might not be true generally).

The relative sensitivities sum up to \( \frac{3}{\log 4} \approx 2.16 \) and, for the absolute settling time, from
\[ \hat{T}_S^F = \hat{T}_S^F + \alpha N \frac{2p}{2s - p}, \quad \text{and} \quad \hat{T}_S^S = \hat{T}_S^S + \alpha N \frac{p - 1}{2s - p + 1}, \quad (3.67) \]
we have \( \hat{T}_S = N(\log 4 + 3\alpha) \). For \( \alpha = 8\% \), the settling time is prolonged by 17%.

Example 3.9 (Global connectivity detection)
GCD can be parameterized by
\[ A = \begin{bmatrix} 0 & s & 0 \\ s & p & s \\ 0 & s & 0 \end{bmatrix}; \quad B = \begin{bmatrix} 0 & -s & 0 \\ -s & b_c & -s \\ 0 & -s & 0 \end{bmatrix}; \quad I = 0. \quad (3.68) \]
As in the case of CCD, all the parameters with nominal value ±s may be perturbed independently, but from (3.47), only the sensitivity with respect to one common s-parameter, \( S_s^{\hat{T}_S} \), can be deduced. Simulations show that also for GCD, the sum of all 8 individual sensitivities are approximately equal to 8 \( S_s^{\hat{T}_S} \). For \( p \) and \( b_c \), the plots in Fig. 3.13 exhibit good correspondence between the simulation results and the linear sensitivity analysis. The total variation of \( T_S \) is upperbounded by

\[
\alpha \left( |S_p^{\hat{T}_S}| + |S_{b_c}^{\hat{T}_S}| + 8 |S_s^{\hat{T}_S}| \right).
\]  

Table 3.2: Relative sensitivities for GCD. \( p = 3, b_c = -1, s = 2 \).

<table>
<thead>
<tr>
<th>Sensitivity</th>
<th>( z := p )</th>
<th>( z := s )</th>
<th>( z := b_c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_z^{\hat{T}_S} ) (simulation)</td>
<td>1.651</td>
<td>-1.112</td>
<td>-0.748</td>
</tr>
<tr>
<td>( S_z^{\hat{T}_S} ) (analysis)</td>
<td>1.664</td>
<td>-1.105</td>
<td>-0.737</td>
</tr>
<tr>
<td>relative error</td>
<td>1.3%</td>
<td>-0.6%</td>
<td>-1.5%</td>
</tr>
</tbody>
</table>

For \( p = 3, b_c = -1, \) and \( s = 2 \), the respective values are listed in table 3.2; we note that the analysis agrees precisely with the simulation results. (3.69)
3.6 Robustness vs. Processing Speed

3.6.1 Uncoupled Templates

From section 3.2, we know that the settling time $T_S$ for a CNN programmed with an uncoupled template is given by

$$T_S = \frac{1}{a_c - 1} \log \left( \pm 1 + \frac{w_{wc}}{a_c - 1} \right).$$

(3.70)

where $w_{wc} := B \ast u_{wc} + I + \partial_i$, and $u_{wc}$ and $x_{wc}$ denote the worst case constellations of cells in the input image and in the initial state, respectively.

Let $\mathbf{p}$ be a correctly operating template vector and $T_S(\mathbf{p})$ the corresponding settling time. If $\mathbf{p}$ is scaled by a factor $q$, the logarithmic part in (3.70) will not change, since $\frac{w_{wc}}{a_c - 1}$ remains constant. However, $a_c - 1$ is multiplied by

Figure 3.13: Relative sensitivity of the settling time of GCD. $s = 2$, $p = 3$, $b_c = -1$. The dots are the simulation results, the lines represent the linear approximation from the sensitivity analysis.

evaluates to $11.3\alpha$; hence, a perturbation of $\alpha = 9\%$ may double the settling time.

3.6 Robustness vs. Processing Speed
\( q \), and it follows directly that for uncoupled templates, the settling time is \textit{inversely proportional} to the scaling factor \( q \),

\[
T_S(q \cdot \tilde{p}) = \frac{1}{q} \cdot T_S(\tilde{p}). \tag{3.71}
\]

In Chapter 2, it is shown that by scaling \( \tilde{p} \) by a factor \( q > 1 \), proportionally higher absolute robustness and a slightly higher relative robustness is achieved. Hence, for uncoupled templates, \textit{optimization for robustness and optimization for speed go hand in hand}: it is not a trade-off.

Depending on the initial state, there is an additional degree of freedom we can exploit to obtain a faster template without sacrificing robustness. For bipolar initialization, \( \tilde{p} = a_c - 1 \) may be freely chosen within some range \([0, c]\). In order to optimize the processing speed, \( \tilde{p} \) ought to be as large as possible.

\( x(0) = -1 \) : From the system of inequalities, \( c := \tilde{p} - I \) is fixed. If \( c > 0 \) we may choose \( \tilde{p} \in [0, c] \) and \( I = \tilde{p} - c \) without affecting the robustness. If \( c \leq 0 \), then \( \tilde{p} = 0 \) and \( I = -c \) is the only optimally robust solution. Hence, for \( c > 0 \), \( \tilde{p} := c \) results in maximum processing speed.

\( x(0) = 1 \) : In this case, \( c := \tilde{p} + I \) is determined by the system of inequalities. If \( c > 0 \), all \( \tilde{p} \in [0, c] \), \( I = c - \tilde{p} \) are optimum values, whereas for \( c \leq 0 \), only one optimally robust solution exists, namely \( \tilde{p} = 0 \), \( I = c \). Again, \( \tilde{p} := c \) is optimally fast for \( c > 0 \).

\( x(0) = u \) : Here, \( c := \tilde{p} + b_c \). The optimum solutions for \( \tilde{p} \) and \( b_c \) are \( \tilde{p} \in [0, c] \), \( b_c = c - \tilde{p} \) if \( c > 0 \), and \( \tilde{p} = 0 \), \( b_c = c \) if \( c < 0 \). Also in this case, \( \tilde{p} := c \) is the best choice if \( c > 0 \).

**Example 3.10 (Uncoupled horizontal line detection)**

Using input initialization, we obtained in example 3.3

\[
A = [2\gamma + 1]; \quad B = [\gamma \ \ 0 \ \ \gamma]; \quad I = -\gamma \quad \text{and} \quad A = [1]; \quad B = [\gamma \ \ 2\gamma \ \ \gamma]; \quad I = -\gamma
\]

as the extreme cases of optimally robust templates. (\( c := \tilde{p} + b_c = 2\gamma \) and &nolimits;\( \tilde{p}_{\text{opt}} \in [0, 2\gamma] \) for a safety margin of \( \gamma \).)

The settling time as a function of \( \tilde{p} \) is

\[
T_S(\tilde{p}) = \frac{1}{\tilde{p}} \log\left(\frac{2\tilde{p}}{\gamma} + 1\right), \quad \tilde{p} \in [0, 2\gamma]. \tag{3.72}
\]
(For \( \bar{p} = 0 \), either Bernoulli-de l’Hôpital’s rule or the formula \( T_S = \frac{-1 - x_{bc}}{w_{wc}} \) may be applied, resulting in \( T_S = \frac{2}{\gamma} \).)

Thus, as expected, increasing \( \bar{p} \) leads to higher processing speed (cf. example 3.3).

### 3.6.2 Propagation-type Templates

For this template class, the settling time is very robust with regard to template scaling, since it depends primarily on the ratio of the parameters, not on their absolute values (cf. section 3.4). If the ratio determines the settling time, we immediately conjecture that the fastest templates lie on the boundary of \( \mathcal{R} \) and thus have zero robustness.

Fig. 3.14 depicts robustness and speed contour plots for three typical tasks, reduced to two dimensions, and Table 3.3 presents optimally robust and optimally fast templates. It is easily seen that a joint optimization is difficult, since robustness has to be balanced against speed. However, if a CNN chip permits

<table>
<thead>
<tr>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
</tr>
<tr>
<td>SH</td>
</tr>
<tr>
<td>HLD</td>
</tr>
<tr>
<td>CCD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization for Robustness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
</tr>
<tr>
<td>SH</td>
</tr>
<tr>
<td>HLD</td>
</tr>
<tr>
<td>CCD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization for Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
</tr>
<tr>
<td>SH</td>
</tr>
<tr>
<td>HLD</td>
</tr>
<tr>
<td>CCD</td>
</tr>
</tbody>
</table>

**Table 3.3:** Optimum points in Fig. 3.14. The optimization for speed yields templates with robustness zero. The optimally robust templates are between 2.3 and 8.3 times slower than the optimally fast ones.

a reproduction of the parameters with a relative accuracy of, say, 8%, rather
Figure 3.14: Robustness and speed contour plots. The robustness (gray scale) resolution is 2%, and brighter colors depict higher robustness. In the speed plots, brighter colors indicate higher processing speed.

than creating a (slow) template with 20% robustness, we should focus on maximizing the processing speed to find the fastest template with 8% robustness. Thus, a sufficient degree of robustness has to be considered as a constraint which a template must satisfy, not as a quantity to be optimized. This new optimization problem may now be formulated as follows:

Find the fastest template with robustness \( D \) and \( \| \mathcal{T} \|_1 = \beta \).

(A larger robustness than \( D \) and/or a smaller \( L_1 \)-norm than \( \beta \) would be tolerable, but such a template is always slower than the fastest template with a robustness of \( D \) and \( \| \mathcal{T} \|_1 = \beta \).)

An approximate solution may be graphically found by investigating robustness and speed plots as depicted in Fig. 3.14, but this is possible only in two dimen-
3.6. Robustness vs. Processing Speed

... and does not yield accurate results. To solve the problem analytically, we exploit the results from the robustness analysis. From

\[ D = \frac{\gamma}{\|\mathbf{p}\|_1 + 1} = \frac{\gamma}{\beta}, \]

it follows that \( \gamma = D \cdot \beta \), i.e., the safety margin of the template is given by the product of both constraints, namely the robustness and the \( L_1 \)-norm. It remains to maximize the ratio of some template parameters (depending on the task) under the constraints

\[
\min_i (K \mathbf{\hat{p}})_i = D \cdot \beta \\
\|\mathbf{\hat{p}}\|_1 = \beta - 1.
\]  

(3.73)

(3.74)

Example 3.11 (Connected component detection)

Once again, we consider the CCD template prototype

\[ A = [s \ \bar{p} + 1 - s]; \quad B = [0]; \quad I = 0 \quad \text{with } 0 < \bar{p} < 2s. \]

In Sec. 3.4.3, we found that the settling time for CCD decreases monotonically with increasing \( s / \bar{p} \). With the coefficient matrix \( K \), \( K \mathbf{\hat{p}} \) evaluates to

\[
K \cdot [s, \ \bar{p}, -s]^t = \begin{bmatrix}
-1 & 1 & -1 \\
1 & -1 & -1 \\
1 & 1 & 1
\end{bmatrix} \cdot [s, \ \bar{p}, -s]^t = [\bar{p}, 2s - \bar{p}, \ \bar{p}]^t.
\]

Expecting that our template will turn out to be rather fast, i.e., \( s > \bar{p} \), the minimization yields \( \gamma = \bar{p} \). Hence, \( \bar{p} = \gamma = D \beta \) is already determined. By inserting \( \|\mathbf{\hat{p}}\|_1 = 2s + \bar{p} \) into (3.74), we obtain

\[ s = \frac{1}{2} (\beta - 1 - \bar{p}) = \frac{1}{2} (\beta (1 - D) - 1). \]

If, for example, a CNN chip has parameter perturbations of up to 10% and an \( L_1 \)-norm of 10 (\( D = 0.1 \) and \( \beta = 10 \)), the fastest template meeting these requirements is

\[ A = [4 \ 2 \ -4]; \quad B = [0]; \quad I = 0 \]

with a settling time of \( T_S = N \log \frac{4}{3} \). The commonly used CCD template \( A = [1 \ 2 \ -1] \), by comparison, settles at \( T_S = N \log 4 \), which is almost 5 times slower. On the other hand, its robustness is 25%, but such a high degree of robustness is rarely demanded.
3.6.3 Pseudo-gradient Optimization for Speed

Tuning those parameters which have a major influence an $T_s$ is another possible method for increasing the processing speed of a template. Using the sensitivity analysis proposed in section 3.5, we may formulate the following, very broadly applicable, iterative algorithm:

Start with a correctly operating template set $\mathcal{T}$ with a robustness greater than the limit $D$, and select the step size $q$.

1. Evaluate $S_{z_i}^{\hat{T}_s} \forall z_i \in \mathcal{T}$ and determine the index $m := \arg \max_i |S_{z_i}^{\hat{T}_s}|$.

2. Perform a step $z_m' := z_m - q \cdot \text{sgn} \left(S_{z_m}^{\hat{T}_s}\right)$.

3. Calculate the robustness $D'$ and the $L_1$-norm $\|\mathcal{T}'\|$ of the new template. If $D' > D$ and $\|\mathcal{T}'\| < \beta$ then accept the step $(z_m := z_m')$ and go back to 1. Otherwise, reject the step and terminate.

This algorithm may be improved by concentrating on parameters with positive $S_{z_i}^{\hat{T}_s}$, since they can be decreased to generate a faster template — with the advantage of a smaller $L_1$-norm.
3.7 Conclusions

In this chapter, we have analyzed the settling time of cellular neural networks. Different approaches were used for the two main classes of CNNs, the uncoupled and the propagation-type class. Furthermore, the mutual influence of processing speed and template robustness has been studied.

**Uncoupled templates.** For the classes of uncoupled and non-propagating coupled templates, the settling time $T_S$ of a CNN has been analytically determined. It depends on the input image, which is normally not known a priori. Hence, an upper bound $T^*_S \geq T_S$ is derived. This is the exact settling time for an input image containing the "worst case" constellation of cells. $T^*_S$ is the product of a factor $1/(a_c - 1)$ and a logarithmic expression including all parameters and the initial state $x(0)$. For this template class, $x(0)$ is not prescribed by the task but may be selected in order to achieve higher processing speed. The optimum initial state and the optimum boundary condition can be deduced from the proposed methodology.

It is shown that template scaling results in proportionally faster templates (Fig. 3.15(b)).

**Propagation-type templates.** For propagation-type templates, the settling time is approximated by the product of the length $L_P$ of so-called longest propagation strings and the one-cell propagation time $\Delta t$. A "worst case" input image yields a bound $L^*_P \geq L_P$, and for $\Delta t$, a tight upper bound $\hat{\Delta t}$ can be derived from the analysis of the state trajectories. The analytical expressions for $\hat{T}_S = L^*_P \cdot \hat{\Delta t}$ permit the processing time in CNN simulations, and in programs for CNN chips, to be predicted and shortened.

It is remarkable that the ratio of the center to the off-center elements in the $A$-template determines $T_S$, rather than their absolute value. One therefore simply has to increase the off-center entries or to decrease $a_c$ to raise the processing speed. Template scaling only has a substantial effect on the settling time if the parameters are in the order of 1 (Fig. 3.15(b)).

**Optimization for Robustness and Speed.** Clearly, a joint optimization for both speed and robustness is desirable. This is possible for uncoupled templates, where template scaling results in both proportionally higher speed and monotonically increased relative robustness (Fig. 3.15(a)).

For propagation-type templates, the fastest templates turn out to have zero
robustness — hence, there is a trade-off between speed and robustness. Noting that it is not sensible to design a template with a robustness which is higher than the relative inaccuracy of a CNN chip, it is advisable to concentrate on the processing speed (instead of balancing speed versus robustness), and to regard a sufficient degree of robustness as a constraint in the optimization process. The problem is now to find the fastest template with a robustness of \( D \) (at least) and a (maximum) \( L_1 \)-norm of \( \beta \). By means of the analysis of the robustness and the settling time, it is shown that this general class of problems can be solved in a straightforward manner.
Chapter 4

Unification of Continuous- and Discrete-time CNNs

4.1 Discrete-time CNNs

Discrete-time CNNs (DTCNNs) have been considered along with continuous-time CNNs (CTCNNs) both in terms of theory [44] and implementation as sampled-data systems [45]. The most common DTCNN is described by

\[
x_{ij}[k+1] = \sum_{mn \in \mathcal{N}_{ij}} a_{mn} \text{sgn}(x_{mn}[k]) + \sum_{mn \in \mathcal{N}_{ij}} b_{mn} u_{mn} + I.
\] (4.1)

Although sgn(·) may appear as a natural choice in the context of DTCNNs, there is no reason to exclude other nonlinearities. In order to define a more general type of DTCNNs we introduce a parameterized piecewise linear function \( f_c(\cdot) \) (Fig. 4.1) to be

\[
f_c(x) := \frac{1}{2c}(|x+c| - |x-c|), \quad 0 \leq c \leq 1.
\] (4.2)

Note that \( f_c(x) \) is linear for \(|x| < c\) with a slope of \(1/c\), and that

\[
\lim_{c \to 0} f_c(x) = \text{sgn}(x).
\] (4.3)
Chapter 4. Unification of Continuous- and Discrete-time CNNs

Figure 4.1: The nonlinear output function $f_c(x)$.

This nonlinearity is a generalization of the $\text{sat}(\cdot) \equiv f_1(\cdot)$ function. Instead of (4.1) we now propose the following equation

$$x_{ij}[k+1] = \sum_{mn \in \mathcal{N}_{ij}} a_{mn} f_c(x_{mn}[k]) + \sum_{mn \in \mathcal{N}_{ij}} b_{mn} u_{mn} + I,$$

(4.4)

which allows a continuous transition from $\text{sgn}(\cdot)$ to a $\text{sat}(\cdot)$ nonlinearity by varying the parameter $c$ from 0 to 1. This chapter aims at providing a unifying framework for CT- and DTCNNs with different types of nonlinearities. In Sec. 4.2.2, we introduce the $(\delta, c)$-CNN as a superset of the classes of CT- and DTCNNs with $\text{sat}(\cdot)$ and $\text{sgn}(\cdot)$ output functions, and Sec. 4.2.3 investigates robustness and template design issues.

In order to incorporate the discrete-time approach in a framework that allows passing to and from the continuous-time case, we introduce the delta operator approach to discrete-time CNNs, and denote it by $\delta$-CNNs [46,47]. This approach will encompass both the CTCNNs and DTCNNs. Due to the properties of the delta operator, a CTCNN can be recovered from a corresponding $\delta$-CNN as a limiting case by increasing the sampling rate.

4.2 Delta Operator Based CNNs

4.2.1 The Delta Operator

Given a continuous-time signal $x(t)$, we can obtain a discrete-time sequence $x(kT)$ by sampling $x(t)$ at a sampling rate of $T$. We denote the sampled
signal by \( x[k] \) for \( k \in \mathbb{Z} \). The delta operator is defined on the set of causal sequences [48] by

\[
\delta x[k] = \frac{x[k+1] - x[k]}{T}.
\]

(4.5)

Clearly, for \( x[k] = x(kT) \), i.e., for the sequence obtained by sampling a signal at the period \( T \), the delta operator is an approximation of the signal's derivative. In the limiting case of \( T \to 0 \), we obtain

\[
\lim_{T \to 0} \delta x[k] = \dot{x}(t)_{t=kT}.
\]

4.2.2 The \((\delta, c)\)-CNN

The dynamic equation of a delta-operator based discrete-time CNN is defined by

\[
C \delta x_{ij}[k] = -\frac{1}{R} x_{ij}[k] + A \ast f_c(x_{ij}[k]) + B \ast u_{ij} + I,
\]

(4.6)

where we apply the spatial convolution operator \( \ast \). Due to the two parameters \( T \) and \( c \), this class of CNNs is designated as \((\delta, c)\)-CNNs. By expanding (4.6), we obtain

\[
x_{ij}[k+1] = \left(1 - \frac{T}{\tau}\right) x_{ij}[k] + \frac{T}{C} (A \ast f_c(x_{ij}[k]) + B \ast u_{ij} + I)
\]

(4.7)

with \( \tau = RC \) as the internal time scale of the analog system. By inspection of (4.7), we can see that CTCNNs and DTCNNs are the limiting cases of \((\delta, c)\)-CNNs, as \( T \to 0 \) and \( T \to \tau \), respectively. In the blockdiagram in Fig. 4.2, the continuous-time integrator \( \int \) and the shift operator \( q \circ \bullet z^{-1} \), respectively, are replaced by the delta inverse operator \( \delta^{-1} \). Hence, by an appropriate choice of \( T \) and \( c \), the \((\delta, c)\)-CNN qualifies as a CT- or DTCNN with any of the nonlinearities \( sat(\cdot) \) or \( sgn(\cdot) \). In the following, for the sake of simplicity but with no loss in generality, we normalize both \( R \) and \( C \) to unity.

Since the edges in the \((T, c)\)-plane (Fig. 4.3) are reached by a continuous change of \( T \) and \( c \), any value \( 0 \leq T \leq 1, 0 \leq c \leq 1 \) is possible, which is particularly interesting for two reasons. On the one hand, high-gain CNNs \((c < 1) [11]\) and numerically forward Euler integrated CTCNNs, as discussed in Sec. 4.3, are covered. On the other hand, it can be expected that theoretical results for \((\delta, c)\)-CNNs are applicable to CT- and DTCNNs and vice versa.
Chapter 4. Unification of Continuous- and Discrete-time CNNs

Figure 4.2: Blockdiagram of a \((\delta, c)\)-CNN.

Figure 4.3: The \((T, c)\) plane.
For our considerations in the next section, we restrict ourselves to CNN operations that are feasible by all \((\delta, c)\)-CNNs. This excludes tasks with equilibria \(|y^*| = |f_c(x^*)| < 1\) (gray-scale output) and tasks whose equilibria depend on mutual dynamical influence of neighboring cells\(^1\), such as the Laplacian operator \([4]\). Most bipolar the CNN tasks, including propagating operations such as shadowing, connected component detection, hole filling, and global connectivity detection, however, can be done by all \((\delta, c)\)-CNNs. This implies that the equilibria of \((\delta, c)\)-CNNs do not depend on \(T\) or \(c\). Solving (4.7) for its equilibrium yields

\[
x_{ij}^* = R \cdot \left( A \ast f_c(x_{ij}^*) + B \ast u_{ij} + I \right),
\]

where, in fact, \(T\) is cancelled out, and \(c\) does not matter due to the assumption \(f_c(x^*) = 1\).

The stability is not influenced by \(T\) or \(c\), either. Stability proofs do not rely on a particular choice of nonlinearity but include the whole class of odd, bounded, and monotonic functions, which clearly is a superset of \(f_c(\cdot)\). For the so-called locally regular templates and \((8,c)\)-CNNs with \(0 \leq T \leq 1\), stability will be proved in Sec. 4.3.

### 4.2.3 Robust Template Design for \((\delta, c)\)-CNNs

The slope for \(|x| < c\) is \(-1 + a_c/c\); \(a_c > c\) therefore guarantees that all stable equilibria lie in the saturated region \(|x^*| > c\), resulting in a bipolar output (\(|f_c(x)| = 1 \forall |x| > c\)). Fig. 4.4 (a) shows two dynamic routes with a single equilibrium point, while in (b), a dynamic route with two stable equilibria is depicted. Note that for all stable equilibria \(|x^*| > c\).

**Definition 4.1 (Critical point.)**

A critical point in the trajectory of a cell is a point \(x_c[k_c]\), where the smallest absolute perturbation in \(\delta x[k]\) is sufficient to cause a wrong output at equilibrium.

**Lemma 4.1 (Set of critical points.)**

The set of critical points \(\mathcal{X}_c\) is

\[
\mathcal{X}_c := \left\{ x[k] \mid |x[k]| = c \right\}.
\]

---

\(^1\) A rigorous definition of such locally irregular tasks will be given in Sec. 4.3.
Chapter 4. Unification of Continuous- and Discrete-time CNNs

(a) Phase plot with two dynamic routes, \( a_c > c \).

(b) Phase plot with two stable equilibria, \( a_c > c \).

**Figure 4.4:** Phase plots for a \((\delta, c)\)-CNN.

**Proof:** A wrong output occurs if and only if the perturbation creates a new equilibrium point in any of the saturation regions, or if an existing equilibrium is eliminated. Suppose that the dynamic route of a cell suffers from a translation parallel to the \( \delta x \) axis, caused by perturbations in the template set. Since the dynamic route in the linear region \( |x| < c \) is a straight line with positive slope in the phase plot, equilibrium points are created or eliminated exactly when one of the points \( \delta x|_{|x|=c} \) toggles its sign. Any horizontal translation is not more critical than this vertical shift, due to \( |x^*-c| > \delta x|_{|x|=c} \).

This lemma can be verified in Fig. 4.4, where \( \Lambda \) denotes the safety margin \( \delta x|_{|x|=c} \). Note that bipolar initialization is assumed. A single local rule for a CNN task may be formulated as

\[
-x + k^T p > 0 \quad \text{or} \quad -x + k^T p < 0,
\]

depending on the desired sign for \( \delta x \). \( k \) is a vector of the same dimension as the template vector \( p \) and represents the (bipolar) configuration of the neighboring cells. If (4.9) is satisfied at the critical points, then it is guaranteed that the correct operation is performed. Taking \( a_c \) out of \( p \) and noting that \( |f_c(x_c)| = 1 \), we obtain

\[
(-c + a_c) f_c(x_c) + \hat{\mathbf{k}}^T \hat{\mathbf{p}} > 0
\]

(4.10)

for reduced coefficient and template vectors \( \hat{\mathbf{k}} \) and \( \hat{\mathbf{p}} \), respectively. Defining \( \tilde{a}_c := a_c - c \) and introducing a template vector \( \tilde{\mathbf{p}} \) which includes \( \tilde{a}_c \) instead
of \( a_c \), we may write (4.10) as
\[
\tilde{k}^t \tilde{p} > 0
\]
by appropriately choosing the signs of the coefficients in \( \tilde{k} \). With \( q \) as the number of local rules, we define a coefficient matrix \( K \in \mathbb{R}^{q \times m} \) to be
\[
K = \begin{bmatrix}
\text{sgn}(\tilde{k}_1^t \tilde{p}) \tilde{k}_1^t \\
\text{sgn}(\tilde{k}_2^t \tilde{p}) \tilde{k}_2^t \\
\vdots \\
\text{sgn}(\tilde{k}_q^t \tilde{p}) \tilde{k}_q^t
\end{bmatrix}
\]
(4.12)

By means of \( K \), the characterization of a CNN task simplifies to the following linear and homogeneous system of inequalities
\[
(K \tilde{p})_i \geq 0 \quad \forall 1 \leq i \leq q,
\]
(4.13)
which is solved for the optimally robust template vector by (cf. Chapter 2)
\[
\tilde{p}_{\text{opt}}(\gamma) = (K^t K)^{-1} K^t \gamma 1^m.
\]
(4.14)
\( \gamma \) is the minimum safety margin of the template set, and
\[
D(\tilde{p}_{\text{opt}}(\gamma)) = \frac{\gamma}{\|\tilde{p}_{\text{opt}}(\gamma)\|_1 + c} = \frac{1}{\|\tilde{p}_{\text{opt}}(1)\|_1 + \frac{c}{\gamma}}
\]
(4.15)
is its relative robustness. To obtain the optimum template for any value of \( c \leq 1 \), we have to add \( c \) to the value for \( \tilde{a}_c \) from (4.14). The safety margin does not depend on \( c \). In the case of \( c = 0 \) (SGN-CNN), (4.14) directly yields the optimum template, and the relative robustness is larger than for any \( c > 0 \) due to the smaller denominator in (4.15). If we dispose of an optimum template for any \( c \), we can easily determine the optimum template for any other \( c' \) simply by adding \( c' - c \) to \( a_c \). All other template entries remain unchanged. We are now ready to state the following theorem which is already proved by the above deduction.

**Theorem 4.1 (Robust template design for \((\delta, c)\)-CNNs.)**
*If, for any \((\delta, c)\)-CNN,*
\[
A = \begin{bmatrix}
a_1 & a_2 & a_3 \\
a_4 & a_c & a_6 \\
a_7 & a_8 & a_9
\end{bmatrix} \ ; \ B ; \ I
\]
is an optimally robust template set, then the optimally robust template set for a \((\delta, c')\)-CNN is

\[
A = \begin{bmatrix}
a_1 & a_2 & a_3 \\
a_4 & a_c - c + c' & a_6 \\
a_7 & a_8 & a_9
\end{bmatrix}; \quad B; \quad I,
\]

independent of the value of \(T\). The robustness given by (4.15) is monotonically decreasing with increasing \(c\). To directly design an optimum template vector, \(\hat{p}_{\text{opt}}(\gamma)\) has to be calculated (see (4.14)) yielding the solution for \(c = 0\). Simple addition of \(c\) to \(a_c\) gives the solution for any \(c \leq 1\).  

\[\square\]

**Example 4.1 (Connected component detection)**

For a SAT-CTCNN, the connected component detector

\[
A = [\gamma \gamma + 1 - \gamma]; \quad B = [0]; \quad I = 0
\]

is optimally robust with a safety margin of \(\gamma\) and a robustness of \(\gamma/(3\gamma + 1)\). The very same template set is optimal for any \((\delta, c)\)-CNN with \(c = 1\), including SAT-DTCNNs. For a \((\delta, c)\)-CNN with sgn-nonlinearity,

\[
A = [\gamma \gamma \gamma - \gamma]; \quad B = [0]; \quad I = 0
\]

is optimal with 33\% robustness, irrespective of \(\gamma\). In general,

\[
A = [\gamma \gamma + c - \gamma]; \quad B = [0]; \quad I = 0
\]

is the optimally robust connected component detector.

### 4.3 Efficient Numerical Integration of CNNs

#### 4.3.1 Numerical Integration and Sampling

The mathematical analysis of electronic circuits with discrete elements generally yields nonlinear or linear ordinary differential equations (ODEs). By applying vector notation, an ODE of any order can be recast in

\[
\dot{x}(t) = F(x(t)), \quad x(0) = x_0,
\]

where \(F(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^n\) is possibly nonlinear, and \(x(0)\) denotes the initial condition of the system. For non-trivial cases, (4.16) cannot be solved analytically
for \( x(t) \). Hence, one has to resort to numerical integration algorithms. We define a numerical integration operator \( \mathcal{N}_T \) as

\[
\mathcal{N}_T : C \rightarrow D, \quad x \mapsto \mathcal{N}_T x =: \hat{x}_d
\]

such that \( \hat{x}_d[k] \approx x(kT) \quad \forall k \in \mathbb{Z}_0^+ \) \hfill (4.17)

with \( C \) and \( D \) as the set of continuous and discrete functions, respectively:

\[
C := \{ f \mid f(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^m \}
\]
\[
D := \{ u_d \mid u_d[\cdot] : \mathbb{Z}^n \rightarrow \mathbb{R}^m \}
\]

Note that this definition is restricted to integration methods with constant step size \( T \). The difference between the (unknown) trajectory \( x(kT) \) and \( \hat{x}_d[k] \) is the integration error \( n_J([], k \right) \), i.e., \( \hat{x}_d[k] = x(kT) + n(J, k) \). For the choice of the step size \( T \), there is a tradeoff between accuracy and computational effort.

The well-known sampling operator \( \mathcal{S}_T \) is defined by

\[
\mathcal{S}_T : C \rightarrow D, \quad x \mapsto \mathcal{S}_T x =: x_d
\]

such that \( x(kT) = x_d[k] \quad \forall k \in \mathbb{Z} \), \hfill (4.19)

where \( x(\cdot) \) is a continuous-time signal of bandwidth \( B \), and \( x_d[\cdot] \) is the corresponding discrete-time signal. The sampling theorem states that the inverse sampling operator \( \mathcal{S}_T^{-1} \) allowing perfect reconstruction of \( x(\cdot) \) from \( x_d[\cdot] \) exists under certain conditions on the sampling rate \( T \) and the bandwidth \( B \):

- \( T < 1/2B \): Oversampling
- \( T = 1/2B \): Sampling at Nyquist Rate
- \( T > 1/2B \): Undersampling

In digital systems, \( x_d[k] = x(kT) \) cannot be achieved for almost any \( x(kT) \) due to the quantization error \( n_Q[k] \). (4.20) may then be written as \( \hat{x}_{dQ}[k] = x(kT) + n_Q[k] \).

The comparison of the sampling operator and the numerical integration operator in Table 4.1 shows that they are closely related. The question arises whether, in analogy to the sampling theorem, there is a numerical integration theorem which permits the optimization of the step size \( T \).
Chapter 4. Unification of Continuous- and Discrete-time CNNs

### Table 4.1: Comparison of the sampling and numerical integration operators.

<table>
<thead>
<tr>
<th>$\mathcal{J}_T$</th>
<th>$\mathcal{N}_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{C} \rightarrow \mathcal{D}$</td>
<td>$\mathcal{C} \rightarrow \mathcal{D}$</td>
</tr>
<tr>
<td>Sampling period $T$</td>
<td>Integration step size $T$</td>
</tr>
<tr>
<td>Exact function $x(\cdot)$ is known.</td>
<td>$x(\cdot)$ is unknown but determined by an ODE.</td>
</tr>
<tr>
<td>Quantization noise $n_Q[\cdot]$</td>
<td>Integration error $n_I[\cdot]$ and quantization noise $n_Q[\cdot]$</td>
</tr>
<tr>
<td>$|n_I| \gg |n_Q|$</td>
<td></td>
</tr>
<tr>
<td>Sampling Theorem</td>
<td>$\Rightarrow$</td>
</tr>
</tbody>
</table>

4.3.2 Forward Euler Integration and the Delta Operator

For a general first order differential equation

$$\dot{x}(t) = F(x(t)), \quad x(0) = x_0,$$

we obtain an approximate solution for $x(t)$ by

$$x(T) = x(0) + T F(x(0)), \quad x(2T) = x(T) + T F(x(T)), \quad \ldots, \quad (4.21)$$

where, in every step, an error of $O(T^2)$ is introduced. The integrated system is inherently a discrete-time system, which will be indicated by square brackets in the following. It is not trivial to find an optimum value for $T$, since there is a trade-off between computational effort and accuracy.

Solving (4.21) for $F(x[nT])$ yields

$$F(x[nT]) = \frac{x[(n+1)T] - x[nT]}{T}. \quad (4.22)$$

Comparing this with the definition of the delta operator (4.5), we see that the delta operator is in fact a forward Euler operator with $k = nT$, approximating the derivative of $x(\cdot)$.

Hence, the label $\delta_T$ CNN is appropriate to denote a CNN that is forward Euler integrated with step size $T$. Equivalently, a $\delta_T$ CNN can be viewed as a delta-operator based CNN with $T$ being the sampling rate, according to (4.5). Note that a $\delta_0$ CNN is a CTCNN, a $\delta_r$ CNN is a DTCNN.
4.3. Efficient Numerical Integration of CNNs

4.3.3 The CNN Sampling Theorem

Assuming that the main purpose of numerical integration of CNNs is to find the equilibrium state, we investigate under which circumstances a step size as large as the cell’s time constant $\tau$ yields the desired result.

**Definition 4.2 (\(\tau\)-integrable CNNs.)**
A CNN for which \(x[n\tau] = x^*\) for \(n \to \infty\) is a \(\tau\)-integrable CNN.

**Definition 4.3 (\(\tau\)-integrable template sets.)**
A template set \(\mathcal{T}\) is \(\tau\)-integrable if the CNN programmed with \(\mathcal{T}\) is \(\tau\)-integrable for all \(x(0), u \in \mathbb{R}^n\).

**Lemma 4.2 (Zero crossings.)**
If the number of zero crossings in a \(\delta_\tau\) CNN is the same as in a CTCNN for each cell, then the equilibrium of both systems will be identical, i.e., \(x[nT] = x^*\) for \(n \to \infty\).

Proof: \(y_{ij}^*\) is determined by the sign of \(x_{ij}(0)\) and the number \(Z_{ij}^0\) of zero crossings in the transient. If \(x_{ij}(0) = 1\), even \(Z_{ij}^0\) lead to \(y_{ij}^* = 1\), odd \(Z_{ij}^0\) to \(y_{ij}^* = -1\), and vice versa:

\[
y_{ij}^* = x_{ij}(0) \cdot (1 - 2(Z_{ij}^0 \mod 2))
\]

\(Z_{ij}^0\) is defined to be the cardinality of the set of all points of time for which \(x_{ij}(t) = 0\) and \(\dot{x}_{ij}(t) \neq 0\), whereas for the \(\delta_\tau\) CNN, \(Z_{ij}^T\) is

\[
Z_{ij}^T := \left| \left\{ n > 0 \mid x_{ij}[nT]x_{ij}[(n-1)T] > 0 \right\} \right|.
\]

Hence, the output values at equilibrium are identical if \(Z_{ij}^0 = Z_{ij}^T\). It remains to show that identical \(y^*\) for both CT (1.5) and DT (4.7) systems lead to identical \(x^*\). From (4.8) we know that \(x^*\) is independent of the value of \(T\). This implies that if the effective output values at equilibrium \(y^*\) are equal for both systems, then the equilibrium states \(x^*\) are equal as well. \(\square\)

CNN tasks may actually be defined by prescribing the number of zero crossings in the state trajectory of each cell. Note that the equality of the number of zero crossings is not necessary, but sufficient, to guarantee the correct equilibrium. It may happen that by numerical integration, a pair of zero crossings is added or eliminated. According to this lemma, it suffices to ensure that no
zero crossings are added or removed by the integration process. To investigate for which class of templates this can be guaranteed in the case of an integration step size of $\tau$, we consider four different classes of cells, according to the following definitions.

**Definition 4.4 (Directly connected cells.)**

Two different cells $C_{ij}$ and $C_{mn}$ are directly connected if $|i - m| \leq r$ and $|j - n| \leq r$, i.e., $m, n \in N_{ij}$, and $a_{i-m,j-n} \neq 0$ and $a_{m-i,n-j} \neq 0$.

**Example 4.2 (Shadowing)**

For the shadowing template $A = [0 \ p \ 1 \ p], \ B = [0], \ I = p, \ p > 0$, there are no directly connected cells, since there is only one non-zero off-center entry in the $A$ template.

**Definition 4.5 (Dynamic influence.)**

A cell is under dynamic influence if there is at least one directly connected linear cell. If all directly connected cells are saturated, the cell is under static influence.

We note that in phases of static influence, the contribution from the neighbors is constant. No two different static phases can directly follow each other, there is always a phase of dynamic influence between.

- **Case 1: Saturated cells under static influence.**

**Lemma 4.3 (Saturated cells under static influence.)**

A saturated cell under static influence in a $\tau$-integrated CNN reaches its equilibrium$^2$ precisely within one single integration step.

**Proof:** When saturated, a cell’s trajectory follows

$$\tau \dot{x}(t) = -x(t) + q \implies x(t) = p e^{-t/\tau} + q,$$

where $p$ and $q$ are given by the neighbors’ input and output and the initial condition. $x(t)$ tends to an (intermediate) equilibrium $q$, there are no local extrema and no zero crossings in this phase of the transient. The integration algorithm has to ensure that $x[nT]$ also tends to $q$. Applying forward Euler

---

$^2$This may be an intermediate equilibrium point, since the cell may get under dynamic influence later. It may lie in the linear region.
4.3. Efficient Numerical Integration of CNNs

![Image of graph showing transient behavior in the saturation region and an Euler step with $T = \tau$.](image)

**Figure 4.5:** A CNN transient in the saturation region and an Euler step with $T = \tau$.

To (4.23) yields

$$x[(n+1)T] = x[nT] + T \frac{-x[nT] + q}{\tau}$$

$$= x[nT] \left(1 - \frac{T}{\tau}\right) + \frac{T}{\tau} q$$

(4.25)

It is immediately seen that for $T = \tau$, the (intermediate) equilibrium $q$ is reached in one single step (Fig. 4.5)! Generally, $x[(n+m)T]$ for any $m > 0$ may be expressed as

$$x[(n+m)T] = x[nT] \left(1 - \frac{T}{\tau}\right)^m + \frac{T}{\tau} q \sum_{i=0}^{m-1} \left(1 - \frac{T}{\tau}\right)^i.$$  

(4.26)

For any $0 < T \leq \tau$, $x[nT]$ will approach the correct equilibrium, since no zero crossings are added. If $T \neq \tau$, the equilibrium is not reached in finite time. For $\tau < T < 2\tau$, the state oscillates\(^3\) around $q$ but still converges, whereas for $T \geq 2\tau$, the Euler integrated system becomes unstable. We therefore conclude that $T := \tau$ is in fact the optimum choice.

\[\square\]

**Case 2:** Saturated cells under dynamic influence.

If the directly connected linear neighbors do not give rise to a local maximum

\(^3\)If the whole transient, and not just the saturated part, is taken into account, then the system may be unstable as soon as $T > \tau$. 

---

---

---
of $|x_S(t_0)|$, this case is similar to case 1, with the only difference that the equilibrium may not be reached in one single step.

If there is such an extremum, the sign of $\dot{x}_S(t)$ for $t > t_0$ is opposite to the sign of $x_S(t_0)$. It is insignificant whether this extremum occurs precisely at $t_0$ or at some $n_0 T > t_0$, when the neighbors may have saturated. If they cause a new sign of $\dot{x}_S(t)$ while being linear, they will still do that when in the saturation region. In the integrated system, $|x_S[nT]|$ may be larger than $|x_S(t_0)|$, but it will still be pushed in the right direction: if the sign of $\delta_T x_S[nT]$ is correct for $|x_S[nT]| = 1$, then it is correct for any $\alpha x_S[nT]$ for $\alpha > 1$.

Hence, it is guaranteed that the extremum is not eliminated for any step size $T > 0$, which implies the correct behavior of the integrated CNN.

**Case 3: Linear cells with static influence.**

Let us focus on a linear $C_L$ under static influence. All cells that are directly connected to $C_L$ must have constant output. The dynamics of $x_L(t)$ in the linear region is then governed by

$$C \dot{x}_L(t) = x_L(t) \left( a_c - \frac{1}{R} \right) + q + B \ast u_L + I,$$

(4.27)

where $q$ comprises the contributions from the neighbor's output values which are constant by assumption. For bipolar tasks, $a_{0,0} > 1/R$ is assumed. The solution then is a single exponential function with a positive argument, which guarantees that the stable equilibria lie in the saturation region. Hence, the sign of $\dot{x}_L(t)$ is determined by the output values of the neighboring cells and cannot change while in the linear region. Hence, there is one zero crossing in this linear region, which is guaranteed by any positive Euler step size $T$.

So far, we have proved the following lemma:

**Lemma 4.4 ($\tau$-integrability.)**

A template is $\tau$-integrable if for all $t \geq 0$, there are never two directly connected linear cells.

The last and most interesting case is the case where there are two directly connected linear cells, dynamically influencing each other.

**Case 4: Linear cells under dynamic influence.**

In this case, the CNN may not be $\tau$-integrable.
4.3. Efficient Numerical Integration of CNNs

Example 4.3 (A two-cell CNN, case 1)
Consider a class of simple two-cell CNNs with the template set

\[ A = [r \ p \ s], \quad s < r < -p + 1 < 0, \quad B = [0], \quad I = 0, \quad (4.28) \]

and an initial state of \( x_1(0), x_2(0) = -1 \). A zero dirichlet boundary condition is assumed. Both cells have positive derivatives at \( t = 0 \), \( \dot{x}_1(0) > \dot{x}_2(0) > 0 \), i.e., they will immediately become linear. The analysis of this linear system reveals that \( x_1(t) \) travels to the positive saturation region, whereas \( x_2(t) \) cannot cross the zero line: The system matrix has a positive and a negative eigenvalue, \( \lambda_{1,2} = p - 1 \pm \sqrt{rs} \). For \( x_1(t) \), the mode with the positive eigenvalue has a positive sign, and for \( x_2(t) \), it has a negative sign. This implies that there is a local maximum for \( x_2(t) \) in the negative linear region, and that the equilibrium points are \( x_1^* = p - s > 1 \) and \( x_2^* = -p - r < -1 \).

If the integration step size is chosen such that \( x_2[nT] \) becomes positive or even greater than one, oscillations will occur which will lead to one or more zero crossings and eventually cause the system to become unstable. (No other than the correct equilibrium point is possible for this system.) It follows from \( \dot{x}_2(0) = 1 - p - r \) that for \( T \geq 2/(1 - p - r) \), a local maximum in the positive saturation region exists, and the system is unstable. \( \tau \)-integrability is guaranteed if \( p + r \leq 0 \), which is only a small fraction of the templates defined by (4.28).

Example 4.4 (A two-cell CNN, case 2)
The other possibility of two cells entering the linear region is if two cells have different initializations and different signs in their derivative at \( t = 0 \). The simplest CNN with this property is again a two-cell CNN with a template of the class

\[ A = [r \ p \ s], \quad s > r > p - 1 > 0, \quad B = [0], \quad I = 0, \quad (4.29) \]

and an initial state of \( x_1(0) = +1, \ x_2(0) = -1 \). Again, \( x_2(t) \) never crosses the zero line. The equilibrium state is \( x_1^* = -p - s, \ x_2^* = -p - r \). From \( \dot{x}_2(0) = 1 - p + r \) we infer that a step size of \( T \geq 2/(1 - p + r) \) pushes the local extremum of \( x_2(t) \) in the positive saturation region, which results in infinite oscillations, whereas for \( r < p \), the template is \( \tau \)-integrable.

The classes of templates we studied in these two examples cover the only two possibilities by which trajectories can enter the linear region. If we can exclude that this occurs for any pair of directly connected cells, the transient contains no part that falls under case 4, and the template is \( \tau \)-integrable by lemma 4.4.
With this in mind, we define a subclass of all bipolar CNN tasks, the so-called locally regular class.

**Definition 4.6 (Local regularity.)**

A CNN is locally regular, if the linear regions of any pair of directly connected cells $C_i$ and $C_j$ do not overlap, i.e.,

$$\exists t \geq 0 \text{ such that } |x_i(t)| < 1 \text{ and } |x_j(t)| < 1.$$  \hfill (4.30)

Equivalently, a CNN is locally regular if and only if for all pairs of directly connected cells $C_i$ and $C_j$

$$x_i(t) \cdot \dot{x}_i(t) < 0 \Rightarrow |x_j(t)| \geq 1 \quad \forall t \geq 0.$$  \hfill (4.31)

A template set $\mathcal{T}$ is locally regular, if this condition is satisfied for any input and initial state. The other templates are locally irregular.

The definition of local regularity leads to the core theorem of this section:

**Theorem 4.2 (The CNN Sampling Theorem.)**

Locally regular templates are $\tau$-integrable. For a step size $T := \tau$, the equilibrium of (1.5) is reached precisely in a finite and minimum number of integration steps.

**Proof:** This is a consequence of the fact that locally regular templates do not allow directly connected cells to enter the linear region. Hence, lemma 4.4 applies, and all cells travel to the correct saturation region, where lemma 4.3 ensures that within one step, the equilibrium state is reached precisely. $\square$

Note that whereas local regularity is a sufficient condition for $\tau$-integrability, it is not a necessary condition.

Condition (4.31) ensures that at most one of two directly connected cells becomes linear; any template can be tested for local regularity by evaluating (4.31) for all pairs of directly connected cells. In practice, due to symmetry and bipolarity, only a limited number of all possible configurations have to be checked, which is rapidly done on a computer.

If we analyze the template class $A = [r \ p \ s]$, $B = [0]$, $I = 0$ (cf. examples 4.3 and 4.4), we find that the condition for local regularity for a two-cell CNN with zero boundary condition is

$$|r + s| \leq p - 1.$$  \hfill (4.32)
4.3. Efficient Numerical Integration of CNNs

As expected, this region is completely disjoint from the regions that are given by the constraints in the examples, namely $s < r < -p + 1 < 0$ in example 4.3, and $s > r > p - 1 > 0$ in example 4.4. Interestingly, this region is in some sense orthogonal to the region of local diffusion, as defined in [49]. In Fig. 4.6, both regions are depicted for $p = 3$. In principle, one might suspect that propagation-type templates are locally regular, whereas diffusion-type templates are not, since propagation is intuitively assumed to be directed, in contrast to diffusion. A directed propagation of information would exclude that two cells mutually and dynamically influence each other. In fact, most propagation-type tasks such as shadowing, connected component detection, and global connectivity detection operate on a series of cells that can be considered as a set of dominoes, where the falling dominoes do not affect those already fallen. Such templates are locally regular, but there exists another group of propagation-type templates that is not.

Often, CNN tasks are characterized by a set of local rules (cf. Def. 2.1 on p. 11).

**Corollary 4.1 (Local rules.)**

Templates satisfying a set of local rules are locally regular.

**Proof:** For a task defined by a set of local rules, the linear regions of neighboring cells must not overlap. Therefore, lemma 4.3 applies. \qed
Chapter 4. Unification of Continuous- and Discrete-time CNNs

The operation of uncoupled templates can be specified by a set of local rules, since there is no influence from the neighbors' output. Thus they are locally regular and \( \tau \)-integrable.

**Example 4.5 (Connected component detection)**

An instance of a CCD template is \( A = [1 \ 2 \ -1] \). In Fig. 4.7, the transients of a CT-CNN and a \( \delta_\tau \) CNN for the CCD operation

\[
A = [1 \ 2 \ -1] : \quad \begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet
\end{array} \quad \Rightarrow \quad \begin{array}{cccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet
\end{array}
\]

are plotted. In the DT case, not only the time is discrete, but also the values of \( x[nT] \). Because the template consists of integer values, all possible \( x[nT] \) are also integers.

\( ^4 \)In fact, it is a \( \delta_{0.01} \) CNN — CT-CNNs cannot be simulated.
There is an infinite number of templates performing the CCD operation, and not all of them satisfy these local rules. \( A = [2 \ 1.25 \ -2] \) clearly is a CCD template, but there are local extrema in the linear region, which is forbidden by the local rules. According to (4.32) this template is still locally regular. The question arises whether it is true that the property of being locally regular can be generalized from any locally regular template to all templates performing the same task. From lemma 4.2 we know that a task is defined by the number of zero crossings for each cell. The existence of a \( \tau \)-integrable CNN for a task proves that the number of zero crossings does not depend on directly connected linear cells. Since any template performing the same task leads to the same number of zero crossings for each cell, a useful corollary can be stated.

**Corollary 4.2 (Equivalence relation.)**

*If a template is \( \tau \)-integrable, this holds for the whole equivalence class of templates performing the same mapping.*

Since the template space \( \mathcal{T} \in \mathbb{R}^{19} \) is infinite, whereas the set of possible CNN mappings \( M \)

\[
M : \mathbb{B}^n \times \mathbb{B}^n \rightarrow \mathbb{B}^n, \quad y^* := M(u, x(0) | \mathcal{T}).
\] (4.33)

is finite, there must, at least for some mappings, be an infinite number of equivalent templates. This equivalence relation divides \( \mathbb{R}^{19} \) in at most \( 2^{2^{2n}} \) equivalence classes.

Other well-known propagation-type templates such as hole filling and global connectivity detection can also be specified by a set of local rules, and there actually are template solutions that prevent directly connected cells from being linear at the same time. Hence, \( \tau \)-integrability is guaranteed for these tasks.

An example of a locally irregular template is the Laplacian template [4]. We will, however, concentrate on a one-dimensional template.

**Example 4.6 (A locally irregular template)**

Consider the local diffusion template \( A = [4 \ 3 \ 5] \) acting on a 8 cell CNN. It performs the mapping

\[
x(0) : \begin{array}{ccccccc}
\text{■■■■■■■□}
\end{array} \quad \Rightarrow \quad y^* : \begin{array}{ccccccc}
\text{□■■■■■■■□}
\end{array}.
\]
Cells $C_3$ through $C_6$ all enter the linear region at $t = 0$, and they are pairwise directly connected. We therefore suspect that $\tau$-integration of this CNN will lead to a wrong equilibrium. In fact, the output of $C_4$ turns out to be black in the case of a $\delta_\tau$ CNN.

It is easily seen that there is no set of local rules that describes this task: the initial state configuration $\begin{array}{c} \downarrow \end{array}$ of $C_2 \ldots C_4$ results in $\begin{array}{c} \downarrow \end{array}$, whereas for the identical initial configuration of $C_4 \ldots C_6$, the output at equilibrium is $\begin{array}{c} \square \square \square \square \square \square \end{array}$.

In the case of locally irregular templates, there is some ongoing race occurring between directly connected linear cells. A cell $C_L$ may change its sign before reaching the saturation region, depending on the speed of another linear cell. Thus, the equilibrium $x_L^*$ is sensitive to the ratio of the time constants of both cells. For the template in example 4.6, deviations of the time constants of less than 10% from the nominal value $\tau$ cause the operation to fail: The equilibrium of $C_4$ will be wrong if $\tau_4$ is perturbed by only $-6\%$, or $\tau_3$ or $\tau_5$ by $7\%$, or $\tau_6$ by $-7\%$, although $C_6$ is not even directly connected to $C_4$.

Fig. 4.8 shows the transients of $C_3$ through $C_6$ for nominal values of all time constants (left) and for the case of a perturbation of $\tau_4$ by $-6\%$ (right).

In the general case of two directly connected linear cells $C_1$ and $C_2$ whose neighbors are saturated, their dynamics is governed by

$$
\begin{bmatrix}
\dot{x}_1(t) \\
\dot{x}_2(t)
\end{bmatrix} =
\begin{bmatrix}
p - 1 & s \\
r & p - 1
\end{bmatrix}
\begin{bmatrix}
x_1(t) \\
x_2(t)
\end{bmatrix} +
\begin{bmatrix}
q_1 \\
q_2
\end{bmatrix}.
$$

Without loss of generality, we may assume that $C_1$ and $C_2$ are horizontal neighbors, and the $A$ template is $A = [r \ p \ s]$. ($A$ may have higher connec-
tivity, but the other parameters are irrelevant for our investigation; their influence is subsumed under the constant contribution from bias and neighbors.) Assume that the nominal time constant is \( \tau = 1 \), and that \( \tau_2 \) is perturbed, i.e., \( \tau_2 \neq 1 \). This changes (4.34) to

\[
\dot{x}_2(t) = \frac{p-1}{\tau_2} x_2(t) + \frac{q_2}{\tau_2},
\]

which leads to a different system matrix, and, in turn, to different eigenvalues \( \lambda_1, \lambda_2 \) and coefficients \( C, q' \) in the transients

\[
\begin{bmatrix}
x_1(t) \\
x_2(t)
\end{bmatrix} = C \begin{bmatrix} e^{\lambda_1 t} \\ e^{\lambda_2 t} \end{bmatrix} + q'.
\]

The effect on the dynamics of the system are not generally predictable; a stable mode may become unstable, or any coefficient may change its sign, resulting in a wrong equilibrium point, as in the example above.

On the other hand, locally regular templates are very robust with respect to mismatch of cell’s time constants. Considering (4.7), it is not surprising that robustness against deviations of \( T \) and \( \tau \) go hand in hand, since both affect the dynamics of the network in a similar manner. In fact, simulations show that locally regular templates, including CCD, which is often considered as a critical case, can easily bear inaccuracies of \( \tau \) of 80% and more!

In VLSI implementations of the CNN universal chip, the time constants will not match very precisely, and the accuracy cannot be guaranteed to be better than 10%. This is the reason why one mostly relies on locally regular templates for most applications of CNN chips. Locally irregular ones, in contrast, do not play a significant role in applications of CNN chips — they are often too sensitive to manufacturing inaccuracies.

## 4.4 Summary

We investigated discrete-time CNNs in the framework of the delta operator. This allowed us to generalize the results concerning the robustness measure, known for the continuous-time case, to the discrete-time case.

\((\delta,c)\)-CNNs were introduced as new class of CNNs. They are based on the delta operator which allows a continuous transition between continuous-time and discrete-time systems. CT- and DTCNNs appear as limiting cases for
$T \to 0$ and $T \to 1$, respectively. The output nonlinearity is defined by the parameter $c$ denoting the width of the linear region. Four important standard types of CNNs are located as the four corners in the $(T,c)$ plane, bounded by $0 \leq T,c \leq 1$. Hence, $(\delta,c)$-CNNs comprise all these standard CNNs and allow the development of a unified theory. While stability issues do not depend on $T$ or $c$, the robustness of a template set decreases slightly with increasing $c$. Restricting ourselves to the important class of locally regular templates, the analysis of the dynamic route shows that results for robust template design can be generalized in a very direct fashion for $(\delta,c)$-CNNs. Surprisingly, $c$ has simply to be added to the center element of the $A$-template in order to obtain the optimally robust template for any $c$. All other template parameters remain unchanged.

The CNN sampling theorem 4.2 shows that the locally regular class can be numerically integrated with a step size $T$ as large as the internal time constant $\tau$ of the CNN cells, while it is still guaranteed that the trajectories of the integrated system tend to the correct equilibrium point. Even more intriguing is the fact that with this particular choice of $T := \tau$, the equilibrium is reached precisely in a finite number of steps, and, at the same time, with minimum computational effort. Smaller step sizes (oversampling) increase the number of steps, whereas larger values (undersampling) lead to oscillations or even instability.

Often, rather sophisticated integration methods such as 4th order Runge-Kutta algorithms are used to simulate the dynamics of a CNN — we show that this is not necessary, as long as one is not interested in very accurate trajectories. Most learning algorithms, for example, are based on the distance between the actual and the desired output vector. They can be drastically accelerated simply by using an Euler algorithm with the proposed step size of $T = \tau$.

The small class of templates that is not included in our theorem is only marginally important for practical applications of a VLSI CNN chip, since the operation of the cells of this class requires precisely identical cell time constants.

Finally, due to the equivalence of DTCNNs and Euler integrated CNNs, the theorem proves that for the locally regular class, the same templates can be used for both continuous- and discrete-time operation. Locally irregular templates, however, may not lead to the desired result in DTCNNs, and they are bound to fail if a sgn($\cdot$) nonlinearity is used instead of the continuous sat($\cdot$) function.
Chapter 5

Stochastic Optimization

5.1 Introduction

The applicability of the template design method proposed in Chapter 2 is restricted to single-layer CNNs with bipolar outputs and linear templates, and to cases where a set of local rules is specified. For all other template design problems, one has to resort to more general optimization techniques, which will not provide analytical but numerical solutions.

A very generally applicable class of optimization algorithms is the class of stochastic methods. These methods try to cleverly derive benefit from the information obtained by random sampling the search space, which consists, in our case, of all possible template sets. A number of candidate solutions is generated randomly, and, depending on their performance, new and desirably better solutions are produced. This process is repeated iteratively until a solution is found that meets the requirements. An objective or cost function is used as a quality measure for the candidate solutions.

The search space, whose dimension equals the number of free parameters, is huge. Assuming a moderate parameter resolution of 0.25 and a range of $[-5, 5]$ we have $41^{19} \approx 10^{31}$ different template sets. Fortunately symmetries reduce the number of free parameters, but the search space remains in most cases very large. Exhaustive search is very unlikely to yield a satisfactory solution within finite time, and in most cases the nonlinearity, multimodality,
and discontinuity of the objective function impede the use of gradient-based search algorithms for template design problems. But this is precisely the class of problems for which stochastic techniques are the appropriate tools.

Two specific algorithms have been explored for CNN-related issues, genetic algorithms [23, 50] and simulated annealing [24, 50]. Both methods are inspired by nature. A self-contained introduction and comparison of these techniques is found in [51].

Simulated annealing (SA) is a process that is derived from statistical mechanics and thermodynamics. It exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system. The algorithm is based upon that of Metropolis et al. [52], which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. Kirkpatrick [53] noted the connection between this algorithm and mathematical minimization and proposed it as an optimization technique.

SA's major advantage over other methods is an ability to avoid becoming trapped in local minima. The algorithm starts at a randomly selected point in the search space and employs a random search which does not only accept changes that decrease the cost function \( f(\cdot) \), but also some changes that increase it. The latter are accepted with a probability

\[
P = \exp\left(-\frac{\Delta f}{T}\right),
\]

where \( \Delta f \) is the increase in \( f \) and \( T \) is a control parameter, which by analogy with the original application is known as the system "temperature". In the optimization process, \( T \) is decreased slowly according to a cooling or annealing schedule, which reduces the probability of acceptance for uphill steps. The principle underlying the choice of a suitable annealing schedule is easily stated — the initial temperature should be high enough to 'melt' the system completely and should be reduced towards its 'freezing point' as the search progresses.

For the template design problem, SA and particularly its more elaborate companion adaptive SA [54] have demonstrated their capabilities to perform an efficient search [24, 50]. The drawback of SA is its sensitivity to the (random) choice of the starting point, and the fact that its search space is inherently continuous. Since the set of programmable values on most CNN chips is discrete, a discrete search method is expected to be more efficient.
Genetic algorithms overcome these two shortcomings. They are discussed in detail in the following sections.

5.2 Genetic Optimization of CNNs

5.2.1 Genetic Algorithms

The basic principles of genetic algorithms (GAs) were first proposed by Holland [55]. GAs are inspired by the mechanism of natural selection where stronger individuals are the likely winners in a competing environment. GAs use a direct analogy of such natural evolution. Through the genetic evolution method, an optimal solution can be found and represented by the final winner of the genetic game. When discussing GAs, it is common to use biologists' terminology. A few of the most common terms are explained in Table 5.1.

GAs presume that the potential solution of any problem is an individual and can be represented by a set of parameters. These parameters are regarded as the genes of a chromosome and can be structured by a string of values in binary form. The value of an objective function, in the context of GAs generally known as fitness function, is used to reflect the quality of the chromosome. Throughout the genetic evolution, the fitter chromosome has a tendency to yield high quality offspring which means a better solution to the problem. In a practical GA application, a population of chromosomes has to be installed and these can be randomly set initially. The size of this population varies from one problem to another. In each cycle of genetic operation, termed an evolving process, a subsequent generation is created from the chromosomes in the current population. This can only succeed if a group of these chromosomes, generally called “parents” is selected via a specific selection routine. The genes of the parents are mixed and recombined for the production of offspring in the next generation. It is expected that from this process of evolution (manipulation of genes), the fitter chromosome will create a larger number of offspring, and thus a higher chance of surviving in the subsequent generation, emulating the survival-of-the-fittest mechanism in nature. A GA therefore works similarly to natural evolution and differs from traditional search methods in four ways [56]:

1. GAs work with a coding of the parameter set, not the parameters themselves.
**Chromosome.** A bitstring representing the encoded version of a candidate solution. The parameter set can be retrieved by splitting the chromosome into its genes and decoding them.

**Gene.** A substring of a chromosome that encodes a single parameter value.

**Genotype.** The set of all genes of a chromosome.

**Phenotype.** The parameter set.

**Individual.** A chromosome and its phenotype.

**Locus.** The place of a gene in the chromosome.

### Table 5.1: Short glossary of GA-related terms originating from biology.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Range:</td>
<td>[-4, 3.5]</td>
</tr>
<tr>
<td>Resolution:</td>
<td>0.5</td>
</tr>
<tr>
<td>Parameter set: (Phenotype)</td>
<td>-3.5 0.5 1.0</td>
</tr>
<tr>
<td>Binary representation: (Genotype)</td>
<td>0001 1001 1010</td>
</tr>
<tr>
<td>Bitstring: (Chromosome)</td>
<td>000110011010</td>
</tr>
</tbody>
</table>

**Figure 5.1: Parameter encoding.**

2. GAs search from a population of points, not from a single point.

3. GAs use only fitness function information, not derivatives or other auxiliary knowledge.

4. GAs use probabilistic transition rules, not deterministic rules.

Note that by 1. and 2., a GA contrasts with other stochastic techniques such as simulated annealing.

A characteristic feature of GAs is the parameter coding. The parameter set (in our case the templates entries) is encoded into a single bitstring. The parameters are converted into binary format using some coding scheme, and these binary substrings are concatenated to form the so-called chromosome (an example is given in Fig. 5.1). Many other coding methods have been proposed [57], but this is the simplest and performs well on many problems.
5.2. Genetic Optimization of CNNs

Genetic Operators

To achieve the goal of iteratively creating child populations that, on average, perform better than the parent population, three stochastic operators are sufficient: selection (sel) or reproduction, crossover (cross), and mutation (mut). Let \( \mathcal{P}^N \) denote a population of size \( N \), and \( \mathcal{P}^N \) the set of all possible populations of this size. All three operators take a population \( \mathcal{P}^N \) as argument, and yield a population of the same size: \( \text{sel, cross, mut} : \mathcal{P}^N \rightarrow \mathcal{P}^N \).

Selection. The selection operator \( \text{sel} \) picks \( N \) individuals \( s_1, \ldots, s_N \) from a population \( \mathcal{P}^N \) such that

\[
\bar{f}(\text{sel} \mathcal{P}^N) > \bar{f}(\mathcal{P}^N) \quad \text{and} \quad \forall i \in [1, N] \text{ s.t. } s_i \in \text{sel} \mathcal{P}^N \Rightarrow s_i \in \mathcal{P}^N, \tag{5.1}
\]

where \( s_i \) is any individual and \( \bar{f}() : \mathcal{P} \rightarrow \mathbb{R} \) is the average fitness of a population. To guarantee that the average fitness increases, the probability for a chromosome to be reproduced is a monotonically increasing function of its fitness, so the fitter ones are likely to be selected several times, while weak ones tend to die off. (5.2) indicates that the diversity\(^1\) in the population decreases in the selection process.

Mainly two selection schemes are used, fitness-proportional or roulette-wheel selection and tournament selection. In the former, as the name suggests, the probability of an individual to be selected is proportional to its fitness value. The latter randomly permutes the individuals in the population, pairwise compares their fitness, and selects the fitter one. In order to keep the population size constant, this process is repeated twice (Fig. 5.2).

Crossover. \( N/2 \) pairs of chromosomes are randomly taken from the population to be the parents of two offspring each (Fig. 5.3). In one-point crossover, the parents are aligned, and a crossing site between 1 and the length of the chromosomes is chosen randomly. Now, the substrings after the crossing site are exchanged to create the offspring. Accordingly, two-point crossover uses two crossing sites, and the substring in between is exchanged. More general is random crossover, where each bit in the chromosome is exchanged with probability 0.5. Crossover is usually performed with a probability \( p_c \) between 0.5 and 1. The crossover operator increases the diversity, since new chromosomes

\(^1\)The number of different chromosomes in the population.
Old population:

New population:

Figure 5.2: Example for tournament selection. The shapes represent the individuals, and the number below is the fitness value.

<table>
<thead>
<tr>
<th>Parent 1</th>
<th>0110110100</th>
<th>Parent 1</th>
<th>0101101000</th>
<th>Parent 1</th>
<th>0101101000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent 2</td>
<td>1100001111</td>
<td>Parent 2</td>
<td>1100001111</td>
<td>Parent 2</td>
<td>1100001111</td>
</tr>
<tr>
<td>Offspring 1</td>
<td>0110001111</td>
<td>Offspring 1</td>
<td>0100100100</td>
<td>Offspring 1</td>
<td>0101001010</td>
</tr>
<tr>
<td>Offspring 2</td>
<td>1101101000</td>
<td>Offspring 2</td>
<td>1101101111</td>
<td>Offspring 2</td>
<td>1100101011</td>
</tr>
<tr>
<td>crossing site</td>
<td></td>
<td></td>
<td></td>
<td>Random String</td>
<td>0010111010</td>
</tr>
</tbody>
</table>

One-point crossover Two-point crossover Uniform crossover

Figure 5.3: Crossover operators.
are created. On the other hand, if all chromosomes have the same bit value at a particular position, **cross** will never change that bit. This is the task of **mut**.

**Mutation.** The mutation operator randomly flips the bits of each chromosome with a low probability. By selection and crossover the algorithm may lose useful genetic material preventing the exploration of some regions of the search space. Mutation helps in such situations by creating new genes and thus increasing the diversity of the population. The mutation probability $p_m$ usually varies between 0.001 and 0.05.

One iteration in the optimization process, i.e., the transition from generation $k$ to generation $k + 1$ can be formulated as

$$P_{k+1}^N = \text{mut} \circ \text{cross} \circ \text{sel} P_k^N,$$

where $\circ$ denotes operator composition.

Before the next iteration is started, the fitness function has to be evaluated for every individual, which is usually much more expensive in terms of computing power than the three genetic operators.

### 5.2.2 Problem Statement

We first study the problem of designing robust templates for a given task:

$$p_{\text{opt}} = \arg \max_p \left\{ D(p) \mid y^*(p) = y_d \right\}$$

It is identical to the problem (2.9) in Section 2.3 (p. 16), where we restricted ourselves to locally regular templates.

Using GAs, the problem may be solved in two steps. First, find a template or a set of templates that perform a given task, neglecting robustness aspects. For this part of the problem, genetic algorithms have already been investigated in [23]. As a standard coding method, every free parameter was assumed to be in the approximate range of [-5,5] and coded with 10 bits, and these binary strings were concatenated to build a chromosome. The fitness function (5.5) was based on the Euclidean distance between the desired and the actual output image where $g(\cdot)$ denotes a linear scaling function, and $k$ is the number of
Figure 5.4: Two-stage genetic approach.

cells:

\[ f(p) = g \left( \sum_{i=1}^{k} (y_{di} - y^*_i(p))^2 \right) \]  

(5.5)

With roulette wheel selection and two-point crossover, this algorithm was shown to be successful for small numbers of free parameters. However, the templates it found were inherently not robust, and the accuracy of the values (≈ 0.01) is, on the one hand, too high for an efficient VLSI implementation, and on the other, it leads to long chromosomes which increases the computing time. Another disadvantage of this algorithm was that the (numerical) integration time of CNN was chosen randomly, which can result in high fitness values even in the case of instable CNNs with oscillating output values.

In this work, attempts were made to overcome these shortcomings. The basic idea behind our new approach is to use a first GA to generate a population of solutions (i.e., templates that perform a given task) without considering their robustness, and, in a second stage, to optimize the robustness using a second GA (Fig. 5.4).

5.2.3 Coding

From our considerations in Chapter 2 we conclude that an absolute accuracy of the parameters of 0.25 (i.e., the values are programmable in steps of 0.25) is sufficient, and that values greater than 5 are inconvenient to realize. Hence, 5 bits are sufficient for all template entries, and, as in [23], the strings are concatenated. A priori knowledge of CNN dynamics is used to reduce the chromosome length. Very often, for example, a given desired mapping has symmetrical behavior along one or even two axes, or it is obvious that some template entries have to be zero as the cells do not depend on their neighbors in certain directions. To guarantee bipolar output, the center element of the \( A \)-template, \( a_c \), is coded by 4 bits in the range from \([1.00, 4.75]\). This type of coding is used in both \( GA_1 \) and \( GA_2 \).
5.2. Genetic Optimization of CNNs

5.2.4 Search for Solutions (GA1)

This algorithm resembles the one proposed in [23]. The fitness function (5.6) is proportional to the number of correct output pixels.

\[
    f_1(p) = \beta \left( 1 - \frac{1}{2^k} \sum_{i=1}^{k} |y^*_i - y^*_i(p)| \right)
\]

(5.6)

where \( \beta = \begin{cases} 
    0.5 & \text{if } y(p) = 1^k \text{ (black) or } y(p) = -1^k \text{ (white)} \\
    1 & \text{otherwise.} 
\end{cases} \)

For tasks with only a relatively small number of black (or white) output pixels (e.g., a corner detector), this function yields a high fitness for templates that create completely white (or completely black) images \( (y_p = \pm 1^k) \). To prevent these templates from becoming too dominant, they are penalized by halving their fitness values (\( \beta = 0.5 \)).

The other parameters of GA1 are: population size \( n_0 = 64 \), roulette wheel selection, bit mutation rate \( p_m = 0.03 \), nonoverlapping populations, one-point crossover with probability \( p_c = 0.6 \). Compared to a standard GA, the main difference is the fact that during the whole process, every solution (\( f(p) = 1 \)) found in any generation is stored to become a member of Population II (Fig. 5.4). The algorithm terminates when the desired number of solutions \( n_1 = 32 \) has been found.

5.2.5 Robustness Optimization (GA2)

In this stage, the optimization problem (5.4) is to be solved. In a straightforward manner, the fitness function is defined to be equal to the robustness: \( f_2(p) = D(p) \). If the output is not correct, the fitness is 0. To avoid premature convergence\(^2\), the mutation operator has to be defined very carefully. \( p_m \) must not be chosen too small, but uniform mutation with relatively high probability is likely to generate a "wrong" template set, as high significant bits in the parameters may be affected which always results in a "bad" template. This leads to a considerable number of chromosomes with fitness 0, which in turn forces the reproduction operator to carry out the selection from the small remaining subpopulation with non-zero fitness. Therefore, in GA2,

\(^2\)A drastic loss of genetic diversity in early stages of the optimization process
mutation operates only on the 3 least significant bits of each encoded parameter, with $p_m = 0.05$. If, nonetheless, a newly created individual has fitness 0, it is randomly replaced by an individual with non-zero fitness of its parent generation (overlapping populations). The best solution is always preserved in the optimization process.

### 5.2.6 Parallel Implementation on a Supercomputer

The algorithm runs on an Intel Paragon XP/S-22 MP, a massively parallel supercomputer with 160 computing nodes, with 2 CPUs and 64MB RAM each. Due to the lack of shared memory, the inter-node communication has to be provided using message passing. The manager program containing the genetic algorithms is written in C and invokes a number of client routines which evaluate the fitness functions (5.6) for the population (Fig. 5.5). This entails

![Figure 5.5: Program structure.](image)

the numerical integration of a large system of nonlinear differential equations. The time it takes the system to reach its equilibrium depends strongly on the template set and cannot be predicted. Thus, the integration method should auto-detect the equilibrium state to terminate, and, to be efficient and accurate, provide adaptive integration step size control. This part of the program is written in Fortran 77 and uses a combined 4/5th-order Runge-Kutta method with a step width controlled by the maximum admissible relative error. In case of oscillating systems, it terminates after 500 timesteps and sets the fitness value to zero.

To determine the robustness of a template in GA2, the manager perturbs the template according to Def. 2.3 (Chapter 2) and lets the clients check if the condition $y_p = y(p)$ is still satisfied, i.e., if the template still operates correctly. If it does, the perturbation $\alpha$ is increased, otherwise it is decreased, until the desired accuracy of the robustness of 1% is reached. For a template set with
5.2. Genetic Optimization of CNNs

$m$ non-zero entries, $2^m$ combinations of perturbations have to be evaluated for each $\alpha$, and approximately 5 different values for $\alpha$ are necessary for 1% accuracy. Neglecting duplicate chromosomes in the population, the number of CNN integrations per generation in GA\textsubscript{2} is given by (5.7).

$$N_f \approx 5n_1 2^m$$

(5.7)

The inherently sequential genetic operators, namely selection, recombination and mutation typically consume less than 0.01% of the computation time.

5.2.7 Results

In this section, 3 examples with increasing complexity are presented. Note that we did not exploit local regularity, neither for the definition of the fitness functions nor in the numerical integration of the CNNs. Hence, the optimization routines are generally applicable to any problem where a desired output image is specified.

Example 5.1 (Shadowing)

The input image is stored in the initial state $x(0)$, there is no time-independent input $u$. In addition to the whole $B$-template, also the elements $a_1$, $a_2$, $a_3$, $a_7$, $a_8$, and $a_9$ of the $A$-template can be set to zero, since cells in different rows do not influence each other (row-wise operation). Moreover, cells are independent of their left neighbor; accordingly, $a_4$ is zero as well. In the template prototype (5.8), only $a_5$, $a_6$, and $I$ remain as optimization parameters.

$$A = [0 \ a_5 \ a_6]; \ B = [0]; \ I$$

(5.8)

The phenotype $a_5 \in \{1.00, 1.25, \ldots, 4.75\}$, $a_6, I \in \{-4.00, -3.75, \ldots, 3.75\}$ is coded into the 14bit-string

using standard binary coding (most significant bits are $s_1$, $s_5$, and $s_{10}$).

Fig. 5.6 shows after how many generations GA\textsubscript{1} has found $n_1 = 32$ (indicated by the dashed line) solutions. In 8 of 10 runs, this is achieved in only 2 generations. Already in generation 0 we find 6.7 correctly operating templates in the mean; this is not surprising as $\approx 10\%$ of the parameters in the search
space are shadowing templates. The diversity is fairly large, as there are 21 different chromosomes in Population II.

When it comes to robustness optimization (GA$_2$), Fig. 5.7 shows the histograms of the initial population (generation 0) and the 10$^{th}$ generation, summed over 10 runs. The average fitness (dashed line) increases from 11.1\% to 23.9\%, and in 9 runs the global optimum

$$A = [0 4.75 3.75]; \quad B = [0]; \quad I = 3.75$$

with a robustness of 30.6\% was found in less than 10 generations. This result corresponds to the analytical result in Example 2.2 on page 23.
5.2. Genetic Optimization of CNNs

Example 5.2 (Connected component detection)
The template prototype

\[ A = [a_4, a_5, a_6]; \quad B = [0]; \quad I \]

has 4 possible non-zero entries, and, using the same type of coding, the chromosomes consist of 19 bits. In 10 runs, the number of generations to find the 32 solutions for CCD was 4.5 in the mean and 6 in the worst case. The expected number of solutions increases linearly (Fig. 5.8, left). Note that GA$_1$ terminates when 32 correct templates are found. Fig. 5.8 (right) and Fig. 5.9 show the average and maximum fitness and the histograms, respectively, visualizing the performance of the robustness optimization. In the worst case of 10 runs, the solution after 10 generations had 25\% robustness, which is still far beyond the critical value which ranges between 5\% and 10\%. In most runs, the algorithm ends with a solution having the structure (5.11), which was ana-

**Figure 5.8:** CCD, GA$_1$ and GA$_2$.

**Figure 5.9:** CCD, histograms of GA$_2$.

**Example 5.2 (Connected component detection)**
The template prototype

\[ A = [a_4, a_5, a_6]; \quad B = [0]; \quad I \]

has 4 possible non-zero entries, and, using the same type of coding, the chromosomes consist of 19 bits. In 10 runs, the number of generations to find the 32 solutions for CCD was 4.5 in the mean and 6 in the worst case. The expected number of solutions increases linearly (Fig. 5.8, left). Note that GA$_1$ terminates when 32 correct templates are found. Fig. 5.8 (right) and Fig. 5.9 show the average and maximum fitness and the histograms, respectively, visualizing the performance of the robustness optimization. In the worst case of 10 runs, the solution after 10 generations had 25\% robustness, which is still far beyond the critical value which ranges between 5\% and 10\%. In most runs, the algorithm ends with a solution having the structure (5.11), which was ana-
lytically shown to be the most robust for CCD in Chapter 2. \( s \geq 2 \) results in a robustness \( \geq 28\% \).

\[
A = [s \ s+1 \ -s]; \quad B = [0]; \quad I = 0
\]  \hspace{1cm} (5.11)

**Example 5.3 (Hole filling)**

In this task, the input image is stored in \( u \) and the states are initialized with 1. Since there is symmetric propagation along both axes, the template has the structure given by (5.12).

\[
A = \begin{bmatrix}
0 & a_o & 0 \\
a_o & a_c & a_o \\
0 & a_o & 0
\end{bmatrix}; \quad B = \begin{bmatrix}
0 & 0 & 0 \\
0 & b_c & 0 \\
0 & 0 & 0
\end{bmatrix}; \quad I
\]  \hspace{1cm} (5.12)

\( a_o \) denotes the off-center elements in the \( A \)-template. Again there are 4 parameters to be optimized, but the higher connectivity makes it a considerably more difficult problem. Although all the off-center elements in \( A \) are supposed to have the same value, they have to be perturbed independently to determine the robustness of the template, which entails \( 2^7 = 128 \) CNN integrations due to the 7 non-zero entries in the whole template set. As expected, it takes significantly longer to generate 32 solutions: 55 generations in the mean, ranging from 16 to 142 in 10 runs. The robustness optimization algorithm creates templates with 8.2\% (±1.5\%) robustness, which is rather good compared with the global optimum of 9.8\% \( (a_c = 2, \ a_o = 1, \ b_c = 4, \ I = 0) \).

The evaluation of one generation of size 32, which implies the integration of more than 16'000 CNNs, takes about one minute on 128 nodes on the Paragon.

### 5.3 Other and Combined Approaches

#### 5.3.1 The Hill Climbing Approach

For the hill climbing approach, we assume that we already have a correctly operating template set \( \mathcal{T}_c \) (or parameter vector \( p_c \)) having the robustness \( D(p_c) \) at our disposal. By means of a *hill climbing* algorithm, we now seek an improved template with greater (ideally maximum) robustness. Since zero entries and symmetry conditions in \( p \) are to remain unaffected, we consider a reduced parameter vector \( r \in \mathbb{R}^q \), where \( q \) is the number of *free* template entries, i.e., parameters that can be altered independently. The steepest-ascent
5.3. Other and Combined Approaches

algorithm

\[ r_{\text{new}} = r_{\text{old}} + \beta \arg \max_{r \in \mathbb{R}^q} \left\{ D(r_{\text{old}} + \beta r) \right\} \]  

(5.13)

is repeated iteratively until \( r_{\text{new}} = r_{\text{old}} \) which indicates a local optimum. The step size \( \beta \) should equal the accuracy achievable in the CNN chip; \( \beta = 0.25 \) is a good choice which corresponds to the resolution in the GA. The following examples demonstrate that the local maximum found by this algorithm very often is the desired global maximum.

In Fig. 3.14 (p. 66), contour plots of the robustness landscapes for three typical CNN tasks, shadowing (SH), horizontal line detection (HLD), and connected component detection (CCD), are displayed. It can readily be seen that there are no local maxima. Accordingly, the algorithm (5.13) is bound to yield the global optimum. However, depending on the starting point \( r_e \), a considerable computational effort is needed. Assuming that 6 steps are sufficient to get to the optimum, and that \( 5 \cdot 2^m \) CNN integrations are necessary to determine the robustness of a template, we obtain \( 6 \cdot 3^q \cdot 5 \cdot 2^m \) as an approximation for the total number of integrations for this hill climbing algorithm. Typical values (e.g., for a hole filling template) are \( m = 7 \) and \( q = 4 \), resulting in over 300'000 integrations. Even on the Paragon, this takes almost an hour for medium size images (\( n \approx 1000 \) cells). On the other hand, this effort has to be made only once for every task.

5.3.2 The Averaging Approach

The idea behind this approach is that the arithmetical average of a whole set of templates performing the same operation is a good candidate for a robust template. To generate such a set of templates, we may use a local learning algorithm or profit from the fact that GAs produce not only one single solution but whole populations of solutions. Using GA1 from Sec. 5.2, for example, we end up with 32 solutions. Under certain conditions, their average can be expected to be robust. Let \( \mathcal{C} \) denote the search space of the GA, and \( \mathcal{R} \) the parameter subspace where the template operates correctly. If \( \mathcal{C} \cap \mathcal{R} \) is connected and convex, and if the solutions are, to some extent, uniformly distributed over \( \mathcal{C} \cap \mathcal{R} \), then the robustness of the average solution is expected to be larger than the average robustness of all solutions:

\[ D(\text{mean}(\mathcal{P}_i)) > \text{mean}(D(\mathcal{P}_i)) \]  

(5.14)

\(^3\)A symmetrical distribution with respect to the center of \( \mathcal{C} \) is sufficient.
Chapter 5. Stochastic Optimization

For GAs, the condition of "uniform" distribution is satisfied rather well, whereas local learning algorithms tend to find solutions along one single boundary of $\mathcal{R}$.

Example 5.4 (Shadowing)
Shadowing templates have only three non-zero entries, namely $p := a_5$, $s := a_6$, and $I$. In Fig. 5.10, three crossections through this threedimensional parameter space are shown; the shaded region indicates $\mathcal{R}$, and $\mathcal{C} \cap \mathcal{R}$ is bounded by a solid line. The squares represent the solutions found by a GA, the black circle their average. As expected, the average solution lies approximately in the center of $\mathcal{C}$ in each crossection. With a population size of 32, the mean gain in robustness in 10 different runs by this averaging operation was almost a factor of two (Fig. 5.11). Thus, (5.14) holds perfectly.

The advantage of this approach is that we do not have to evaluate the robustness of large sets of templates, which is computationally very expensive. Compared to the hill climbing algorithm, this method is up to three orders of magnitude cheaper, since a GA finds 32 shadowing templates in less than three generations, i.e., in less than 200 CNN integrations.

5.3.3 The Hybrid Approach

To combine the advantages of GAs and hill climbing algorithms, we propose a hybrid approach. We start with a GA to generate a population of correctly operating templates for a given task. In a second step, we use the average template as the initial template for the hill climbing algorithm (Fig. 5.12). This
5.3. Other and Combined Approaches

Figure 5.11: Comparison of the averaged robustness and the robustness of the mean template.

![Comparison of the averaged robustness and the robustness of the mean template](image)

The hybrid approach reduces the computational effort drastically, while providing optimum results. For the shadowing task, for example, the hill climber reaches the *global* optimum in 1.5 steps in the mean, even in the case of a moderate population size of 32. For larger populations, the number of steps tends towards zero. Compared with the purely genetic approach (Section 5.2), the GA used here corresponds to GA₁, whereas GA₂ is replaced by averaging and hill climbing.

**Example 5.5 (Global connectivity detection)**

With a population size of 64, the GA yields a first correct solution in 2.6 generations in the mean; in less than 12 generations, 32 solutions are found. Using the hybrid approach, we find

\[
A = \begin{bmatrix} 0 & 2 & 0 \\ 2 & 3 & 2 \\ 0 & 2 & 0 \end{bmatrix}; \quad B = \begin{bmatrix} 0 & -2 & 0 \\ -2 & 0 & -2 \\ 0 & -2 & 0 \end{bmatrix}; \quad I = 0
\]

with 10.4% robustness. Interestingly, the hybrid approach almost always tends to such globally optimum symmetric⁴ solutions although there exist locally optimum solutions with non-zero bias indicating an asymmetric behavior.

---

⁴In the sense that the template operates in a symmetric manner on both white objects on a black background and black objects on a white background
5.4 Attacking the General Classification Problem

5.4.1 The General Classification Problem

The problems considered so far were comparatively simple, since the input-output mapping \( M : (u, x(0)) \rightarrow y^* \) was given, and we restricted ourselves to bipolar tasks. The general classification problem, however, is much more complex — it will demonstrate that stochastic optimization techniques are in fact applicable to a very broad class of optimization problems.

We assume that we have a set \( \mathcal{U} \) consisting of \(|\mathcal{U}| \) real vectors \( u_i^j \in \mathbb{R}^n \). These vectors are organized in \( c \) clusters or classes. The respective class a vector belongs to is indicated by its upper index \( 1 \leq j \leq c \). The vectors within class \( j \) are numbered by the lower index \( 1 \leq i \leq q(j) \). Hence,

\[
\sum_{j=1}^{c} q(j) = |\mathcal{U}|.
\]

For each class, we define a class prototype \( \bar{u}^j \) by the arithmetical mean

\[
\bar{u}^j := \frac{1}{q(j)} \sum_{i=1}^{q(j)} u_i^j.
\]

To classify a vector \( z \), i.e., to decide to which class it belongs, we evaluate the classification function \( C(\cdot) : \mathbb{R}^n \rightarrow \{1, 2, \ldots, c\} \)

\[
C_\mathcal{U}(z) := \arg \min_{1 \leq j \leq c} \|z - \bar{u}^j\|_p,
\]

where \( \|\cdot\|_p : \mathbb{R}^n \rightarrow \mathbb{R} \) denotes the \( p \)-norm

\[
\|z\|_p = \begin{cases} 
\left( \sum_{i=1}^{n} |z_i|^p \right)^{1/p} & \text{for } p < \infty \\
\max_{1 \leq i \leq n} |z_i| & \text{for } p = \infty.
\end{cases}
\]

The index \( \mathcal{U} \) in \( C_\mathcal{U} \) indicates that the prototypes are based on the vectors in the set \( \mathcal{U} \). In cases where two or more prototypes \( v^j \) have the same distance to \( z \), we define \( C(\cdot) \) to yield the smallest class index \( j \) within this group of equidistant prototypes, which guarantees the uniqueness of \( C(\cdot) \). A classification is said to be correct if \( C(u_i^j) = j \) for any \( u_i^j \in \mathcal{U} \). The classification rate
$R$ of an entire vector set $Z$ is the number of correct classifications divided by the number of elements in $Z$.

$$R_{U}(Z) := \frac{1}{|Z|} \sum_{z_{j} \in Z} \left\{ \begin{array}{ll} 1 & \text{if } C_{U}(z_{j}) = j \\ 0 & \text{otherwise} \end{array} \right. ,$$  \hspace{1cm} (5.18)

where we tacitly assume that $Z$ and $\mathcal{U}$ have the same number of classes. In the context of training and testing a classification scheme, we may identify $\mathcal{U}$ as a training set, and $Z$ as a test set. The corresponding classification rates are then $R_{U}(\mathcal{U})$ and $R_{U}(Z)$, respectively. Since the prototypes are determined by $\mathcal{U}$, we would normally expect $R_{U}(\mathcal{U}) > R_{U}(Z)$.

To increase the classification rate, the vectors may be preprocessed in such a way that the intra-class distance is minimized while, at the same time, the inter-class distance is maximized (see Fig. 5.13):

$$\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{q(j)} \| u_{i}^{j} - \tilde{u}_{i}^{j} \|_{p} \quad \forall 1 \leq j \leq c \\
\text{Jointly} & \quad \sum_{i=1}^{c-1} \sum_{k=i+1}^{c} \| \tilde{u}_{k}^{j} - \tilde{u}_{i}^{j} \|_{p} \quad (5.19)
\end{align*}$$

In this section, we investigate the potential of bipolar CNNs for this preprocessing problem. Image or pattern recognition or classification is a key issue in CNN applications, and it is one of the areas where the CNN has a great potential.

### 5.4.2 Problem Statement

With (1.5), we can use the vectors that are to be classified directly as the CNN input and/or initial state, where, in the context of two-dimensional CNNs, we rather speak of images than of abstract vectors. Without loss of generality, we assume the images to be normalized to $[-1,1]$.

A CNN with bipolar output, programmed with the template set $\mathcal{T}$, performs a mapping

$$M : [-1,1]^{n} \times [-1,1]^{n} \rightarrow \mathbb{B}^{n} , \quad y^{*} := M(u, x(0) | \mathcal{T}) .$$

Let $\mathcal{M}(\cdot, \cdot, \cdot)$ be the set of processed images

$$\mathcal{M}(\mathcal{T}, \xi, \mathcal{U}) := \{ y \in \mathbb{B}^{n} | y = M(u_{j}^{j}, \xi | \mathcal{T}) \quad \forall u_{j}^{j} \in \mathcal{U} \} ,$$
Figure 5.13: The goal of the preprocessing.

where $\xi \in \{-1, 0, 1, u\}$ determines the initial state, i.e., whether $x(0) = -1$ (white), $x(0) = 0$ ('gray'), $x(0) = 1$ (black), or $x(0) = u$ (equal to the input image).

A first version of the general classification problem is that of finding the template set and initial condition that maximize the classification rate:

$$(T_{opt}, \xi_{opt}) = \arg \max_{T \in \mathbb{R}^{10}} R_U(M(T, \xi, U))$$

(5.21)

If $R$ reaches 1, no further optimization of (5.21) is possible, although, in view of the preprocessing goals (5.19) and (5.20), an improvement may still be achievable. Hence, some other objective function has to be included in the problem statement, such as the robustness of the classification. According to (5.16), the prototype with the minimum distance to an image determines to which class the image belongs. If two distances are (almost) equal, the classification is said to be sensitive; if the minimum distance is much smaller than the distance to any other prototype, the classification is robust.

With $d_1 := \min_j (\|z - u^j\|_p)$ as the minimum distance (cf. (5.16)), and $d_2 := \min_{j \neq C(z)} (\|z - u^j\|_p)$ as the second-smallest distance, we define the following robustness measures (see Fig. 5.14):
5.4. Attacking the General Classification Problem

![Diagram](image)

**Figure 5.14:** The range of the robustness $B$ for correct and wrong classifications.

For a correct classification:

$$
\text{Robustness } B(d_1, d_2) = \frac{d_2}{d_1} - 1 \in [0, \infty)
$$

(5.22)

For a wrong classification:

$$
\text{Robustness } B(d_1, d_2) = -\frac{d_2}{d_1} + 1 \in (-\infty, 0]
$$

(5.23)

Note that $d_2 \geq d_1$. For $d_2 = d_1$, the classification may either be wrong or correct. With $\bar{B}(Z)$ as the averaged quantities over an entire image set $Z$, we may now re-formulate the template optimization problem, including the aspect of robustness.

$$(\mathcal{F}_{\text{opt}}, \xi_{\text{opt}}) = \arg \max_{\mathcal{F} \in \mathbb{R}^{19}, \xi \in \{-1, 0, 1, u\}} \left\{ a \left( R_{\mathcal{U}}(\mathcal{M}(\mathcal{F}, \xi, \mathcal{U})) \right) + \beta \left( \bar{B}(\mathcal{M}(\mathcal{F}, \xi, \mathcal{U})) \right) \right\}.
$$

(5.24)

Since we are confronted with a joint optimization problem, the (possibly nonlinear) scaling functions $\alpha(\cdot)$ and $\beta(\cdot)$ are used to adjust the respective weights of classification rate and sensitivity in the optimization process.

The information loss in the CNN mapping $M$ is considerable: even if the original images are quantized with modest 8 bits/pixel, they undergo an information loss of 87.5% in the transformation to bipolar (1 bit/pixel) images. Since this reduction to black and white images facilitates the calculation of class prototypes and distance, the CNN is still useful even if the classification rate is not increased. Another advantage of bipolar CNN operations is, in view of analog VLSI CNN chips, that they can be rendered insensitive to small parameter variations.
5.4.3 The Genetic Approach

In contrast to the problems in the previous sections, the general classification problem does not provide any information on the CNN mapping $M$ except the vague preprocessing goal depicted in Fig. 5.13 and the mathematical transcription (5.24), which will serve as the fitness function (or objective function) of our genetic algorithm.

The second link between the algorithm and the concrete optimization problem is the coding of the parameters in the solution space, which, in our case, consists of a template set and an initial condition. To choose an appropriate coding strategy, we have to consider several constraints imposed by the analog VLSI implementation of the CNN universal chip. On the one hand, only a discrete set of values is programmable as template parameters, on the other hand, when realized as integrated circuits, the parameters are typically subject to perturbations of up to $8 - 10\%$ of their nominal values due to the inherent inaccuracy in the analog implementation. The first constraint is not a drawback but an advantage for a GA (and one of the reasons to use GA for this problem), since its search space is discrete anyway. We just have to ensure to code exactly those values that can be realized on a particular chip. The second constraint forbids the use of any templates with more than 11 non-zero parameters. Templates with higher connectivity are too sensitive to inaccuracies. From Chapter 2 we know that templates with higher connectivity cannot have a robustness of more than 8%. Hence, 8 parameters have to be fixed to zero. For symmetry reasons, the schemes

$$A = \begin{bmatrix} 0 & a_2 & 0 \\ a_4 & a_c & a_6 \\ 0 & a_8 & 0 \end{bmatrix}; \quad B = \begin{bmatrix} 0 & b_2 & 0 \\ b_4 & b_5 & b_6 \\ 0 & b_8 & 0 \end{bmatrix}; \quad I$$

and

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_c & 0 \\ 0 & 0 & 0 \end{bmatrix}; \quad B = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix}; \quad I$$

were used. Note that the second scheme is uncoupled, i.e., all non-center elements in the $A$-template are zero. Due to the restriction to bipolar operations, $a_c$ is not optimized but fixed to a value $> 1$. A chromosome is built by concatenating the coupled/uncoupled bit, 2 bits for the genotype coding the initial condition, and $m$ bits for each of the 10 parameters, leading to a bitstring of length $10m + 3$,
Codewords longer than 5 bits/parameter are generally not necessary due to the limited state swing of the chip and its discrete programmability. The sets of possible parameter values (phenotypes) for the template is determined by the properties of the CNN chip. We used

\[
\begin{align*}
\text{for } m = 3 : & & \pm \frac{1}{2} \{0, 1, 2, 4\}, \\
\text{for } m = 4 : & & \pm \frac{1}{4} \{0, 1, 2, 4, 6, 8, 10, 12\}, \\
\text{for } m = 5 : & & \pm \frac{1}{8} \{0, 1, 2, 3, 4, 6, 8, 10, 12, 16, 20, 24, 28, 32, 40, 48\}.
\end{align*}
\]

With this coding strategy, the GA will optimize both the template set and the initial condition, and the first bit determines whether the coupled or the uncoupled scheme is applied. The boundary condition of the CNN is fixed to \(-1\) for both input and state to achieve maximum template robustness. A tournament selection scheme was implemented. In so doing, only the ranking of the fitness values are relevant, and the selection operator is insensitive to any monotonic fitness scaling. This unique property permits the joint optimization for two different objectives, if one of the objectives is to be prioritized, which is true for the classification problem: correct classification is more important than robustness. Hence, by choosing \(\alpha(x) = x\) and \(\beta(x) = x/1000\), for example, the algorithm will first concentrate on the classification rate, and then, among all individuals with equal \(R\), select those with smaller sensitivity. With fitness-proportional selection, this would not be possible since all individuals with equal \(R\) would have an almost equal probability to be selected.

Note that it is by no means guaranteed that the template candidates are stable; unstable ones, however, receive a fitness value of zero and will therefore not survive the next selection operation.

Deriving benefit from the CNN sampling theorem 4.2, the CNN steady state outputs are calculated by a simple forward Euler algorithm with step size 1, which allows the treatment of fairly large problems on a workstation within reasonable time.
5.4.4 Results

We study the recognition of single digits '0' to '9' in a $10 \times 10$ image. The original black and white images (Fig. 5.15(a)) were subject to multiplicative noise of 30% and additive noise of 0.3 (both uniformly distributed), and finally each pixel was inverted with a probability of 20% ('spot noise'). For each digit, 10 such perturbed images were generated, which have to be classified according to (5.16). The class prototype and the image set for the digit 'one' are shown in Fig. 5.15(b) and (c), respectively. Direct classification on

(a) Original images (unknown to the classificator)

(b) Prototypes

(c) Perturbed image set for digit 'one'

(d) Output image set for digit 'one'

(e) Output prototypes

Figure 5.15: Image sets for the digit recognition task.
the gray-scale images yields a recognition rate of 98% with an average robustness of 57.5%. With the CNN preprocessing, the recognition rate can be increased to 100%, the robustness to 104%! For a population size of 100, a one-point crossover probability of 0.8, a mutation probability of 0.05, and a tournament selection scheme, the following uncoupled template was found within 35 generations:

\[
\alpha_c = 2; \quad B = \begin{bmatrix} -0.5 & -0.5 & 0 \\ 0 & -0.5 & 0.5 \\ 0 & 0 & 3 \end{bmatrix}; \quad I = -3
\]

The initial condition for this template is \( \mathbf{x}(0) = \mathbf{0} \). Fig. 5.15 (d) presents the CNN output images for the digit class 'one', (e) the output prototypes. Interestingly, these preprocessed prototypes are shifted diagonally left- and upwards (due to \( b_9 = 3 \) in the template) compared to the original prototypes in (b). The distance measure for this example was a quadratic (or Euclidean) distance, i.e., \( p = 2 \) in (5.17). In most runs of the GA, uncoupled templates are found to be optimum — coupled ones disappear gradually from the population owing to the selection pressure.

5.5 Summary

Different approaches for robust template design have been presented. In a two-stage genetic approach, first a population of correctly operating templates is generated (GA\(_1\)), which is then used as the initial population for the second step, the robustness optimization (GA\(_2\)). GAs with their discrete search space are very suitable for this optimization problem; as in most types of VLSI implementations, only a discrete set of template values is available for programming. However, the evaluation of the fitness functions in (GA\(_2\)) requires considerable computational effort. Therefore, a massively parallel supercomputer, the Intel Paragon, is used.

The steepest-ascent (hill climbing) method suffers from the same shortcoming. Nevertheless, it often finds globally optimal solutions, since the robustness landscape is not rugged. In the averaging approach, we make use of the fact that GAs inherently produce populations of solutions without additional effort. Their arithmetical average is expected to be robust. Therefore, no robustness evaluations have to be carried out here.

Finally, the advantages of all three methods are combined into a hybrid approach, which clearly outperforms the others. The average solution is a
perfect starting point for a steepest-ascent algorithm: the GA finds the hill, and the hill climber climbs it. This approach is not restricted to the optimization of CNN templates with respect to robustness, but is applicable in other optimization problems as well.

The _general classification problem_ provides a framework comprising virtually all image classification tasks that can be carried out on a CNN chip. Beside the maximization of the classification rate in a _test set_, it aims at increasing the _robustness_ of the classification in order to guarantee a correct classification of unknown vector sets. This two-objective optimization problem can be successfully attacked by genetic algorithms. GAs permit, on the one hand, an efficient search in a chip-specific template space, and, on the other hand, they are well-suited for such joint optimization problems if a tournament selection scheme is applied. For all concrete problems considered so far, the results are encouraging, in spite of the limited set of possible template values. Rather unexpectedly, uncoupled templates seem to perform better than coupled ones.
Chapter 6

Conclusions

The applicability of CNNs in the areas of image and signal processing only becomes efficient if an appropriate analog hardware implementing the underlying network is available. Beside significant advantages of CNN chips over digital hardware such as processing speed, small size, and power consumption there is one severe drawback, namely their relatively poor accuracy. Robust operation can only be guaranteed for locally regular tasks, which makes local regularity a key concept for any practical applications of CNN chips:

- Locally regular tasks can be carried out on virtually all classes of CNN chip prototypes implemented so far, whether they are implemented as continuous-time or as sampled-data systems, and whether a saturation or a high-gain nonlinearity is used. Locally irregular tasks, however, are inherently sensitive and therefore very unlikely to run reliably on an analog chip. Consequently, as long as the accuracy of the CNN circuitry is not substantially improved, the dynamics in the linear region cannot be exploited. From a practical point of view, it is sensible to design chips with the type of nonlinearity that is easiest to implement.

- Numerical simulations of locally regular tasks run extremely fast, as a simple forward Euler algorithm with a large step size yields exact equilibrium points in a number of integration steps that is in the order of the size of the CNN.
• For the class of locally regular templates, an exact and direct analytical design method exists, which permits, on the one hand, to determine the optimum template for a given CNN chip, and, on the other, provides insight into the dynamics of the CNN. Furthermore, analytical expressions can be derived for the settling time of these tasks. For the uncoupled class, they are exact, while for all other locally regular tasks, tight upper bounds can be calculated. This allows the optimization of templates with respect to processing speed, provided that the degree of robustness is still sufficient for a particular chip.

If a CNN optimization problem cannot be attacked by analytic tools, one has to resort to numerical optimization methods. Among these, stochastic optimization techniques have proven their usefulness in problems that are far more complex than template design for a given input-output mapping. On modern workstations, virtually any CNN-related optimization task can be solved efficiently.

Two ideas presented in this thesis may be helpful beyond the scope of CNNs.

Firstly, the method to solve a homogeneous and linear system of inequalities presented in Chapter 2 is not restricted to the CNN template design problem. It can readily be used to solve any such system in an optimum way, in the sense that all inequalities are satisfied with the largest possible safety margin.

Secondly, the CNN sampling theorem 4.2 basically includes all kinds of nonlinear first-order circuits. If the equilibrium points of such circuits are to be numerically determined, it is most efficient to apply forward Euler integration and to choose the step size to be equal to the internal time constant of the system.

We close by some suggestions for future research. The theory on robustness and settling time developed in this thesis is restricted to bipolar CNNs. Since one of the major advantages of analog processing techniques is their continuous range, gray scale image processing deserves serious consideration as well. The idea of robustness will have to be redefined, as strict output invariance is not achievable if the CNN system has equilibrium points in the linear region. Robustness theory turns into sensitivity theory, and the concept of settling time also has to be generalized.
As soon as more complex CNN cells become realizable, it is a worthwhile attempt to extend the template design theory further by including nonlinear and delay-type templates. Analytical approaches may fail, but stochastic optimization techniques can be expected to yield good results. Genetic algorithms and related methods may even be capable of creating CNN programs consisting of several CNN processing steps, which would be a important step towards an adaptive universal CNN processor.
Seite Leer / Blank leaf
Appendix A

CNN Task Library

In this appendix, we describe those CNN tasks that frequently appear in the examples of the thesis. For an exhaustive list of known CNN tasks we refer to [58].

In all tasks we consider bipolar (±1) inputs and outputs, and consider the actual “image” to be in black (+1) on a white (−1) “background”.

The abbreviations introduced below, e.g., “HLD”, may be applied to both the task itself or the network performing that task, i.e., horizontal line detection or horizontal line detector, respectively.

Some of the tasks may be realized by various CNNs, e.g., by means of coupled or uncoupled CNN, with 0 or −1 boundary values, or with different number of parameters. We, therefore, do not make any mention of possible realizations and leave the specifications of a realization to the corresponding sections.

These tasks can be explored by using the graphical simulator presented in Appendix B, which is accessible through the Internet.
**Shadowing (SH)**
The SH template projects the shadow of objects in the input image to the left when illuminated from the right side (Fig. A.1).

![Figure A.1: Shadowing.](image)

**Horizontal Line Detection (HLD)**
A horizontal line can be defined as the succession of at least two black pixels (in the horizontal direction). A HLD extracts these structures and turns any other configuration to background color. In terms of CNNs, a HLD can be described as a network that turns the output of any isolated (in the horizontal direction) black cell into white and leaves the outputs of the remaining cells unchanged (Fig. A.2).

![Figure A.2: Horizontal line detection.](image)

**Connected Component Detection (CCD)**
A connected component is an array of consecutive black cells. A CCD reduces connected components along the horizontal direction to a single pixel and shifts them toward the right with a one-pixel separation. The operation can also be interpreted as that of a counter which counts the number of contiguous blocks in the horizontal direction (Fig. A.3).
Figure A.3: Connected component detection.

Hole Filling (HF)
A HF fills the interior of all closed contours in the image. Fig. A.4 shows typical input and output of a HF.

Figure A.4: Hole filling.

Edge Extraction (EE)
This is the inverse operation of hole filling (Fig. A.5). Black cells whose upper, lower, right, and left neighbors are black turn white.

Figure A.5: Edge extraction.
Global Connectivity Detection (GCD)
A GCD determines whether a given geometric pattern is globally connected in one contiguous piece, or it is decomposed of two or more disconnected components. It deletes any connected objects that are "marked" in the binary image. An object is marked by changing at least one pixel from black to white in the initial state. The output contains the unmarked objects only (Fig. A.6).

![Figure A.6: Global connectivity detection.](image)
Appendix B

Simulation and Visualization of CNN Dynamics

B.1 Introduction

For teaching, research, and demonstration purposes, a graphical simulator for CNNs has been developed. In contrast to other simulators, the CNN cells are visualized in a grid structure, the values of input and states being represented by colors. Input and initial images can easily be generated and changed even while the integration of the system is in progress, and an oscilloscope function allows the quantitative study of CNN transients, thus providing insight into the dynamics of the network.

For those who are new to the world of CNNs, a series of predefined templates sets and demonstrations are available, which makes the simulator a valuable educational tool. Advanced users and CNN expert can examine manually-entered and parameterized templates and carry out experiments in a very broad spectrum of CNN theory and applications, including quantitative behavior, robustness aspects, settling time, state limitations, different output functions, and numerical integration methods.
Since the program is written in Java and accessible through World Wide Web on

http://www.isi.ee.ethz.ch/haenggi/CNN_web/CNNsim_adv.html,

there is no need to download and compile it explicitly, and it is easily and publicly accessible.

A short introduction to CNNs and a detailed manual about the simulator is available under


A detailed description of the features of the simulator is published in [39].

### B.2 Brief Characterization

The simulator basically integrates a CNN defined by (1.5). The cloning template is restricted to the nearest-neighbor case and assumed to be spatially invariant.

The graphical user interface (GUI, Fig. B.1) provides two $M \times N$ grids representing the input $u_{ij}$ on the left grid, and the state $x_{ij}(t)$ or the output $y_{ij}(t)$ on the other. White pixels represent the value of -1, black pixels +1. When the color of a cell turns red it means that its state $x_{ij}(t)$ is greater than 1, whereas green cells signify $x_{ij}(t) < -1$. For the values in between, a grayscale is applied. A useful feature of this simulator is that it permits toggling input and state values not only before starting the simulation, but even during the integration process which enlarges the scope of interesting investigations.

The simulator is based on all the features of an earlier version [59] and its operation is very similar. However, its functionality is greatly enhanced, which permits the user to carry out far more interesting experiments within a broad range of CNN theory and design.

In addition to the two grids, the user interface consists of buttons, numerical fields, and menu cards (choices). Temporarily inactive buttons and menus appear in light gray, and values in numerical fields may be mutated only when their background is white.

Most features are accessible through the main menu in the lower left corner of the GUI (Fig. B.1). When starting, the menu card Templates is open;
Figure B.1: The graphical user interface of the simulator.

by clicking on it, the menu cards Input&State, Size&Boundary, Perturbation&Robustness, Model, Trajectory Viewer, and Library are accessible.

To get acquainted with the simulator (and with CNNs, for those who are new to the world of CNNs), a series of demonstrations with predefined images and cloning templates has been implemented. In the same menu, a number of input images is available to facilitate the experiments.

In summary, the simulator provides a useful tool for the experienced researcher interested in investigating CNN dynamics, and for the CNN designer interested in such important features as robust template design and settling time. It also provides a useful learning tool for the newcomer to the theory and design of the CNN.
B.3 Technical Information

To run the simulator in a web browser the browser needs to understand Java version 1.1.5 or later. Browsers known to work with the CNN Simulator are

- The HotJava Browser.
  http://www.java.sun.com

- The Internet Explorer version 4.x for Windows95/98/NT and some Unix dialects.
  http://www.microsoft.com

- The Netscape Navigator version 4.06 or later.
  For Unix variants (including Linux):
  ftp://sunsite.cnlab-switch.ch/mirror/netscape/communicator/4.08/shipping/english/unix/
  For Windows95/98/NT:
  ftp://sunsite.cnlab-switch.ch/mirror/netscape/communicator/4.08/shipping/english/windows/windows95_or_nt/

For future versions of the communicator, please contact http://www.netscape.com.

In the Preferences or Options of the browser, its Java capabilities must be enabled before the simulator is activated. Note that depending on your operating system, the GUI of the simulator may look different from the examples in this paper.

The size of the simulator is 160KB. Depending on the speed of the internet connection, the download process will take between a few seconds and (at most) one or two minutes. No further transfer of program data is necessary as long as no other URL is visited. If the Disk Cache and/or the Memory Cache option is enabled in the browser, then a local copy of the simulator will be kept on disk or in memory.
List of Symbols

Roman Symbols

$A$ feedback template
$A$ feedback system matrix
$a_i$ feedback template parameter value $i$
$a_c$ self-feedback parameter (center element of the $A$ template)
$B$ control template
$B$ control system matrix
$b_i$ control template parameter value $i$
$b_c$ self-control parameter (center element of the $B$ template)
$\mathbb{B}$ set $\{-1, 1\}$
$\mathbb{B}_0$ set $\{-1, 0, 1\}$
$B(\cdot, \cdot)$ robustness of the classification in a classification problem
$\bar{B}(\cdot)$ average robustness of the classification in a classification problem
$C$ CNN cell capacitance
$C_U(\cdot)$ classification function with respect to $U$ in a classification problem
$c$ (i) half the width of the linear region in a $(\delta, c)$-CNN;
(ii) the number of classes in a classification problem
$\mathcal{C}$ (i) CNN cell; (ii) set of functions with continuous domain
$\mathcal{C}_i$ CNN cell $i$
$\mathcal{C}$ (i) set of cells that start within or enter the linear region at $t = 0$;
(ii) the search space of a genetic algorithm
cross crossover operator in a genetic algorithm
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D(\cdot)$</td>
<td>relative robustness of a template set</td>
</tr>
<tr>
<td>$\mathcal{D}$</td>
<td>set of functions with discrete domain</td>
</tr>
<tr>
<td>$f(\cdot)$</td>
<td>CNN output function</td>
</tr>
<tr>
<td>$f(\cdot)$</td>
<td>fitness function in a genetic algorithm</td>
</tr>
<tr>
<td>$f_c(\cdot)$</td>
<td>parameterized piecewise linear output function</td>
</tr>
<tr>
<td>$I$</td>
<td>spatially invariant bias</td>
</tr>
<tr>
<td>$K$</td>
<td>coefficient matrix comprising the output configuration of the cells in the neighborhood</td>
</tr>
<tr>
<td>$k_i^j$</td>
<td>$i$-th row vector within $K$</td>
</tr>
<tr>
<td>$L$</td>
<td>length of a propagation string</td>
</tr>
<tr>
<td>$L_P$</td>
<td>length of a longest propagation string</td>
</tr>
<tr>
<td>$L_P^*$</td>
<td>upper bound for $L_P$, maximized over all possible inputs and initial states</td>
</tr>
<tr>
<td>$M$</td>
<td>(i) number of rows of a planar CNN; (ii) CNN mapping $\mathbb{B}^n \times \mathbb{B}^n \rightarrow \mathbb{B}^n$</td>
</tr>
<tr>
<td>$m$</td>
<td>number of non-zero entries in a template set</td>
</tr>
<tr>
<td>$\hat{m}$</td>
<td>number of inequalities in a system inequalities that characterizes a CNN task (number of rows in $K$)</td>
</tr>
<tr>
<td>$\mathcal{M}(\cdot, \cdot, \cdot)$</td>
<td>set of vectors that have been processed by a CNN</td>
</tr>
<tr>
<td>mut</td>
<td>mutation operator in a genetic algorithm</td>
</tr>
<tr>
<td>$N$</td>
<td>number of columns of a planar CNN</td>
</tr>
<tr>
<td>$\mathcal{N}_T$</td>
<td>numerical integration operator</td>
</tr>
<tr>
<td>$\mathcal{N}_i^r$</td>
<td>neighborhood or radius $r$ of cell $i$</td>
</tr>
<tr>
<td>$p_m$</td>
<td>mutation rate in a genetic algorithm</td>
</tr>
<tr>
<td>$p_c$</td>
<td>crossover probability in a genetic algorithm</td>
</tr>
<tr>
<td>$p$</td>
<td>template vector containing all non-zero entries of a template set</td>
</tr>
<tr>
<td>$\tilde{p}$</td>
<td>template vector containing $a_c - 1$ (first element) and all other non-zero template parameters</td>
</tr>
<tr>
<td>$\mathcal{P}^N$</td>
<td>a population of size $N$ in a genetic algorithm</td>
</tr>
<tr>
<td>$\mathcal{P}^N$</td>
<td>set of all possible populations of size $N$ in a genetic algorithm</td>
</tr>
<tr>
<td>$q(\cdot)$</td>
<td>number of vectors in a class in a classification problem</td>
</tr>
<tr>
<td>$R$</td>
<td>CNN cell resistance</td>
</tr>
<tr>
<td>$\mathbb{R}$</td>
<td>set of real numbers</td>
</tr>
</tbody>
</table>
### List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_U(\cdot)$</td>
<td>classification rate with respect to $U$ in a classification problem</td>
</tr>
<tr>
<td>$\mathcal{R}$</td>
<td>open subset of $\mathbb{R}^m$ where a template operates correctly</td>
</tr>
<tr>
<td>$\mathcal{R}'$</td>
<td>closed subset of $\mathbb{R}^m$ where a template operates correctly</td>
</tr>
<tr>
<td>$S^T_\varepsilon$</td>
<td>relative sensitivity of the settling time to a template parameter $\varepsilon$</td>
</tr>
<tr>
<td>$\mathcal{S}$</td>
<td>set of slowest cells (cells that determine the settling time)</td>
</tr>
<tr>
<td>$\mathcal{S}_T$</td>
<td>sampling operator</td>
</tr>
<tr>
<td>\text{sat}(\cdot)</td>
<td>piecewise linear output function</td>
</tr>
<tr>
<td>\text{sel}</td>
<td>selection operator in a genetic algorithm</td>
</tr>
<tr>
<td>$t$</td>
<td>normalized time</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>averaged time between two zero-crossings of neighboring cells in a propagation string</td>
</tr>
<tr>
<td>$\hat{\Delta} t$</td>
<td>analytical estimate of $\Delta t$ such that $\hat{\Delta} t \gtrapprox \Delta t$</td>
</tr>
<tr>
<td>$T$</td>
<td>(i) the sampling period of a discrete-time CNN; (ii) step size of an Euler step when numerically integrating a CNN; (iii) temperature control parameter in the context of a simulated annealing algorithm</td>
</tr>
<tr>
<td>$\mathbb{T}$</td>
<td>set of all possible templates that are realizable on a given chip</td>
</tr>
<tr>
<td>$T_S$</td>
<td>settling time of a stable and bipolar CNN</td>
</tr>
<tr>
<td>$T_{S_i}$</td>
<td>settling time of cell $i$</td>
</tr>
<tr>
<td>$T^*_S$</td>
<td>worst case settling time (maximized over all possible inputs and initial states)</td>
</tr>
<tr>
<td>$\hat{T}_S$</td>
<td>analytical estimate of the settling time such that $\hat{T}_S \gtrapprox T_S$</td>
</tr>
<tr>
<td>$u$</td>
<td>CNN input vector</td>
</tr>
<tr>
<td>$u_i$</td>
<td>input of cell $i$</td>
</tr>
<tr>
<td>$u^j_i$</td>
<td>$i$-th vector of class $j$ in a classification problem</td>
</tr>
<tr>
<td>$\bar{u}^j$</td>
<td>prototype of class $j$ in a classification problem</td>
</tr>
<tr>
<td>$U$</td>
<td>set of vectors in a classification problem (training set), cf. $Z$</td>
</tr>
<tr>
<td>$w_i$</td>
<td>equals $B \ast u_i + I$, i.e., the time-independent part in the CNN equation</td>
</tr>
<tr>
<td>$x_i(\cdot)$</td>
<td>state of cell at position $i$</td>
</tr>
<tr>
<td>$x^*_i$</td>
<td>equilibrium state of cell $i$</td>
</tr>
<tr>
<td>$\mathcal{X}$</td>
<td>set of critical points in a $(\delta,c)$-CNN</td>
</tr>
<tr>
<td>$y_i(\cdot)$</td>
<td>output of cell $i$</td>
</tr>
<tr>
<td>$y^*_i$</td>
<td>equilibrium output of cell $i$</td>
</tr>
</tbody>
</table>
List of Symbols

\( y_d \) \hspace{1em} desired output vector
\( Z_i \) \hspace{1em} cardinality of the set of points of time for which \( x_i(t) = 0 \)
\( \mathbb{Z} \) \hspace{1em} set of integers
\( \mathcal{Z} \) \hspace{1em} set of vectors in a classification problem (test set), cf. \( \mathcal{U} \)

Greek Symbols

\( \beta \) \hspace{1em} chip-specific upper bound for \( ||p||_1 \)
\( \delta \) \hspace{1em} delta operator
\( \epsilon(\cdot) \) absolute robustness of a template set
\( \gamma(\cdot) \) safety margin of a template set
\( \mu \) \hspace{1em} multi-index
\( v \) \hspace{1em} multi-index
\( \Psi \) optimum template vector \( \tilde{p}_{opt} \) with a safety margin of 1 ("mother template")
\( \tau \) \hspace{1em} the \( RC \) product representing the cell's internal time constant
\( \mathcal{T} \) \hspace{1em} (i) template set, consisting of \( A, B, \) and \( I \);
\hspace{1em} (ii) template vector or CNN gene in the case of a planar CNN
\( \xi \) \hspace{1em} initial condition of a CNN processing step in a classification problem
\( \zeta_i \) \hspace{1em} time of the first zero-crossing of cell \( i \)

Other Symbols

\( \partial \) \hspace{1em} contribution from boundary cells
\( \partial_x \) \hspace{1em} contribution from boundary cells of the state
\( \partial_u \) \hspace{1em} contribution from boundary cells of the input
\( \mathbf{1} \) \hspace{1em} vector with all elements 1
\( \mathbf{1}^\pm \) \hspace{1em} vector with elements from \( \mathbb{B} \)
\( \| \cdot \|_1 \) \hspace{1em} \( L_1 \) vector norm (sum of absolute values)
\( \| \cdot \|_2 \) \hspace{1em} Euclidean norm
\( \circ \) \hspace{1em} (i) componentwise vector multiplication;
\hspace{1em} (ii) composition of operators
Bibliography


[40] Martin Hänggi and George S. Moschytz, "Stochastic and Hybrid Approaches Toward Robust Templates," in IEEE International Workshop


Curriculum Vitae

Personal Data

Name          Hänggi
First name    Martin
Date of birth May 11, 1969
Citizenship  Swiss

Education

1976–1982  Primary school in Niederglatt/ZH
1982–1988  Gymnasium in Büch, Matura Typus B
1989–1995  Study in Electrical Engineering at ETH Zürich
           1995  Dipl. El.-Ing. (M.Sc.) degree
                 Diploma thesis on “High-frequency Scanning Tunneling Microscopy”
1995–1998  Graduate Study in Information Technology, ETH Zürich
           1998  Post-diploma degree in Information Technology, ETH Zürich
1995–1996  Teaching assistant at Signal and Information Processing Laboratory
1997–1999  Research assistant at Signal and Information Processing Laboratory