LEARNING STATE SPACE TRAJECTORIES IN CELLULAR NEURAL NETWORKS

by

Andreas J. Schuler, Peter Nachbar, Josef A. Nossek, and Leon O. Chua

Memorandum No. UCB/ERL M92/106

29 September 1992

COVER PAGE

LEARNING STATE SPACE TRAJECTORIES IN CELLULAR NEURAL NETWORKS

by

Andreas J. Schuler, Peter Nachbar, Josef A. Nossek,
and Leon O. Chua

Memorandum No. UCB/ERL M92/106

29 September 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

This work is supported in part by the Office of Naval Research under grant N00014-89-J-1402 and the National Science Foundation under grant MIP-9001336.

LEARNING STATE SPACE TRAJECTORIES IN CELLULAR NEURAL NETWORKS

by

Andreas J. Schuler, Peter Nachbar, Josef A. Nossek,
and Leon O. Chua

Memorandum No. UCB/ERL M92/106

29 September 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

This work is supported in part by the Office of Naval Research under grant N00014-89-J-1402 and the National Science Foundation under grant MIP-9001336.

Learning State Space Trajectories in Cellular Neural Networks

Andreas J. Schuler, Peter Nachbar[*] and Josef A. Nossek
Institute for Network Theory and Circuit Design,
Technical University Munich, Germany

Leon O. Chua
Electronics Research Laboratory, College of Engineering
University of California, Berkeley, USA

September 29, 1992

Abstract

This paper presents a learning algorithm similar to the Backpropagation-Through-Time approach, which is well known for other dynamical neural networks. The algorithm is based on the minimization of an error criterion of the continuous dynamical system. The overall error is a product of a function of the state at a given time and the integral of an entire time function of the state over the trajectory prior to this time. The technique of variational calculus provides a way of calculcating the gradient of the error, which can be used to descend to a minimum on the error surface.

We will adopt this theory to CNNs and show its ability to work with piecewise linear systems as well. The described algorithm learns the parameter vector of a locally connected CNN with translationally invariant weights, where the error functional is a minimum over the parameter space. With this algorithm not only the stable and unstable patterns, but whole trajectories of the dynamical system in the state space can be learned.

Some examples show that the algorithm works, but is restricted to small problems due to computational complexity.

1 Introduction

Neural networks have become useful because of their ability to perform a wide variety of useful tasks by choosing a suitable set of weights. There are many different ways of adjusting the weights such that the neural network will perform a specified task.

For dynamic recurrent neural networks Hebbian learning was traditionally used until Almeida [2] and Pineda [11, 12, 13] generalized the feedforward backpropagation algorithm [14] to recurrent neural networks. Pearlmutter [10] described a procedure to design not only the fixed points, but the trajectory of recurrent dynamic neural networks.

We concentrate on a subclass of neural networks: *Cellular Neural Networks* (CNNs) [3, 9], where the interconnections between the neurons or cells are translationally invariant and only local. Therefore, the number of weights, which have to be learned or designed, is very small. One first step towards the systematic design of CNNs has been proposed by Zou et. al. [17], and later by Slot and Kacprzak [16].

[*]Supported by a grant from the Ernst von Siemens foundation.

Chua and Thiran [4] provide a method for synthesizing CNNs for simple applications, and Seiler, et. al. [15] show how to systematically design a CNN with stable and unstable outputs while simultaneously maximizing its robustness.

Except for [4], which is restricted to relatively simple problems, the correct operation of the network is not guaranteed, but only the stable or unstable points of the dynamical system are designed correctly.

We adopt the method of learning fixed points, namely *recurrent backpropagation*, and non-fixed points, namely *backpropagation-through-time*, to the problem of learning state space trajectories in CNNs. After a short review of CNNs in Section 2, we will discuss the purpose of learning in Section 3. In Section 4, the calculus-of-variations technique will be introduced for arbitrary dynamical systems in very similar way as given by Miesbach in [8]. This technique will be applied to CNNs in Section 5. Examples of CNN learning to perform different tasks are given in Section 6.

2 The Architecture of CNNs

Cellular Neural Networks or *CNNs* were introduced by Chua and Yang [3], where the definition was given for regular and locally connected structures. Nossek, Seiler, Roska, and Chua [9] later presented a description with abstract language and arbitrary topologies. For our purpose a description between them is best suited in abstract language and restricted to regular and only locally connected structures, as follows:

Definition 2.1 Strict Cellular Neural Network

If the *feedback operator* **a** and the *control operator* **b** are translationally invariant and only local, then the state equation of a *Strict Cellular Neural Network* can be written as

$$\dot{x}_{c,d} = F_{c,d} = \frac{1}{\tau_x}\left(-x_{c,d} + \sum_{\nu=-r}^{+r}\sum_{\mu=-r}^{+r}\left[a(\nu,\mu)y_{c+\nu,d+\mu} + b(\nu,\mu)u_{c+\nu,d+\mu}\right] + i\right) . \tag{1}$$

The summation is restricted to the rectangular cell grid CG, which is a two-dimensional grid of cells defined by $1 \le c \le N$ and $1 \le d \le M$. Its *output equations* read

$$y_{c,d}(t) = f(x_{c,d}(t)) = \frac{1}{2}\left(|1 + x_{c,d}(t)| - |1 - x_{c,d}(t)|\right) . \tag{2}$$

The interconnections are local, i.e. only cells inside a neighbourhood r are connected by weights.

Usually only the *state* and *output* values $x_{c,d}$ and $y_{c,d}$ are time-dependent in a CNN, whereas the *feedback and control coefficients* $a(\nu,\mu)$ and $b(\nu,\mu)$ as well as the *offset* i are constant. The *inputs* $u_{c,d}$ are also usually constant. τ_x is the time constant of the dynamics.

Furthermore, the initial state values $x_{c,d}(0)$ and the input values $u_{c,d}$ are bounded by:

$$|x_{c,d}(0)| \le 1 , \tag{3}$$
$$|u_{c,d}(t)| \le 1 , \quad \forall t \ge 0 . \tag{4}$$

A system of the form (1), (2), which additionally satisfies (3) is called a *Strict Cellular Neural Network* or *Strict CNN*.

Definition 2.2 Two-dimensional Convolution Operator

The feedback operator **a**, which defines the dynamical behaviour of the CNN, can be arranged as a vector in $\mathbb{R}^{(2r+1)(2r+1)}$ and contains the translationally invariant weights between cells in a r-neighborhood. The *two-dimensional convolution operator* $\star$ defines the weighted sum over the cells inside this neighborhood.

$$\mathbf{a} \star y_{c,d} = \sum_{\nu=-r}^{+r} \sum_{\mu=-r}^{+r} a(\nu, \mu) y_{c+\nu,d+\mu} \,. \tag{5}$$

□

A corresponding equation holds for the control operator **b**.

Definition 2.3 Parameter Space [15]

The *parameter space* $\mathcal{P}$ of a CNN is $\mathbb{R}^m$, where

$$m = 2(2r+1)^2 + 2. \tag{6}$$

and r is the neighbourhood. □

The *parameter vector* $p \in \mathcal{P}$ of a particular CNN can be written as follows:

$$\mathbf{p} = (\mathbf{a}^{\mathsf{T}}, \mathbf{b}^{\mathsf{T}}, i, \tau_x)^{\mathsf{T}} \,. \tag{7}$$

□

With the use of the two-dimensional convolution operator, the description of a CNN can be simplified, so that:

$$\tau_x \dot{x}_{c,d} = -x_{c,d} + \mathbf{a} * y_{c,d} + \mathbf{b} * u_{c,d} + i \,. \tag{8}$$

The state, the output and the input of the CNN can be arranged to a vector in $\mathbb{R}^{NM}$ each and therefore the CNN can be written in the standard form of an ordinary dynamical system

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}, t, \mathbf{p}) \,, \tag{9}$$

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \,. \tag{10}$$

3 Learning as a Two Point Boundary Value Problem (TPBVP)

Learning for CNNs means to determine the network parameter **p** such that the system will perform a specified task. The desired task of a CNN can be described by a set of L triple of patterns $(\mathbf{x}_0^\nu, \mathbf{u}^\nu, \mathbf{y}_*^\nu)$, where $\mathbf{x}_0^\nu$ is the initial state, $\mathbf{u}^\nu$ the input and $\mathbf{y}_*^\nu$ the desired output of the CNN.

The CNN has to be trained to perform the mapping

$$(\mathbf{x}_0^\nu, \mathbf{u}^\nu) \mapsto \mathbf{y}_*^\nu\,, \qquad \nu = 1\,,\ldots,L\,. \tag{11}$$

The information can be fed into the input as well as into the initial states, as usually done in an associative recall memory. This is often called an attractor mechanism [6]: the state converges to this stable fixed points, which provides the desired output.

The goal of learning is to find a parameter vector $\mathbf{p}$, such that this mapping (11) will be performed, i.e.: for each of the given pair of patterns for the input and the initial state, the output of the CNN converges to the desired output:

$$\mathbf{y}(t \to T) = \mathbf{y}_*^\nu\,, \qquad \nu = 1\,,\ldots,L\,. \tag{12}$$

As the transfer function of a CNN is piecewise linear, the output can be specified to converge in finite time T.

The differential equations (1) are required to satisfy boundary conditions at the starting and the ending values, therefore the resulting problem is called *Two Point Boundary Value Problem* (TPBVP) [1]. The boundary condition of the state has the form of a set of inequalities, as for the value at the time T only the output of the CNN is prescribed,

This in general is very hard to solve, because of the high dimension of the problem. There are LNM differential equations with two boundary conditions each, usually at time $t = 0$ and $t = T$. Therefore shooting methods or relaxation methods require immense computational effort to solve this problem.

The next section will show an alternativ approach for solving this problem.

4 The Backpropagation-through-Time Algorithm

We will introduce a scalar error functional to measure the system performance. The problem is then to find the parameter, where the error functional takes its lowest value. Therefore we obtain a nonlinear minimization problem, for which at least a local minimum can be found using gradient techniques. Because of the piecewise linear transfer function of the CNNs the error functionals of Almeida [2] and Pineda [11, 12, 13], which are backpropagation algorithms generalized for recurrent neural networks, can not be used.

Based on the idea of Pearlmutter [10] to design the trajectory of recurrent dynamic neural networks with the backpropagation-through-time algorithm, we will use the trajectory of the CNN state to gain information about the gradient of a modified error functional.

The *backpropagation-through-time algorithm* (BTT) can be derived from solving a classical variation problem [8].

Examine the dynamical system, which is given by a system of differential equations

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, t, \mathbf{p})\,, \tag{13}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the system state. In the case of CNNs: $n = NM$.

The performance of the system is assumed to be measured by a scalar functional E which bookkeeps the errors over the whole transient, i.e. over the finite interval $[0, T]$, where T is the given time:

$$E = \mathbf{L}_1^{\mathsf{T}}(\mathbf{x}(T)) \int_0^T \mathbf{L}_2(\mathbf{x}, \dot{\mathbf{x}}, t) dt . \tag{14}$$

$\mathbf{L}_1 : \mathbb{R}^n \to \mathbb{R}^n$ measures the error at the given time T, whereas $\mathbf{L}_2 : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is integrated over the given interval, to take into account the effect of a change of the parameters on the trajectory. This can be used to learn a specific trajectory as well.

The existence of the definite integrals defining the cost or error functional is assumed, and the minimizing functions are to be chosen from the set of all functions having continuous second derivatives over the time interval.

The problem is to adjust the parameter in order to find a function $\mathbf{x}(t)$ such that the error functional E given by (14) is minimal. Minimize the error functional (14) under the equality constraint given by the differential equations (13), which can be written as

$$\mathbf{F}(\mathbf{x}, t, \mathbf{p}) - \dot{\mathbf{x}} = 0 . \tag{15}$$

It can be shown that the solution to the constrained problem can be obtained by minimizing the modified error functional, which includes the contraints

$$E' = \int_0^T \left[\mathbf{L}_1^{\mathsf{T}}(\mathbf{x}(T)) \mathbf{L}_2(\mathbf{x}, \dot{\mathbf{x}}, t) + \lambda^{\mathsf{T}} (\mathbf{F}(\mathbf{x}, t, \mathbf{p}) - \dot{\mathbf{x}}) \right] dt , \tag{16}$$

where $\lambda \in \mathbb{R}^n$ is the vector equivalent of the Lagrange multiplier.

4.1 The Calculus-of-Variations Approach

A necessary condition for an extremum of E' is that the first variation of E' (the Frechet-derivative) $\delta E'$ be zero (independent of the variations $\delta\mathbf{x}$, $\delta\dot{\mathbf{x}}$ and $\delta\lambda$):[1]

$$\delta E' = \int_0^T \left[\mathbf{L}_2^{\mathsf{T}} \frac{\partial \mathbf{L}_1}{\partial \mathbf{x}} \delta\mathbf{x} \Big|_T + \mathbf{L}_1^{\mathsf{T}} \left(\frac{\partial \mathbf{L}_2}{\partial \mathbf{x}} \delta\mathbf{x} + \frac{\partial \mathbf{L}_2}{\partial \dot{\mathbf{x}}} \delta\dot{\mathbf{x}} \right) + \lambda^{\mathsf{T}} \left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}} \delta\mathbf{x} - \delta\dot{\mathbf{x}} \right) + (\mathbf{F}(\mathbf{x}, t, \mathbf{p}) - \dot{\mathbf{x}})^{\mathsf{T}} \delta\lambda \right] dt. \tag{18}$$

Since the variation $\delta E'$ must be zero independent of $\delta\lambda$, which can be chosen independent of the other terms, the factor of $\delta\lambda$ must be zero. This guarantees, that the contraints are fulfilled.

[1]The partial derivative of a scalar function $f(\mathbf{x})$ with respect to a vector $\mathbf{x}$ is defined to be a row vector:

$$\frac{\partial f}{\partial \mathbf{x}} = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right) . \tag{17}$$

After rearranging and integrating by parts the terms containing $\delta\dot{\mathbf{x}}$, the variation of E' becomes:

$$\begin{aligned}\delta E' = \quad & \int_0^T \mathbf{L}_2^{\mathsf{T}} dt \frac{\partial \mathbf{L}_1}{\partial \mathbf{x}} \delta\mathbf{x}\bigg|_T + \left(\mathbf{L}_1^{\mathsf{T}}(\mathbf{x}(T)) \frac{\partial \mathbf{L}_2}{\partial \dot{\mathbf{x}}} - \lambda^{\mathsf{T}}\right) \delta\mathbf{x}\bigg|_0^T + \\ & + \int_0^T \left[\left(\mathbf{L}_1^{\mathsf{T}}(\mathbf{x}(T)) \frac{\partial \mathbf{L}_2}{\partial \mathbf{x}} + \lambda^{\mathsf{T}} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} - \mathbf{L}_1^{\mathsf{T}}(\mathbf{x}(T)) \frac{d}{dt}\frac{\partial \mathbf{L}_2}{\partial \dot{\mathbf{x}}} + \dot{\lambda}^{\mathsf{T}}\right) \delta\mathbf{x}\right] dt . \end{aligned} \tag{19}$$

Since $\delta E'$ must equal zero independent of the first variation $\delta\mathbf{x}$ of $\mathbf{x}$, the second line of (19) yields the *Euler-Lagrange equation:*

$$\dot{\lambda} + \left(\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right)^{\mathsf{T}} \lambda + \left(\frac{\partial \mathbf{L}_2}{\partial \mathbf{x}} - \frac{d}{dt}\frac{\partial \mathbf{L}_2}{\partial \dot{\mathbf{x}}}\right)^{\mathsf{T}} \mathbf{L}_1(\mathbf{x}(T)) = 0 . \tag{20}$$

The initial state $\mathbf{x}(0) = \mathbf{x}_0$ is fixed, thus the variation of $\mathbf{x}$ is zero for $t = 0$:

$$\delta\mathbf{x}(0) = 0 \tag{21}$$

The variation $\delta\mathbf{x}|_T$ is arbitrary, and the first two terms of (19) can be combined to get a condition for the Lagrange multipliers λ at the time T. This condition is called the *associated transversality condition*:

$$\lambda(T) = \left(\frac{\partial \mathbf{L}_1(\mathbf{x}(T))}{\partial \mathbf{x}}\right)^{\mathsf{T}} \int_0^T \mathbf{L}_2 dt + \left(\frac{\partial \mathbf{L}_2}{\partial \dot{\mathbf{x}}}\right)^{\mathsf{T}} \mathbf{L}_1(\mathbf{x}(T))\bigg|_{t=T} . \tag{22}$$

This is once again a two point boundary value problem (TPBVP) of $\mathbf{x}$ and λ, which in the general case cannot be solved analytically.

4.2 Calculating the Gradients of the Error

The objective is to minimize E with respect to the parameter vector $\mathbf{p}$. This can be done by using efficient gradient techniques, where the gradient[2] $\frac{\partial E}{\partial \mathbf{p}}$ can be obtained by the formal differentiation of (14) with respect to $\mathbf{p}$:

$$\frac{\partial E}{\partial \mathbf{p}} = \int_0^T \mathbf{L}_2^{\mathsf{T}} dt \, \frac{\partial \mathbf{L}_1}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\bigg|_T + \mathbf{L}_1^{\mathsf{T}} \int_0^T \left(\frac{\partial \mathbf{L}_2}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}} + \frac{\partial \mathbf{L}_2}{\partial \dot{\mathbf{x}}} \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{p}}\right) dt . \tag{23}$$

This can be simplified by integrating the second term of the integral in (23) by parts

$$\frac{\partial E}{\partial \mathbf{p}} = \int_0^T \mathbf{L}_2^{\mathsf{T}} dt \, \frac{\partial \mathbf{L}_1}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\bigg|_T + \mathbf{L}_1^{\mathsf{T}} \int_0^T \left(\frac{\partial \mathbf{L}_2}{\partial \mathbf{x}} - \frac{d}{dt}\frac{\partial \mathbf{L}_2}{\partial \dot{\mathbf{x}}}\right) \frac{\partial \mathbf{x}}{\partial \mathbf{p}} dt + \mathbf{L}_1^{\mathsf{T}} \frac{\partial \mathbf{L}_2}{\partial \dot{\mathbf{x}}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\bigg|_0^T \tag{24}$$

[2]This is again a row vector corresponding to the idea of 1-forms in differential geometry.

and from the definition of λ from the Euler-Lagrange equation (20), the parenthesized term in the integral can be substituted

$$\frac{\partial E}{\partial \mathbf{p}} = \int_0^T \mathbf{L}_2^{\mathsf{T}} dt \left.\frac{\partial \mathbf{L_1}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right|_T + \mathbf{L}_1^{\mathsf{T}} \left.\frac{\partial \mathbf{L_2}}{\partial \dot{\mathbf{x}}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right|_0^T + \int_0^T \left(-\lambda^{\mathsf{T}} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} - \dot{\lambda}^{\mathsf{T}}\right) \frac{\partial \mathbf{x}}{\partial \mathbf{p}} dt \,. \tag{25}$$

Using integration by parts again we obtain

$$\frac{\partial E}{\partial \mathbf{p}} = \int_0^T \mathbf{L_2}^{\mathsf{T}} dt \left.\frac{\partial \mathbf{L_1}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right|_T + \left.\left(\mathbf{L}_1^{\mathsf{T}} \frac{\partial \mathbf{L_2}}{\partial \dot{\mathbf{x}}} - \lambda^{\mathsf{T}}\right) \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right|_0^T + \int_0^T \lambda^{\mathsf{T}} \left(\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{p}} - \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right) dt \,. \tag{26}$$

The differentiation of (13) with respect to **p** gives:

$$\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{p}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{p}} + \frac{\partial \mathbf{F}}{\partial \mathbf{p}} \,. \tag{27}$$

Since the initial state does not depend on the parameters

$$\left.\frac{\partial \mathbf{x}}{\partial \mathbf{p}}\right|_{t=0} = 0 \,, \tag{28}$$

the first two terms of (26) comprise the transversality condition, and thus are zero. This finally leads together with (27) to a simplified equation for the gradient of E with respect to **p**:

$$\frac{\partial E}{\partial \mathbf{p}} = \int_0^T \lambda^{\mathsf{T}} \left(\frac{\partial \mathbf{F}}{\partial \mathbf{p}}\right) dt \,. \tag{29}$$

A necessary condition for an extremum of E is

$$\frac{\partial E}{\partial \mathbf{p}} = 0 \,. \tag{30}$$

4.3 Minimizing the Error Functional using the Gradients

Using the gradients of the error, minima of the error functional can be found by using various gradient techniques like conjugate gradient methods or Newton-like methods. Of course, this algorithm does not always find the global minimum of the error functional, but descends only to a local minimum. The algorithm stops if the gradient gets zero, whether it has found a local minimum or the global one.

The algorithm works as follows:

```
begin  Minimizing the Error Functional $E$
  start with a random parameter vector $\mathbf{p}$
  repeat
    integrate $\dot{\mathbf{x}}$ from $t = 0$ to $T$
    integrate $\dot{\lambda}$ from $t = T$ to 0 and simultaneously evaluate the gradient
    perform a gradient step to reduce the value of the error
  until gradient of $E$ becomes zero
end.
```

5 The BTT-Algorithm for CNNs

In this section the backpropagation-through-time algorithm will be applied to CNNs. Eqn. (1) and (2) have to be used for the dynamical system (13) and an appropriate measure of the error has to be found.

The partial derivative of the state equations with respect to the state variables, which describes the influence of the change of one state $x_{c,d}$ to the time derivative $\dot{x}_{c+\nu,d+\mu}$ of a cell $c+\nu, d+\mu$, can be computed from the right hand side of (1):

$$\frac{\partial F_{c+\nu,d+\mu}}{\partial x_{c,d}} = \frac{1}{\tau_x}\begin{cases} 0\,, & \text{if } |\nu| > r \text{ or } |\mu| > r\,;\\ -1 + a(0,0)f'(x_{c,d})\,, & \text{if } \nu = 0 \text{ and } \mu = 0\,;\\ a(-\nu,-\mu)f'(x_{c,d})\,, & \text{else}\,. \end{cases} \tag{31}$$

The partial differentiation of (1) with respect to the parameters yields:

$$\frac{\partial F_{c,d}}{\partial a(\nu,\mu)} = \frac{1}{\tau_x}f(x_{c+\nu,d+\mu})\,,\quad \frac{\partial F_{c,d}}{\partial b(\nu,\mu)} = \frac{1}{\tau_x}u_{c+\nu,d+\mu}\,,\quad \frac{\partial F_{c,d}}{\partial i} = \frac{1}{\tau_x}\,.\quad \frac{\partial F_{c,d}}{\partial \tau_x} = -\frac{1}{\tau_x}\dot{x}_{c,d}\,. \tag{32}$$

The gradient of the error can now be written with the use of λ, which has the same dimension as $\mathbf{x}$:

$$\frac{\partial E}{\partial a(\nu,\mu)} = \frac{1}{\tau_x}\int_0^T \sum_{c=1}^{N}\sum_{d=1}^{M} f(x_{c+\nu,d+\mu})\lambda_{c,d}\,dt\,, \tag{33}$$

$$\frac{\partial E}{\partial b(\nu,\mu)} = \frac{1}{\tau_x}\int_0^T \sum_{c=1}^{N}\sum_{d=1}^{M} u_{c+\nu,d+\mu}\lambda_{c,d}\,dt\,, \tag{34}$$

$$\frac{\partial E}{\partial i} = \frac{1}{\tau_x}\int_0^T \sum_{c=1}^{N}\sum_{d=1}^{M} \lambda_{c,d}\,dt\,. \tag{35}$$

$$\frac{\partial E}{\partial \tau_x} = -\frac{1}{\tau_x}\int_0^T \sum_{c=1}^{N}\sum_{d=1}^{M} \dot{x}_{c,d}\lambda_{c,d}\,dt\,. \tag{36}$$

The equations so far were independent of the choice of an error functional, but of course, the equations for λ, which are the Euler-Lagrange-Equations (20) and the associated transversality conditions (22), are not.

5.1 Selection of an Error Functional

We will now show the error functional, which we used for the learning in CNNs. The component c, d of the vector valued errors depend only on the corresponding state $x_{c,d}$ and measure the distance of a function of the state to the desired output $y^*_{c,d}$ using a given norm $2k$, where $k \in \mathbb{N}$:

$$\begin{aligned} L1_{c,d}(\mathrm{x}(T)) &= \frac{1}{2k}\left(g(x_{c,d}(T)) - y^*_{c,d}\right)^{2k} = \left\|g(x_{c,d}(T)) - y^*_{c,d}\right\|_{2k}^{2k}, \\ L2_{c,d}(\mathrm{x}, \dot{\mathrm{x}}, t) &= \frac{1}{2k}\left(g(x_{c,d}(t)) - y^*_{c,d}\right)^{2k} = \left\|g(x_{c,d}(t)) - y^*_{c,d}\right\|_{2k}^{2k}. \end{aligned} \tag{37}$$

To assure that the state is not trained to be exactly on the boundary of the saturation regions, we use not the transfer function $f(x_{c,d})$ directly for $g(x_{c,d})$, but one of the family with a parameter ρ:

$$g_\rho(x_{c,d}) = f\left(\frac{x_{c,d}}{1+\rho}\right), \qquad \rho \geq 0. \tag{38}$$

The boundaries of the saturation regions of $g_\rho(x_{c,d})$ are at an absolute value of $|x_{c,d}| = 1 + \rho$. Consequently the functions $L1_{c,d}$ and $L2_{c,d}$ get zero, if the state $x_{c,d}$ is in the saturation where the output of this cell is equal to the desired output, and has at least a distance of ρ to the boundary of the saturation in the transfer function.

If $\rho = 0$, the function $g_\rho(x_{c,d})$ clearly is equal to the transfer function of the CNN, $g_0(x_{c,d}) = f(x_{c,d})$, and therefore the error functions $L1_{c,d}$ and $L2_{c,d}$ measure the distance of the actual output to the desired one.

With the definition of $L1_{c,d}$ and $L2_{c,d}$ (37) the error functional, which will be minimized, becomes:

$$E = \sum_{c=1}^{N}\sum_{d=1}^{M}\left[\frac{1}{2k}\left(g(x_{c,d}(T)) - y^*_{c,d}\right)^{2k} \int_0^T \frac{1}{2k}\left(g(x_{c,d}(t)) - y^*_{c,d}\right)^{2k} dt\right]. \tag{39}$$

Remarks: *Properties of this Error Functional*

- The error (39) is non-negativ, as it is a sum of a product of non-negative terms.
- The integrated error of a cell c, d does not influence the error E, if $L2_{c,d}(x_{c,d}(T))$ is zero.
- The error functional (39) gets zero with the use of (38), if and only if the output of all cells are in the desired saturation and the state variables have at least a distance of ρ from the boundaries of the transfer function.

Minimizing this functional, where

$$\frac{\partial E}{\partial a(\nu,\mu)} = 0, \qquad \frac{\partial E}{\partial b(\nu,\mu)} = 0, \qquad \frac{\partial E}{\partial i} = 0, \qquad \frac{\partial E}{\partial \tau_x} = 0, \tag{40}$$

yields the Euler-Lagrange equation for the CNN:

$$\begin{aligned}\dot{\lambda}_{c,d} &= \frac{1}{\tau_x}\left(\lambda_{c,d} - \sum_{\nu=-r}^{+r}\sum_{\mu=-r}^{+r}\left[a(-\nu,-\mu)f'(x_{c,d})\lambda_{c+\nu,d+\mu}\right]\right) - \\ &\quad - \left(g(x_{c,d}(t)) - y^*_{c,d}\right)^{2k-1} g'(x_{c,d}(t))\frac{1}{2k}\left(g(x_{c,d}(T)) - y^*_{c,d}\right)^{2k}, \end{aligned} \qquad (41)$$

and the associated transversality condition:

$$\lambda_{c,d}(T) = \left(g(x_{c,d}(T)) - y^*_{c,d}\right)^{2k-1} g'(x_{c,d}(T)) \int_0^T \frac{1}{2k}\left(g(x_{c,d}(t)) - y^*_{c,d}\right)^{2k} dt\,. \qquad (42)$$

6 Examples

In this section some examples of learning state space trajectories in CNNs are given, most of them requiring only the knowledge of the desired output. The whole transient must not be given by a teacher, but it is used for learning, as the error may be integrated over the whole transient.

The descent in the parameter space to minimize the error functional is done with the conjugate gradient minimization method of Fletcher and Reeves [5], but a simpler method such as steepest descent will work in general, requiring more iteration steps.

A measure of the computational effort of the minimization is the number of function evaluations, that is running of the CNN forward and integrating the error over the transient, and the number of gradient evaluations, that is running the CNN backward in time simultaneously evaluating the gradients.

The last example will show the use of the learning algorithm not only to learn fixed points, but as well learning the complete transient of a simple oscillatory two-cell CNN, which was designed robustly [15].

We start the procedure of learning with a initial parameter vector, where all components are zero except the time constant $\tau_x = 1$. Thus all cells are decoupled from the input as well from each other, and the state of each cell exponentially tends to zero as time evolves to infinity. In some examples we have not trained the time constant τ_x of the CNN, although it is useful especially for learning the oscillations in Section 6.3. In most cases we used the transfer function for the error directly, and the squared error, i.e. $k = 1$.

6.1 State-based Logical OR

The design of a simple two-cell CNN which produces an output of both cells which is equal the logical "OR" of the initial state, where $+1$ is interpreted as 'true' and -1 as 'false' was reported in [15]. As there is no input of the CNN the operation is called state-based. The four different combinations of logical values were used as the learning samples. After three gradient steps with a total number of seven forward and backward integrations the error was zero. The resulting template values (Fig. 1) and the phase-plane trajectory of the four different initial conditions of this two-cell CNN are shown together with the boundary of the basins of attraction in Fig. 2. Although the self-feedback is slightly smaller than one, this template obtains a stable CNN. The symmetry of the templates is obtained automatically because of the symmetry of the problem.

$a =$	0.716	0.964	0.716
$b =$	0.000	0.000	0.000
$i =$	0.033		
$\tau_x =$	1.000	(fixed)	

Fig. 1: The trained template values of the State-based logical OR CNN.

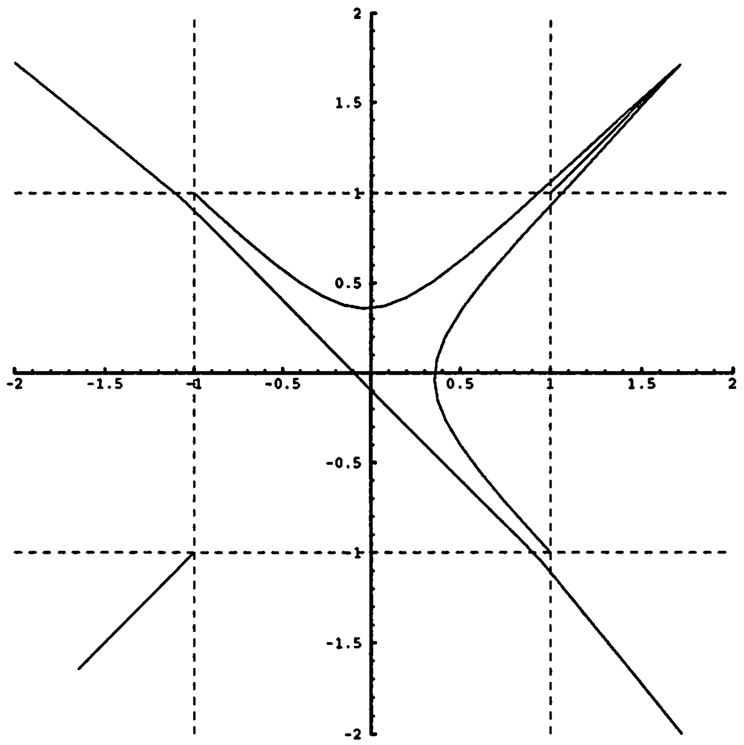

Fig. 2: The phase-plane trajectories together with the boundary of the basins of attraction.

6.2 Connected-Component-Detector

To learn the task of "Connected-Component-Detection" [7] we used all possible combinations of black and white cells in a linear CNN with five cells, which were the 32 learning samples. After only 13 gradient descents with 47 forward and backward integrations the error and the gradient were zero. In Fig. 3 the learned template values for this problem are given.

$a =$	0.430	1.509	−0.430
$b =$	0.000	0.000	0.000
$i =$	−0.003		
$\tau_x =$	1.000	(fixed)	

Fig. 3: The resulting template values of the Connected Component Detector.

6.3 Two-Cell Oscillator

The last examples will show that this algorithm can be used to train one cell of a simple two-cell CNN to follow a given trajectory. Depending on the desired waveform of the signal and its amplitude we will use the state or the output of one cell as the signal. For this examples, the functions $\mathbf{L}_1$ and $\mathbf{L}_2$ were chosen differently to measure only the error of the state of the output of one cell.

Sinusoidal Oscillator: As a first example we trained the trajectory of cell 1 to be a sinusoidal function of the time with an amplitude of 1 and a period time of 5: $x_1^*(t) = \sin(2\pi t/5)$. Since we wanted to learn the state we used the following error functional:

$$E = \frac{1}{2} \int_0^T (x_1(t) - x_1^*(t))^2 dt \,. \tag{43}$$

Fig. 4 shows the trained trajectory of cell 1, starting from the initial state $\mathbf{x} = (0, 1)^{\mathsf{T}}$. The error can be reduced further with an increased value of the time T. The corresponding values of this CNN are shown in Fig. 4.

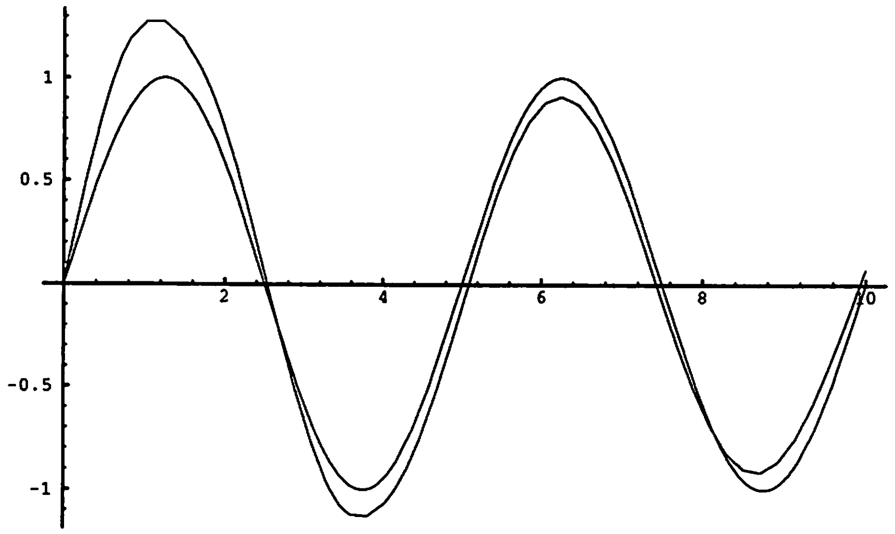

Fig. 4: The trajectory of cell 1 of a sinusoidal oscillating two-cell CNN.

$a =$	-0.875	0.961	1.931
$b =$	0.000	0.000	0.000
$i =$	0.000		
$\tau_x =$	1.000	(fixed)	

Fig. 5: The template values of a sinusoidal oscillating two-cell CNN.

In the following examples we always trained the output of one cell to follow a desired trajectory. Therefore the error functional was:

$$E = \frac{1}{2}\int_0^T (f(x_1(t)) - y_1^*(t))^2 dt. \tag{44}$$

Trapezoidal Oscillator: The desired trajectory of the output of cell 1 was a trapezoidal function of the time with an amplitude of 1, a period of 5 units of time and an absolute value of the slopes of 2. The saturation of the piecewise linear transfer function is used to get the saturated parts of the trapezoidal function. Fig. 6 shows the trained output of cell 1, starting from the initial state $\mathbf{x} = (0, 0.7)^T$ and Fig. 7 the corresponding values of the template.

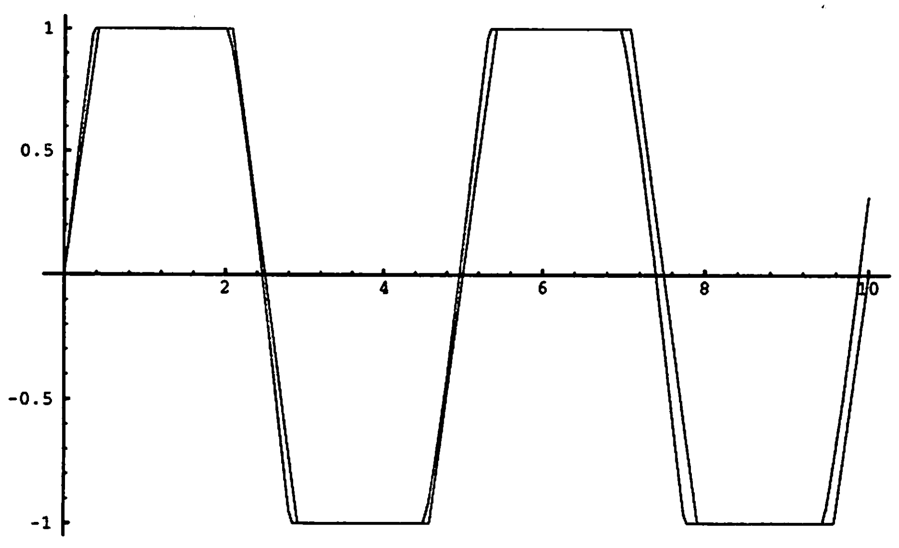

Fig. 6: The output of cell 1 of the trapezoidal two-cell oscillator.

$a =$	−0.672	1.197	4.000
$b =$	0.000	0.000	0.000
$i =$	0.006		
$\tau_x =$	1.000	(fixed)	

Fig. 7: The template values of the CNN with an trapezoidal output.

Triangular Oscillator with three cells: The last example will show an CNN with three cells, where the output of cell 1 is trained to be a triangular signal with an amplitude of 1 and a period time of 5. Of course the output is continuous in the linear region and the straight edges can not be obtained, but the output approximates the triangular signal (see Fig. 9). This time we learned the time constant of the system τ_x too (see Fig. 8). The initial condition was $\mathbf{x}(0) = (0, 2, -2)^T$.

$a =$	−4.426	1.217	2.146
$b =$	0.000	0.000	0.000
$i =$	0.095		
$\tau_x =$	2.447		

Fig. 8: The template values of the CNN with an triangular output.

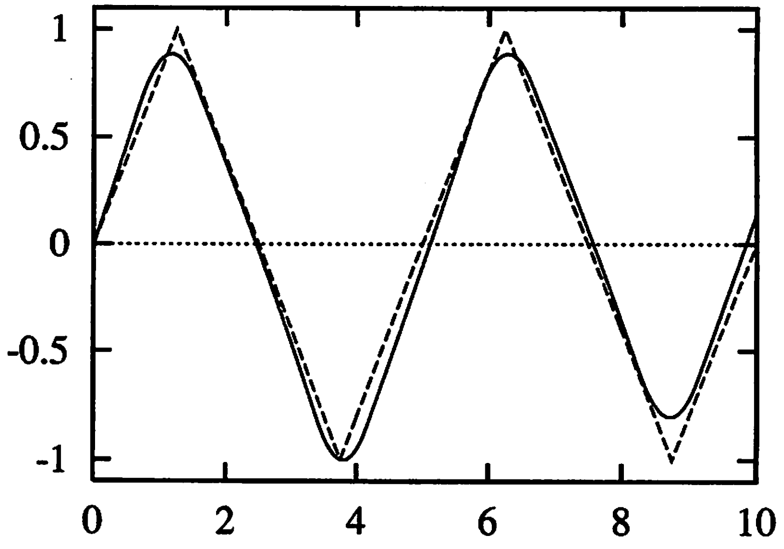

Fig. 9: The output of cell 1 of the triangular oscillator with three cells.

7 Conclusion

Our goal has been to find a learning algorithm for CNNs, which is able to learn the parameters of a CNN to perform a transition from an initial state to a state, which yields a prescribed output. We have shown a gradient technique which is able to learn the parameters such that a given error functional is minimized. Of course we find not always the global minimum, because the proposed algorithm uses the gradient of the error functional in the parameter space. Thus the algorithm stops, if the gradient is zero, whether this is a local minimum or a global one. Another drawback of the algorithm is the computational complexity. In order to compute the gradient of the error functional not only the differential equation of the CNN has to be integrated forward in time, but also the Euler-Lagrange equation, which is of the same dimensionality has to be integrated backward in time. Therefore the complete trajectory of the states has to be stored. This restricts the application of this algorithm to small examples. But in most cases this is not a severe restriction, as the learning can be achieved with relatively small samples.

Further investigations have to be made to examine the influence of error norm $2k$ and the parameter ρ on the speed, the ability to generalize and the avoidance of reaching local minima. And it is necessary to find algorithms with less computational effords in storage space and computing time.

References

[1] Forman S. Acton. *Numerical Methods That Work.* Harper and Row, New York, 1970.

[2] Luis B. Almeida. A learning rule for asynchonous perceptrons with feedback in a combinatorial environment. In *Proc. of the ICNN*, volume II, pages 609-618, San Diego, Calif., USA, 1987.

[3] L. O. Chua and Lin Yang. Cellular Neural Networks: Theory. *IEEE Tr. CAS*, 35:1257-1272, 1988.

[4] Leon O. Chua and Patrick Thiran. An analytic method for designing simple Cellular Neural Networks. *IEEE Trans. CAS*, 38(11):1332-1341, November 1991.

[5] R. Fletscher. *Practical Methods of Optimization.* Wiley + Sons, Chichester, second edition, 1989.

[6] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proc. Natl. Acad. Sci.*, volume 79, pages 2554-2558, April 1982.

[7] T. Matsumoto, L. O. Chua, and H. Suzuki. CNN cloning template: Connected component detector. *IEEE Trans. CAS*, 37:633-635, 1990.

[8] Stefan Miesbach. Efficient gradient computation for continuous and discrete time-dependent neural networks. *Proc. of the ISCAS-91, Singapore*, pages 2337-2342, June 1991.

[9] J. A. Nossek, G. Seiler, T. Roska, and L. O. Chua. Cellular Neural Networks: Theory and circuit design. Technical Report TUM-LNS-TR-90-7, Technical University Munich, 12 December 1990.

[10] Barak A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263-269, 1989.

[11] Fernando J. Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59(19):2229-2232, November 1987.

[12] Fernando J. Pineda. Dynamics and architecture for neural computation. *Journal of Complexity*, 4:216-243, November 1988.

[13] Fernando J. Pineda. Recurrent backpropagation and the dynamical approach to adaptive neural computation. *Neural Computation*, 1:161-172, 1989.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Parallel Distributed Processing*, volume 1. M.I.T. Press, 1986.

[15] Gerhard Seiler, Andreas J. Schuler, and Josef A. Nossek. Design of robust Cellular Neural Networks. Technical Report No. TUM-LNS-TR-91-13, Technical University Munich, September 1990.

[16] Krysztof Slot and Thomasz Kacprzak. Cellular neural network design problems for certain class of applications. In *Proc. of the ECCTD-91*, pages 516-523, Copenhagen, Denmark, 1991.

[17] Fan Zou, S. Schwarz, and J. A. Nossek. Cellular Neural Network design using a learning algorithm. In *Proc. of the first IEEE Int. Workshop on Cellular Neural Networks and their Applications, CNNA-90*, pages 73-81, December 1990.