Jens Müller

Eine digitale Implementierung Zellularer Nichtlinearer Netzwerke

Beiträge aus der Informationstechnik

Jens Müller

Eine digitale Implementierung Zellularer Nichtlinearer Netzwerke

VOGT

Dresden 2015

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Bibliographic Information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.dnb.de.

Zugl.: Dresden, Techn. Univ., Diss., 2015

Die vorliegende Arbeit stimmt mit dem Original der Dissertation „Eine digitale Implementierung Zellularer Nichtlinearer Netzwerke“ von Jens Müller überein.

Gesetzt vom Autor

ISBN 978-3-938860-83-0

Jörg Vogt Verlag
Niederwaldstr. 36
01277 Dresden
Germany

Phone: +49-(0)351-31403921
Telefax: +49-(0)351-31403918
e-mail: info@vogtverlag.de
Internet : www.vogtverlag.de

Technische Universität Dresden

Eine digitale Implementierung Zellularer Nichtlinearer Netzwerke

Jens Müller

von der Fakultät Elektrotechnik und Informationstechnik der Technischen Universität Dresden

zur Erlangung des akademischen Grades eines

Doktoringenieurs

(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. habil. Schüffny
Gutachter:	Prof. Dr. phil. nat. habil. Tetzlaff Prof. Dr. phil. nat. habil. Klauer
Tag der Einreichung:	01.12.2014
Tag der Verteidigung:	12.02.2015

Danksagung

Die vorliegende Arbeit entstand am Lehrstuhl für Grundlagen der Elektrotechnik an der Technischen Universität Dresden. Ich danke Herrn Prof. Ronald Tetzlaff für die langjährige Unterstützung und Betreuung am Lehrstuhl sowie für die Hinweise zur Erstellung und Überarbeitung des Manuskripts. Herrn Prof. Bernd Klauer danke ich herzlich für die Gutachtertätigkeit und Hinweise zur Arbeit.

Bei allen Kollegen des Lehrstuhls möchte ich mich für die wissenschaftliche Zusammenarbeit bedanken. Mein besonderer Dank gilt dabei Dr. Jan Müller für die Hilfe bei der Entwicklung und Implementierung, seine zahlreichen kritischen Anmerkungen zu dieser Arbeit und etlichen Publikationen und dem täglichen Espresso, der stets zu regen fachlichen Diskussionen führte. Besonders danken möchte ich auch Vanessa Senger für die Zusammenarbeit in mehreren Forschungsprojekten sowie Michele Lempke und Dr. Detlef Tolksdorf für die Hilfsbereitschaft und wertvolle Unterstützung bei der täglichen Arbeit am Lehrstuhl.

Ich danke Robert Braunschweig, Lennart Schierling, Simon Plaga, Xiangyi Meng und allen studentischen Hilfskräften für ihr großes wissenschaftliches Engagement, die meine Arbeit erheblich beflügelten. Miriam Helbig danke ich für die Anmerkungen zur Erstellung des Manuskripts. Des Weiteren danke ich Prof. Renate Merker, Prof. Torsten Schmidt, Dr. Jens Döge, Dr. Andreas Blug, Dr. Zoltán Nagy, Julia Hollmach, Nico Hoffmann, Ralf Becker, Peter Reichel und Stephan Günther für ihre Hilfsbereitschaft, Zusammenarbeit und viele inspirierende Diskussionen.

Bei meinen Eltern möchte ich mich herzlich dafür bedanken, dass sie mir ermöglichten, zu studieren und dass sie mich lehrten, kritisch zu hinterfragen. Mein Interesse für mathematische Zusammenhänge verdanke ich meinem Großvater Karl Heinz Schneider, der mich frühzeitig für Zahlen und Geometrie begeisterte, wofür ich ihm rückblickend nicht genug danken kann.

Abschließend möchte ich mich bei Luise bedanken, die mich immer wieder ermutigte, mir Freiräume zum Schreiben schaffte und mich mit ihrer Liebe täglich neu motiviert.

Inhaltsverzeichnis

1. Einleitung

Die Entwicklung der Computertechnik ist stets bestrebt, die Leistungsparameter eines Rechners zu verbessern. Dies betrifft in erster Linie die Rechenleistung und Energieeffizienz, die seit der Vorstellung des ersten Mikroprozessors (1971 durch Intel) von vielen Verbesserungen der Architektur, vor allem aber vom technologischen Fortschritt in der Mikroelektronik profitierten [67]. Durch eine Verringerung der Strukturbreite der Transistoren mit jeder Prozessorgeneration konnte die Integrationsdichte erhöht und dadurch immer mehr Transistoren auf einem integrierten Schaltkreis implementiert werden. Gleichzeitig ermöglichte die geringere Abmessung ein schnelleres Schalten der Transistoren und so den Betrieb mit höheren Taktfrequenzen, was zu einer höheren Rechenleistung führte. Damit folgt die Entwicklung seit etwa 50 Jahren dem Mooreschen Gesetz, das besagt, dass sich die Komplexität der integrierten Schaltkreise etwa alle 18 Monate verdoppelt [99].

Mit dem Prozessortakt steigt jedoch auch der Energieverbrauch um ein Vielfaches an, sodass vor etwa 15 Jahren ein Umdenken einsetzte und die Entwicklung von Mehrkernprozessoren zunehmende Bedeutung erlangte [7]. Mit steigender Anzahl an Rechenkernen nimmt aber auch die Komplexität der Softwareentwicklung zu, denn ein Programm kann nur dann von der Implementierung auf einem Mehrkernprozessor profitieren, wenn sich die Algorithmen effizient parallelisieren lassen und die Berechnung auf alle Rechenkerne verteilt werden kann. Zu den größten Herausforderungen der Softwareentwicklung für Mehrkernprozessoren mit gemeinsamem Speicher gehören daher der Lastenausgleich und die Kommunikation sowie die Synchronisierung der einzelnen Verarbeitungskerne, wobei die Herausforderung mit wachsender Zahl an Prozessorkernen zunimmt [109].

Eine Alternative zu konventionellen Mikroprozessoren mit Von-Neumann-Architektur und riesigem Befehlssatz sind sogenannte SIMD-Architekturen (*single instruction, multiple data*) für die hochgradig parallele Datenverarbeitung auf Vektorprozessoren oder Prozessorarrays [44]. Die einzelnen Prozessorelemente dieser Systeme führen gleichartige Operationen gleichzeitig auf verschiedenen Daten aus, wobei die Steuerung der einzelnen Rechenelemente nicht autonom, sondern von außen erfolgt [52]. Vor allem in Grafikprozessoren werden diese Architekturen seit über 30 Jahren für Aufgaben der Bild- und Videoverarbeitung eingesetzt. Zuletzt erlangte auch die Ausführung von Allzweck-Berechnungen (*general-purpose computing on graphics processing units*, GPGPU) auf diesen SIMD-Prozessoren vor allem im wissenschaftlichen Umfeld an Bedeutung [69, 70]. Die große Herausforderung besteht jedoch auch hier in der Entwicklung geeigneter Compiler, die es ermöglichen, einen bestehenden Algorithmus optimal auf der parallelen Architek-

tur abzubilden [138].

Entspricht die Datenstruktur jedoch exakt der Topologie des Prozessors, so ist die Suche nach einer geeigneten Abbildung des Algorithmus nicht erforderlich, da das zugrundeliegende Verarbeitungsprinzip bereits inhärent parallel ist. Mit dem Zellularen Nichtlinearen/Neuronalen Netzwerk (CNN) wurde von Chua und Yang solch ein Konzept für die multivariate, massiv-parallele Datenverarbeitung vorgestellt, das in der Folge zur Realisierung völlig neuer Rechnerarchitekturen führte [22]. CNNs besitzen häufig die Struktur einer zweidimensionalen Anordnung lokal gekoppelter dynamischer Systeme und nutzen komplexe räumlich-zeitliche Phänomene wie die Ausbreitung von Wellen oder die Generierung von Mustern, für die Lösung von Problemstellungen, deren Berechnung auf konventionellen Mikroprozessoren ineffizient ist. Dazu gehören z. B. Aufgaben der Mustererkennung und Klassifikation oder die Lösung partieller (nichtlinearer) Differentialgleichungen [121].

Die *CNN Universal Machine* wurde ursprünglich als analoger Universalrechner konzipiert, der die Rechenleistung eines Supercomputers auf einem einzigen Schaltkreis versprach [18]. Tatsächlich belegen zahlreiche VLSI-Implementierungen im Analog- und Mixed-Signal-Design die hohe Leistungsfähigkeit von CNNs, die vor allem durch die Integration in Kamerasystemen eine sehr schnelle und energieeffiziente Bildvorverarbeitung ermöglicht [32,56,80,111,119,120]. Durch die Verwendung von CNN-basierten Kamerasystemen konnten so bereits neue industrielle und akademische Anwendungen erschlossen werden, deren Realisierung mit konventionellen Bildverarbeitungssystemen nicht möglich gewesen wäre [11, 106, 158].

Aufgrund der geringen Rechengenauigkeit und störanfälligen Signalverarbeitung der analogen Schaltkreise werden zunehmend digitale Architekturen vorgestellt, die robustere und vor allem reproduzierbare Berechnungen ermöglichen [47,83,90,157]. Da die Entwicklung anwendungsspezifischer Schaltkreise mit einem hohen Zeit- und Kostenaufwand verbunden ist, erfolgt die Implementierung der digitalen Architekturen vor allem auf rekonfigurierbaren Bausteinen wie z. B. FPGAs (*field programmable gate arrays*) [2, 14, 87, 103, 108]. Dies erlaubt eine deutlich flexiblere Konfiguration der Hardware und damit auch die praktische Anwendung von CNNs abseits der Bildverarbeitung [53, 76, 104]. Obwohl in den letzten Jahren zahlreiche Architekturen und Implementierungen zur Emulation von CNNs auf FPGAs vorgestellt wurden, erfüllt keine dieser den ursprünglichen Anspruch einer universellen Rechenmaschine, die dem CNN zugeschrieben wird [20].

Ziel dieser Arbeit ist die Entwicklung eines Universalrechners, der das inhärent parallele Verarbeitungsprinzip des CNNs mit der Flexibilität und Rechengenauigkeit eines Digitalprozessors verbindet. Im Vordergrund steht dabei die Frage, wie ein zeitdiskretes CNN-Modell optimal in eine digitale Architektur überführt werden kann, um die Berechnung beliebiger Algorithmen zu ermöglichen und deren Verarbeitungszeit bei begrenzt verfügbaren Ressourcen zu minimieren. Dafür sollen verschiedene zeitdiskrete Modelle hinsichtlich ihrer Eignung für eine digitale Im-

plementierung verglichen und Verfahren zur Abbildung des bevorzugten Modells auf einem Prozessorarray systematisch untersucht werden. Auf Grundlage dieser Untersuchungen soll die Entwicklung einer Architektur erfolgen, deren Implementierung auf einem FPGA die Rekonfiguration mit geänderten Systemparametern ermöglicht und somit eine individuelle Anpassung der Hardware an die jeweilige Problemstellung gewährleistet.

In Kapitel 2 werden zunächst verschieden Modelle des CNNs mathematisch beschrieben und deren Anwendung an Beispielen der Bildverarbeitung und der Lösung partieller Differentialgleichungen erläutert. Ein Überblick zu bestehenden analogen und digitalen Implementierungen wird in Kapitel 3 gegeben. Dabei werden die Grenzen bisheriger digitaler Emulationen bezüglich der Realisierung eines CNN-Universalrechners dargelegt. Im Hinblick auf die spätere Hardware-Implementierung werden in Kapitel 4 verschiedene Verfahren zur Integration der Zustandsgleichung untersucht. Dies berücksichtigt neben der Genauigkeit der Lösung auch den potenziellen Ressourcenbedarf und die Rechenzeit. Methoden zur Abbildung des Netzwerks auf einem Prozessorarray werden in Kapitel 5 diskutiert. In Kapitel 6 wird die entwickelte Architektur vorgestellt und deren Implementierung auf einem FPGA erläutert. Mit dem Aufbau eines Echtzeit-Videoverarbeitungssystems wird der praktische Einsatz der Rechnerarchitektur in Kapitel 7 beschrieben. Abschließend wird eine Möglichkeit zur Erweiterung in Kapitel 8 diskutiert und in Kapitel 9 werden die Ergebnissse der Arbeit zusammengefasst.

2. Zellulare Nichtlineare Netzwerke

2.1. Einführung

1988 wurde das Modell der Zellularen Nichtlinearen[1] Netzwerke von Chua und Yang als ein neuartiges Paradigma der Informationsverarbeitung vorgestellt [21, 22]. Gegenüber den Theorien künstlicher Neuronaler Netzwerke stand hier jedoch von Anbeginn an die hochintegrierte schaltungstechnische Realisierung für Anwendungen in der Bildverarbeitung und Mustererkennung im Vordergrund. Das CNN wurde als analoge Schaltung bestehend aus linearen und nichtlinearen Netzwerkelementen konzipiert, deren Modell bis heute die Grundlage für die große Mehrheit aller praktischen Implementierungen ist. Das ursprüngliche Prinzip einer zweidimensionalen, regelmäßigen Anordnung einfacher dynamischer Elemente – sogenannter Zellen – wurde in der Folge immer wieder erweitert, verallgemeinert oder verändert, sodass CNNs heute eine Klasse verschiedener Systeme bilden, deren Definition sehr allgemein in [16] gegeben ist:

> A CNN is any spatial arrangement of locally-coupled cells, where each cell is a dynamical system which has an input, an output, and a state evolving according to some prescribed dynamical laws.

Demnach sind alle Variationen von CNNs durch eine regelmäßige Anordnung von lokal gekoppelten Elementen charakterisiert, deren dynamische Entwicklung für die Informationsverarbeitung essenziell ist, sei es durch die Auswertung eines stationären Zustands oder räumlich-zeitlicher Entwicklungen im Netzwerk. Obwohl der Bereich der direkten Wechselwirkungen auf die lokale Nachbarschaft einer Zelle beschränkt ist, können komplexe Phänomene wie die Propagation von Wellen und die Ausbildung von Mustern, auftreten. Eine Informationsverarbeitung, die sich dieser Phänomene bedient, unterscheidet sich von herkömmlichen Methoden, die auf logischen und arithmetischen Operationen beruhen und beispielsweise sequenziell auf Von-Neumann-Architekturen oder parallel auf Prozessorarrays implementiert werden können. Inspiriert vom biologischen Vorbild des menschlichen Gehirns bildet hier die zeitkontinuierliche Entwicklung einer Erregung im Verbund gekoppelter Verarbeitungseinheiten die Grundlage für Algorithmen der Kognition und Klassifikation und wird in Anlehnung einer wellenförmigen Ausbreitung im Netzwerk als *Cellular Wave Computing* bezeichnet [121].

[1] Die Begriffe Zellulares *Nichtlineares* Netzwerk und Zellulares *Neuronales* Netzwerk sind gleichberechtigt. In dieser Arbeit wird durchgehend die Bezeichnung Zellulares *Nichtlineares* Netzwerk verwendet.

2.2. Das Chua-Yang-Modell

Als Standardmodell für die Beschreibung von CNNs dient allgemein ein zweidimensionales, zeitkontinuierliches, homogenes Netzwerk $\mathcal{V}$, bestehend aus $M_\mathrm{v} \times N_\mathrm{v}$ Zellen [22]. Eine isolierte Zelle $C_{ij} \in \mathcal{V} = \left\{ C_{ij} \;\middle|\; 1 \leq i \leq M_\mathrm{v}, 1 \leq j \leq N_\mathrm{v} \right\}$ ist die kleinste Verarbeitungseinheit des Systems. Sie besitzt einen Eingang $u_{ij}(t)$, einen Schwellwert $z_{ij}(t)$, einen Ausgang $y_{ij}(t)$ und einen Zustand $x_{ij}(t)$ (Abbildung 2.1). Jede Zelle besitzt Verbindungen zu anderen Zellen innerhalb ihrer Nachbarschaft.

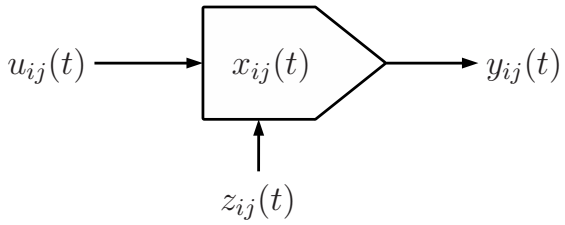


Abbildung 2.1.: Schematische Darstellung einer isolierten Zelle.

Dabei ist die Nachbarschaft $S_{ij}(r)$ mit dem Radius r einer Zelle C_{ij} definiert als:

$$S_{ij}(r) = \left\{ C_{kl} \;\middle|\; |k-i| \leq r, |l-j| \leq r, 1 \leq k \leq M_\mathrm{v}, 1 \leq l \leq N_\mathrm{v} \right\} . \tag{2.1}$$

Für viele Anwendungen der Bildverarbeitung [143] sowie der Lösung partieller Differentialgleichungen [127] ist $r = 1$ genügend, selten finden größere Umgebungen mit $r = 2$ oder $r = 3$ Verwendung [122]. Da ein wachsender Radius das zugrunde liegende Prinzip einer Architektur mit ausschließlich lokalen Kopplungen zunehmend konterkariert, fällt der 3×3-Nachbarschaft eine zentralen Bedeutung zu. In der vorliegenden Arbeit wird daher nur der Spezialfall $r = 1$ betrachtet.

Der Ausgang wird über eine beliebige Funktion $y_{ij}(x_{ij}(t)) = f(x_{ij}(t))$ gebildet, für die häufig die stückweise lineare Abbildung:

$$y_{ij}(x_{ij}(t)) = \frac{1}{2} \left(|x_{ij}(t) + 1| - |x_{ij}(t) - 1| \right) \tag{2.2}$$

verwendet wird. Damit ist der Ausgang auf das Intervall $[-1; 1]$ beschränkt, während der Dynamikbereich des Zustands unbegrenzt ist. Andere Ausgangsfunktionen wie z. B. die Sigmoidfunktion oder die Sprungfunktion spielen bei schaltungstechnische Implementierungen nur eine untergeordnete Rolle und werden in dieser Arbeit nicht betrachtet.

Die Verbindungen zu den Ausgängen der benachbarten Zellen werden als Rückkopplungen (*feedback*) und zu deren Eingängen als Vorwärtskopplungen (*feedforward*) bezeichnet. Die Art der Wechselwirkung ist über Gewichtsfunktionen definiert und bestimmt maßgeblich das dynamische Verhalten des Netzwerks. Obwohl

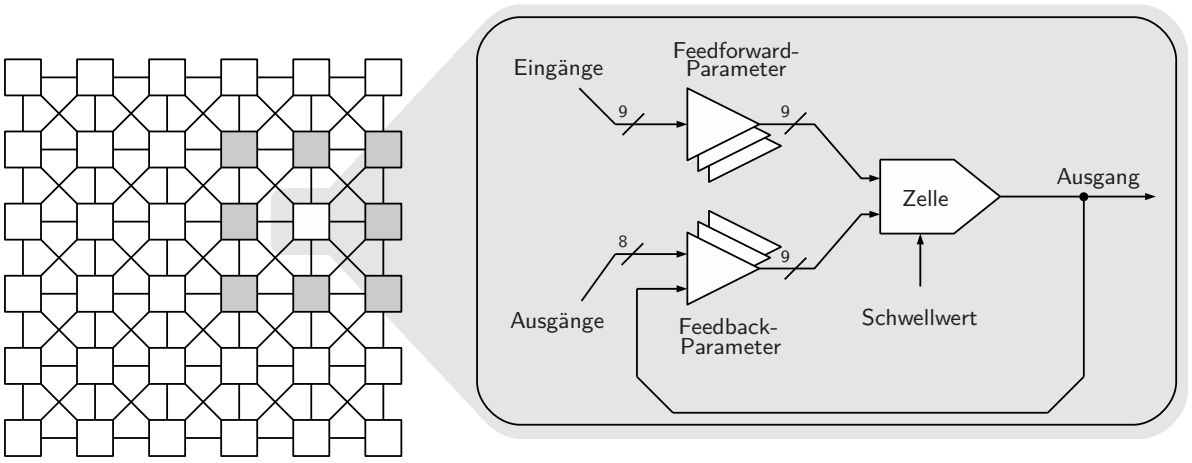


Abbildung 2.2.: Schematischer Aufbau des Netzwerks mit vereinfachter Darstellung der internen Wechselwirkung. Die Zellen innerhalb der 3 × 3-Nachbarschaft sind für die hervorgehobene Zelle dunkelgrau hinterlegt.

in Anlehnung an Neuronale Netze überwiegend lineare Gewichte verwendet werden, können beliebige Funktionen die Kopplungseigenschaften beschreiben [9] (siehe Kapitel 2.4). Abbildung 2.2 zeigt schematisch den Aufbau des Chua-Yang-Modells für ein Netzwerk mit 6×6 Zellen.

Die mathematische Beschreibung des Modells erfolgt in einem System gekoppelter nichtlinearer Differentialgleichungen. Die allgemeine Zustandsgleichung für die Zelle C_{ij} lautet für $u_{ij}(t) = u_{ij} = \text{const.}$ und $z_{ij}(t) = z_{ij} = \text{const.}$:

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{|k|,|l|\leq 1} a_{ij,kl} \cdot y_{i+k,j+l}(t) + \sum_{|k|,|l|\leq 1} b_{ij,kl} \cdot u_{i+k,j+l} + z_{ij}\,, \qquad (2.3)$$

mit den Feedback-Gewichten $a_{ij,kl}$ und den Feedforward-Gewichten $b_{ij,kl}$. Schaltungstechnisch relevant sind vor allem translationsinvariante Netzwerke, bei denen die Stärke der Wechselwirkung ausschließlich von der relativen Lage der Zellen zueinander abhängt und nicht von deren globalem Ort. Damit ist das dynamische Verhalten für jede Zelle durch die gleichen Parameter gegeben und mit $a_{ij,kl} = a_{kl}$, $b_{ij,kl} = b_{kl}$ und $z_{ij} = z, \forall i, j$ vereinfacht sich (2.3) zu:

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{|k|,|l|\leq 1} a_{kl} \cdot y_{i+k,j+l}(t) + \sum_{|k|,|l|\leq 1} b_{kl} \cdot u_{i+k,j+l} + z\,. \qquad (2.4)$$

Damit (2.4) für alle C_{ij} erfüllt ist, müssen diese eine vollständige Nachbarschaft besitzen. Dafür werden virtuelle Randzellen $C^*_{ij} \in \mathcal{V}^+ \backslash \mathcal{V}$, mit:

$$\mathcal{V}^+ = \left\{ C_{ij} \;\middle|\; 0 \leq i \leq M_\text{v} + 1, 0 \leq j \leq N_\text{v} + 1 \right\} \qquad (2.5)$$

definiert, die eine der folgenden Randbedingungen erfüllen müssen:

- **Dirichlet-Randbedingungen:** Alle virtuellen Randzellen C^*_{ij} besitzen einen konstanten Ausgangswert $y_{ij} = \xi_y$ und einen konstanten Eingangswert $u_{ij} = \xi_u$. In der Regel werden ξ_y und ξ_u auf den Wert null gesetzt.

- **Neumann-Randbedingungen:** Alle virtuellen Zellen C^*_{ij} besitzen den Eingangs- und Ausgangswert der nächstgelegenen Randzelle C_{kl} des Netzwerks.

- **Periodische Randbedingungen:** Die Ränder des Netzwerks werden periodisch in Form eines Torus fortgesetzt.

Das dynamische Verhalten des Netzwerks ist über die Ausgangsfunktion (2.2) der Zellen, die Kopplungsparameter, die Randbedingungen und die Anfangszustände $x_{ij}(0)$ vollständig beschrieben. Für translationsinvariante Netzwerke können die Kopplungsgewichte a_{kl}, b_{kl} und z für eine bessere Übersichtlichkeit als sogenanntes Template mit den Komponenten[2] **A**, **B** und **z** angegeben werden:

$$\mathbf{A} = \begin{array}{|c|c|c|}\hline a_{-1,-1} & a_{-1,0} & a_{-1,1} \\ \hline a_{0,-1} & a_{0,0} & a_{0,1} \\ \hline a_{1,-1} & a_{1,0} & a_{1,1} \\ \hline \end{array} \qquad \mathbf{B} = \begin{array}{|c|c|c|}\hline b_{-1,-1} & b_{-1,0} & b_{-1,1} \\ \hline b_{0,-1} & b_{0,0} & b_{0,1} \\ \hline b_{1,-1} & b_{1,0} & b_{1,1} \\ \hline \end{array} \qquad \mathbf{z} = z. \tag{2.6}$$

2.3. Informationsverarbeitung mit CNNs

2.3.1. Überblick

Die Informationsverarbeitung mit CNNs erfolgt wie bei Digitalrechnern nach dem EVA-Prinzip: Eingabe – Verarbeitung – Ausgabe von Daten mit dem Ziel, diese zu manipulieren oder Informationen zu gewinnen. Eingaben sind der Anfangszustand des Netzwerks und/oder konstante oder zeitlich veränderliche Eingänge. Als Ausgabe wird entweder die dynamische Entwicklung des Netzwerks selbst oder der Zellausgang zu einem bestimmten Zeitpunkt betrachtet. Häufig wird ein stabiler Gleichgewichtszustand angestrebt; seltener bilden oszillierende oder chaotische Zellzustände die Ausgabe des Netzwerks [51].

Die Wahl des Templates korrespondiert mit der Definition einer einzelnen arithmetischen oder logischen Funktion auf einem Digitalrechner und dient damit der „Programmierung“ des Netzwerks. Eine *CNN-Operation* ist eindeutig durch ein

[2]Zur Vereinfachung werden nachfolgend die einzelnen Komponenten bei isolierter Betrachtung ebenfalls als *Template* bezeichnet. Der Begriff **A**-*Template* ist dabei gleichbedeutend mit der Komponente **A** und der Begriff **B**-*Template* gleichbedeutend mit der Komponente **B**.

Template, die Randbedingungen des Netzwerks und dessen Anfangszustand definiert und beschreibt eine feste Verarbeitungsvorschrift für Eingabedaten. Die Abfolge mehrerer CNN-Operationen ergibt ein *CNN-Programm*[3] entsprechend einer Folge von Anweisungen auf einer CPU. An zwei einfachen Beispielen wird nachfolgend die Informationsverarbeitung mit CNNs kurz erläutert. Der Anschaulichkeit halber wurden dafür Anwendungen aus der Bildverarbeitung gewählt.

2.3.2. Kantenerkennung: Das Edge-Grey-Template

Eine Standardaufgabe der Bildverarbeitung ist das Erkennen von Übergängen zwischen zusammenhängenden Flächen in Farb-, Graustufen- oder Binärbildern. Diese Kanten lassen sich in der Regel problemlos durch die Anwendung von linearen Filtern (z. B. Sobel-Operator oder Laplace-Operator) extrahieren [64]. Die Möglichkeit zur näherungsweisen Lösung partieller Differentialgleichungen mit CNNs (vgl. Kapitel 2.6.2) erlaubt die Entwicklung einer CNN-Operation, die einem zweidimensionalen Laplace-Filter entspricht und damit Kanten in Graustufenbildern extrahiert.

Sei $\mathcal{B}$ ein zweidimensionales Bild, bestehend aus $M_\mathrm{b} \times N_\mathrm{b}$ Bildpunkten mit den Graustufenwerten:

$$\beta_{mn} \in \mathcal{B} = \left\{ \beta_{mn} \;\middle|\; 1 \leq m \leq M_\mathrm{b}, 1 \leq n \leq N_\mathrm{b} \right\} . \tag{2.7}$$

Für $M_\mathrm{v} = M_\mathrm{b}$ und $N_\mathrm{v} = N_\mathrm{b}$ lässt sich eine eineindeutige Abbildung des Bildes auf das Netzwerk $\mathcal{V}$ finden, die jedem Eingang u_{ij}, Ausgang y_{ij} oder Zustand x_{ij} der Zelle C_{ij} den Graustufenwert des Bildpunktes β_{ij} zuweist. Der Nomenklatur in [21] folgend, wird dabei dem Wert -1 der Tonwert „weiß“ und dem Wert $+1$ der Tonwert „schwarz“ zugeordnet. Graustufenwerte werden auf den Bereich $[-1; 1]$ verteilt.

Zur Kantenerkennung werden die Graustufenwerte des Bildes den Zelleingängen zugewiesen und dienen damit als konstanter Eingang $u_{ij} = \beta_{ij}, \forall\, i, j$ der Operation. Die Anfangszustände der Zellen besitzen in diesem Beispiel alle den Wert null, d. h.:

$$x_{ij}(t = 0) = x_{ij}(0) = 0, \forall\, i, j \tag{2.8}$$

und es gelten Dirichlet-Randbedingung mit:

$$\xi_u = \xi_y = 0. \tag{2.9}$$

Die Anwendung des Edge-Grey-Templates [65]:

$$\mathbf{A} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad \mathbf{B} = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \qquad \mathbf{z} = -0{,}5 \tag{2.10}$$

[3] Ein CNN-Programm kann sich ggf. in verschiedene *CNN-Subroutinen* untergliedern (siehe Kapitel 4.2).

legt die Kopplungsgewichte zwischen den Zellen und damit die dynamischen Eigenschaften des Netzwerks fest. Der Ausgang $y_{ij}(\infty)$ liefert ein weißes Ergebnisbild mit schwarzen Bildpunkten an den Stellen scharfer Kanten des Eingangsbildes.

Das resultierende Netzwerk gehört zur Klasse der ungekoppelten CNNs, d. h. alle Rückkopplungen zu benachbarten Zellen sind null außer der Rückkopplung der Zelle auf sich selbst ($a_{0,0} \neq 0$). Damit vereinfacht sich die Zellzustandsgleichung (2.4) zu:

$$\dot{x}_{ij}(t) = -x_{ij}(t) + a_{0,0} \cdot y_{i,j}(t) + w_{ij}\,, \tag{2.11}$$

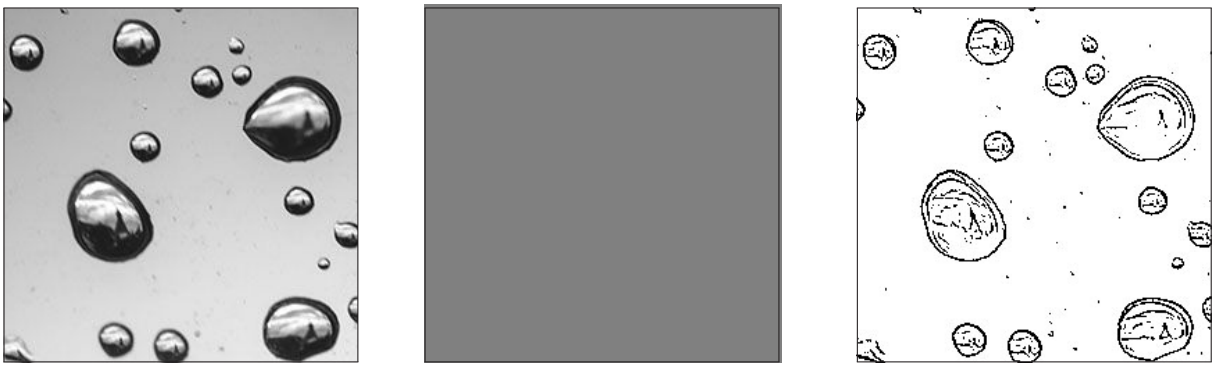
mit dem konstanten Offset:

$$w_{ij} = \sum_{|k|,|l|\leq 1} b_{kl} \cdot u_{i+k,j+l} + z\,. \tag{2.12}$$

Es ist offensichtlich, dass der Offset für jeden Bildpunkt einen anderen Wert annimmt abhängig von der Menge an weißen, schwarzen und grauen Bildpunkten innerhalb der Nachbarschaft. Wie in [19] gezeigt ist, können in Abhängigkeit von w_{ij} für den Zellausgang drei Fälle unterschieden werden:

1. $w_{ij} > 0 \rightarrow y_{ij}(\infty) = 1$ (schwarz)
2. $w_{ij} < 0 \rightarrow y_{ij}(\infty) = -1$ (weiß)
3. $w_{ij} = 0 \rightarrow y_{ij}(\infty) = 0$ (grau).

Abbildung 2.3 zeigt die Wirkung des Edge-Grey-Templates auf ein Eingangsbild mit 256×256 Bildpunkten.



(a) Zelleingänge u_{ij} (b) Anfangszustände $x_{ij}(0)$ (c) Zellausgänge $y_{ij}(\infty)$

Abbildung 2.3.: Beispiel für eine einfache CNN-Operation: Kantendetektion

2.3.3. Komplexe Verarbeitung: Die Connected Component Detection

Ein prominentes Beispiel für eine komplexere Verarbeitung mit CNNs ist die Erkennung zusammenhängender schwarzer Bereiche in einem Bild. Die *Connected Component Detection* (CCD) [94] erfolgt auf gekoppelten CNNs, deren Rückkopplungsgewichte zu benachbarten Zellen ungleich null sind und die in der Regel ein komplexeres dynamisches Verhalten als ungekoppelte CNNs besitzen [19]. Eine CCD zählt die auftretenden Schwarz-Weiß-Übergänge eines binären Eingangsbildes in einer der acht Richtungen (horizontal, vertikal, diagonal) und nutzt dafür eine dynamische Entwicklung von propagierenden Wellen. Als Beispiel wird hier eine CCD in horizontaler Richtung betrachtet.

Die Graustufenwerte des Bildes $\mathcal{B}$ werden den Anfangszuständen der Zellen zugewiesen:

$$x_{ij}(0) = \beta_{ij}, \forall\, i, j\,. \tag{2.13}$$

Das Netzwerk ist autonom, d. h. Zelleingänge haben auf diese CNN-Operation keinen Einfluss und können daher null gesetzt werden:

$$u_{ij} = 0, \forall\, i, j\,. \tag{2.14}$$

Für die Ausgänge der virtuellen Zellen gelten Dirichlet-Randbedingungen:

$$\xi_y = 0\,. \tag{2.15}$$

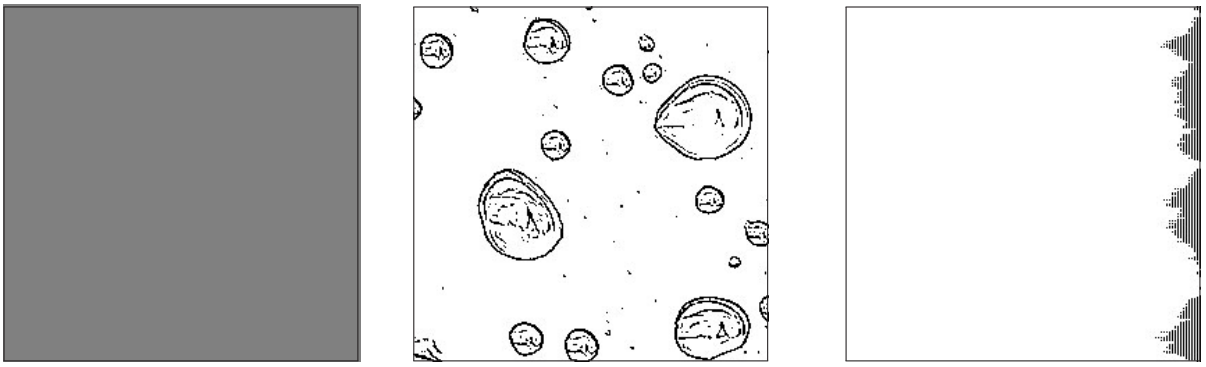
Das Template für eine horizontale CCD nach rechts lautet:

$$\mathbf{A} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 1 & 2 & -1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad \mathbf{B} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad \mathbf{z} = 0. \tag{2.16}$$

Der Ausgang $y_{ij}(\infty)$ ist ein weißes Ergebnisbild mit je einem schwarzen Bildpunkt am rechten Rand für einen zusammenhängenden schwarzen Bereich derselben Zeile. Das Ergebnis liefert eine Aussage über die Anzahl von Schwarz-Weiß-Übergängen je Zeile im Bild und kann in einer weiterführenden Verarbeitung z. B. zur Merkmalsextraktion [53] oder Klassifikation [94] genutzt werden. Ein Beispiel für eine CCD ist in Abbildung 2.4 dargestellt. Als Anfangszustand wurde hier das Ergebnisbild der Kantenerkennung aus Abschnitt 2.3.2 gewählt.

2.3.4. Verarbeitung beliebiger Algorithmen mit der CNN Universal Machine

Für eine praktische Informationsverarbeitung ist das isolierte CNN ungeeignet. Zwar eröffnet die freie Wahl von Kopplungsparametern, Anfangs- und Randbedingungen eine nahezu unbegrenzte Vielfalt im dynamischen Verhalten des Netzwerks;



(a) Zelleingänge u_{ij} (b) Anfangszustände $x_{ij}(0)$ (c) Zellausgänge $y_{ij}(\infty)$

Abbildung 2.4.: Beispiel für eine Connected Component Detection

Algorithmen erfordern jedoch in der Regel die sequenzielle Verarbeitung mehrerer Operationen, die in einer Abfolge von Instruktionen hinterlegt sind. Das motiviert die theoretische Beschreibung eines programmierbaren CNN-Prozessors, der das Netzwerk um Speicher-, Steuer- und Kommunikationseinheiten ergänzt.

Die *CNN Universal Machine* (CNN-UM) [18] beschreibt ursprünglich einen analogen Array-Prozessor mit lokalen analogen und logischen Speicher- und Verarbeitungseinheiten[4]. Die CNN-Zelle wird um jeweils eine analoge und eine logische Ausgabeeinheit ergänzt (*local analogue output unit* – LAOU bzw. *local logic unit* – LLU). Zwischenergebnisse, die im lokalen analogen (*local analogue memory* – LAM) oder logischen Speicher (*local logic memory* – LLM) hinterlegt sind, können hier schnell kombiniert werden, ohne dafür das CNN selbst zu beanspruchen. Beispielsweise kann die boolesche Verknüpfung von zwei binären Ergebnisbildern in der LLU erfolgen [126]. Eine globale Steuereinheit (*global analogic programming unit* – GAPU) umfasst unter anderem den analogen Speicher für Templates, Speicher für den Code der logischen Operationen und Speicher für die Zellkonfigurationen der einzelnen CNN-Operationen.

In [20] wurde gezeigt, dass mit einer CNN-UM das *Game of Life* dargestellt werden kann. Damit ist die CNN-UM eine universelle Turingmaschine und ermöglicht die Verarbeitung beliebig komplexer Algorithmen. Die zugrunde liegende Definition der CNN-UM als analoger/logischer Array-Prozessor muss jedoch als theoretisches Paradigma verstanden werden, da deren praktische Implementierungen teils erheblich von der in [20], [125] und [126] beschriebenen Architektur abweichen (vgl. dazu Kapitel 3). Nach [121] kann die Implementierung der CNN-UM unter anderem auch als digitale Emulation erfolgen, was wiederum die Notwendigkeit einer allgemeingültigen Definition zeigt.

Definition 1. Eine CNN Universal Machine ist ein programmierbarer Array-Prozessor, basierend auf der analogen und/oder digitalen, schaltungstechnischen

[4]In diesem Zusammenhang wurde das Kunstwort *analogic computing* für die Verbindung analoger Rechentechnik mit logischen Operatoren geschaffen.

Implementierung einer CNN-Zustandsgleichung, ergänzt um Speicher-, Steuer- und Kommunikationseinheiten.

Ein praktisches Beispiel für ein Programm auf einer CNN-UM ist die Berechnung der XOR-Funktion. Das XOR gehört zur Klasse der nicht linear separierbaren Funktionen, d. h. es lässt sich keine lineare Funktion (und auch kein CNN-Template mit linearen Gewichten) finden, die die logische Verknüpfung:

$$\mathcal{B}_a = \mathcal{B}_{e1} \oplus \mathcal{B}_{e2} \tag{2.17}$$

realisiert. Um (2.17) trotzdem mit einem CNN lösen zu können, muss entweder das Netzwerkmodell selbst geändert (siehe Kapitel 2.4.3) oder die Funktion als CNN-Programm verarbeitet werden. Die Darstellung von (2.17) als disjunktive Normalform liefert:

$$\mathcal{B}_{e1} \oplus \mathcal{B}_{e2} = \left(\overline{\mathcal{B}_{e1}} \wedge \mathcal{B}_{e2}\right) \vee \left(\mathcal{B}_{e1} \wedge \overline{\mathcal{B}_{e2}}\right) . \tag{2.18}$$

Damit lässt sich das XOR in Algorithmus 1 als Abfolge boolescher Operatoren formulieren, die wiederum jeweils als CNN-Operationen ausgeführt werden können.

Algorithmus 1 XOR

$\mathcal{B}_{t1} =$ NOT $\mathcal{B}_{e1}$
$\mathcal{B}_{t2} =$ NOT $\mathcal{B}_{e2}$
$\mathcal{B}_{t3} = \mathcal{B}_{e1}$ AND $\mathcal{B}_{t2}$
$\mathcal{B}_{t4} = \mathcal{B}_{e2}$ AND $\mathcal{B}_{t1}$
$\mathcal{B}_{a} = \mathcal{B}_{t3}$ OR $\mathcal{B}_{t4}$

2.4. Erweiterung des CNN-Modells

Das oben beschriebene Chua-Yang-Modell dient als Standardmodell von CNNs, da die Struktur des zweidimensionalen, translationsinvarianten Netzwerks mit skalaren Kopplungsgewichten vergleichsweise einfach beschrieben ist und mehrfach mathematisch untersucht wurde [24, 51]. Trotzdem wurde das Modell immer wieder im Hinblick auf spezifische Anwendungen und Implementierungen verändert und erweitert. Auf diese Weise entstand eine große Zahl von Modellen, die sich auf die ursprüngliche Definition in [22] beziehen, bisher jedoch nicht näher klassifiziert wurden. Nachfolgend werden jene Modelle vorgestellt, die für die Entwicklung einer digitale Implementierung im Rahmen dieser Arbeit von Bedeutung sind.

2.4.1. Das zeitdiskrete CNN

Sowohl die Simulation von CNNs auf einem Digitalrechner als auch dessen Implementierung als digitale Schaltung erfordert die zeitliche Diskretisierung der Zellzustandsgleichung (2.4). Zur Lösung der Zustandsgleichung wurden dazu in [58] erstmals verschiedene numerische Integrationsverfahren hinsichtlich ihrer Genauigkeit, Geschwindigkeit und Robustheit untersucht. Wie zu erwarten, zeigten die Autoren am Beispiel einer CCD, dass mit einem Runge-Kutta-Verfahren 4. Ordnung eine deutlich höhere Genauigkeit gegenüber einem einfacheren Prädiktor-Korrektor-Verfahren oder einem expliziten Euler-Verfahren erreicht wird. Obwohl dies von großer Bedeutung für Anwendungen ist, die eine hohe Rechengenauigkeit erfordern, spielt dies nur eine untergeordnete Rolle für CNN-Operationen, die hauptsächlich eine qualitative Aussage liefern. Dazu zählen Funktionen mit binären Eingangs- und Ausgangsdaten wie z. B. logische Verknüpfungen. Für diese ist die zweite Erkenntnis der Autoren interessanter: Unter Verwendung des beschriebenen Runge-Kutta-Verfahrens konnte für eine korrekte Berechnung des binären Ergebnisbildes der CCD die Schrittweite der Diskretisierung gegenüber dem Euler-Verfahren erheblich größer gewählt werden, was tendenziell mit einer Verringerung der Rechenzeit einhergeht. Nachfolgend wird die Diskretisierung der Zustandsgleichung nach dem expliziten Euler-Verfahren kurz erläutert. Eine ausführliche Diskussion verschiedener Integrationsverfahren erfolgt in Kapitel 4.4.

Die Anwendung des expliziten Euler-Verfahrens zur Diskretisierung von (2.4) mit einer konstanten Schrittweite h liefert:

$$\frac{x_{ij}(n+1) - x_{ij}(n)}{h} = -x_{ij}(n) + \sum_{|k|,|l|\leq 1} a_{kl} \cdot y_{i+k,j+l} + \sum_{|k|,|l|\leq 1} b_{kl} \cdot u_{i+k,j+l} + z\,. \quad (2.19)$$

Durch Umstellen erhält man die Differenzengleichung für ein zeitdiskretes CNN:

$$x_{ij}(n+1) = (1-h)x_{ij}(n) + h \sum_{|k|,|l|\leq 1} a_{kl} \cdot y_{i+k,j+l}(n) + h \sum_{|k|,|l|\leq 1} b_{kl} \cdot u_{i+k,j+l} + hz\,. \quad (2.20)$$

Der Begriff *Discrete-Time Cellular Neural Network* (DTCNN) wurde in [57] ursprünglich für $h = 1$ definiert. Damit vereinfacht sich (2.20) zu:

$$x_{ij}(n+1) = \sum_{|k|,|l|\leq 1} a_{kl} \cdot y_{i+k,j+l}(n) + \sum_{|k|,|l|\leq 1} b_{kl} \cdot u_{i+k,j+l} + z\,. \quad (2.21)$$

Während (2.20) für den Grenzfall $h \to 0$ gegen das zeitkontinuierliche Modell konvergiert, weist ein DTCNN mit $h = 1$ unter Umständen eine völlig andere Dynamik auf. Abgesehen von Netzwerken mit chaotischem Verhalten oder Operationen, deren Trajektorie nahe eines Bifurkationspunktes verläuft, liefert bei Operationen der Bildverarbeitung eine Schrittweite von $h \approx 0{,}1$ für das Euler-Verfahren in der Regel vergleichbare Ergebnisse zum Chua-Yang-Modell [19].

2.4.2. Das Full-Signal-Range-Modell

Eine schaltungstechnische Implementierung des Chua-Yang-Modells ist problematisch, da der Zustand in (2.4) theoretisch beliebig groß werden kann[5]. Im Zusammenhang mit der Entwicklung einer CMOS-Implementierung wurde daher in [119] ein verändertes Modell unter dem Namen *full range model* vorgestellt. Nachfolgend wird die später eingeführte Bezeichnung des *Full-Signal-Range-Modells* (FSR-Modell) [37] verwendet.

Die CNN-Zustandsgleichung (2.4) wird um einen nichtlinearen Term $g(x_{ij})$ erweitert:

$$\dot{x}_{ij}(t) = -x_{ij}(t) - g\left(x_{ij}\right) + \sum_{|k|,|l|\leq 1} a_{kl} \cdot y_{i+k,j+l}(t) + \sum_{|k|,|l|\leq 1} b_{kl} \cdot u_{i+k,j+l} + z\,, \quad (2.22)$$

mit

$$g(x_{ij}) = \begin{cases} m\left(x_{ij}+1\right), & x_{ij} < -1 \\ 0, & |x_{ij}| \leq 1 \\ m\left(x_{ij}-1\right), & x_{ij} > 1 \end{cases}. \quad (2.23)$$

Für $m = 0$ reduziert sich (2.22) auf (2.4), daher kann das Chua-Yang-Modell als Speziallfall des FSR-Modells angesehen werden. Mit $m \to \infty$ wird der Zustand selbst beschränkt und es gilt unter der Voraussetzung $|x_{ij}(0)| \leq 1, \forall\, i,j$ und $|u_{ij}| \leq 1, \forall\, i,j$:

$$|x_{ij}(t)| \leq 1, \forall\, i,j\,. \quad (2.24)$$

In diesem Fall nimmt der Zellausgang den gleichen Wert wie der Zellzustand an und (2.22) vereinfacht sich unter der Bedingung (2.2) zu:

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{|k|,|l|\leq 1} a_{kl} \cdot x_{i+k,j+l}(t) + \sum_{|k|,|l|\leq 1} b_{kl} \cdot u_{i+k,j+l} + z\,. \quad (2.25)$$

In [25] wurde gezeigt, dass die hinreichenden Bedingungen für Stabilität des Chua-Yang- und des FSR-Modells identisch sind, obwohl sich die jeweiligen dynamischen Entwicklungen durchaus voneinander unterscheiden können. Laut [37] ist das qualitative Verhalten der beiden Modelle so ähnlich, dass sich fast alle CNN-Algorithmen problemlos auf das FSR-Modell übertragen lassen.

Für das FSR-Modell eines DTCNNs beziehen sich die Autoren in [119] auf das in (2.21) definierte Modell von Harrer. Um den reduzierten Dynamikbereich des Zustandes zu gewährleisten, wird der Zellzustand selbst über die nichtlineare Ausgangsfunktion $f(\bullet)$ begrenzt:

$$x_{ij}(n+1) = f\left(\sum_{|k|,|l|\leq 1} a_{kl} \cdot x_{i+k,j+l}(n) + \sum_{|k|,|l|\leq 1} b_{kl} \cdot u_{i+k,j+l} + z\right). \quad (2.26)$$

[5] Praktisch nimmt der Zustand laut [119] bei typischen CNN-Operationen Maximalwerte zwischen 5 und 10 an.

2.4.3. CNN mit polynomiellen Kopplungen

Im Kapitel 2.3.4 wurde die XOR-Funktion als Beispiel für eine Aufgabe beschreiben, die nicht mit einem Standard-CNN gelöst werden kann. Mit Einführung der CNN-UM ist es zwar möglich, jede beliebige Funktion als CNN-Algorithmus zu beschreiben; unter Umständen geht dabei jedoch durch sequenziell ausgeführte Operationen die Leistungsfähigkeit des Netzwerks selbst verloren. So erfordert bereits die recht simple XOR-Funktion in Algorithmus 1 die Verarbeitung von fünf Einzeloperationen und das Speichern von vier Teilergebnissen ($\mathcal{B}_{ti},\ i = 1 \ldots 4$). Darüber hinaus lassen sich nichtlineare partielle Differentialgleichungen wie die Burgers-Gleichung oder Korteweg-de-Vries-Gleichung nicht mit einem Standard-CNN lösen.

Daher wurden CNNs schon frühzeitig um nichtlineare Kopplungsterme erweitert und das Modell damit verallgemeinert [124]. Die Gewichte sind nun nicht mehr skalar, sondern nichtlineare oder zweidimensionale Funktionen. Dem Approximationssatz von Weierstraß [145] folgend, dass eine stetige Funktion durch ein Polynom approximiert werden kann, haben Netzwerke mit polynomiellen Kopplungen eine besondere Bedeutung erlangt. Für das polynomielle CNN (*Polynomial-Type CNN* – PTCNN) existiert keine einheitliche Definition, jedoch haben sich im Wesentlichen zwei Modelle etabliert.

Polynomielle Kopplungen ohne Mischterme

Die Vorwärtskopplungen des Netzwerks werden zu Polynomen der Zelleingänge und die Rückkopplungen zu Polynomen der Zellausgänge erweitert. Mit der Polynomordnung P lautet die Zellzustandsgleichung (2.4) damit [25]:

$$\begin{aligned}\dot{x}_{ij}(t) = -x_{ij}(t) &+ \sum_{|k|,|l|\leq 1} \left[a_{kl}^{(1)} \cdot y_{i+k,j+l}(t) + \cdots + a_{kl}^{(P)} \cdot y_{i+k,j+l}^{P}(t) \right] \\ &+ \sum_{|k|,|l|\leq 1} \left[b_{kl}^{(1)} \cdot u_{i+k,j+l} + b_{kl}^{(2)} \cdot u_{i+k,j+l}^{2} + \cdots + b_{kl}^{(P)} \cdot u_{i+k,j+l}^{P} \right] + z\,.\end{aligned} \tag{2.27}$$

Durch Zusammenfassen erhält man:

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{|k|,|l|\leq 1} \sum_{p=1}^{P} a_{kl}^{(p)} \cdot y_{i+k,j+l}^{p}(t) + \sum_{|k|,|l|\leq 1} \sum_{p=1}^{P} b_{kl}^{(p)} \cdot u_{i+k,j+l}^{p} + z\,. \tag{2.28}$$

Ein Template besteht nun aus den Komponenten: $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$, ... , $\mathbf{A}^{(P)}$, $\mathbf{B}^{(1)}$,

$\mathbf{B}^{(2)}, \ldots \mathbf{B}^{(P)}$ und $\mathbf{z}$, mit:

$$\mathbf{A}^{(p)} = \begin{array}{|c|c|c|}\hline a^{(p)}_{-1,-1} & a^{(p)}_{-1,0} & a^{(p)}_{-1,1} \\ \hline a^{(p)}_{0,-1} & a^{(p)}_{0,0} & a^{(p)}_{0,1} \\ \hline a^{(p)}_{1,-1} & a^{(p)}_{1,0} & a^{(p)}_{1,1} \\ \hline \end{array} \quad \text{und} \quad \mathbf{B}^{(p)} = \begin{array}{|c|c|c|}\hline b^{(p)}_{-1,-1} & b^{(p)}_{-1,0} & b^{(p)}_{-1,1} \\ \hline b^{(p)}_{0,-1} & b^{(p)}_{0,0} & b^{(p)}_{0,1} \\ \hline b^{(p)}_{1,-1} & b^{(p)}_{1,0} & b^{(p)}_{1,1} \\ \hline \end{array} \quad . \tag{2.29}$$

Damit erhöht sich die Zahl der Kopplungsparameter von bisher 19 für das Chua-Yang-Modell auf:

$$n_{\text{par}} = 2 \cdot 9 \cdot P + 1 \,. \tag{2.30}$$

Für $P = 1$ geht (2.28) in (2.4) über, womit das Standard-CNN als Sonderfall des PTCNNs betrachtet werden kann. Entsprechend Kapitel 2.4.1 können zur Lösung der Zustandsgleichung des PTCNNs verschiedene Integrationsverfahren verwendet werden. Durch Anwendung des expliziten Euler-Verfahrens erhalt man analog zu (2.20):

$$x_{ij}(n+1) = (1-h)x_{ij}(n) + h \sum_{|k|,|l|\leq 1} \sum_{p=1}^{P} a^{(p)}_{kl} \cdot y^p_{i+k,j+l}(n) + h \sum_{|k|,|l|\leq 1} \sum_{p=1}^{P} b^{(p)}_{kl} \cdot u^p_{i+k,j+l} + hz \,. \tag{2.31}$$

Eine mögliche Anwendung des polynomiellen Modells wird in Kapitel 2.6.2 beschrieben.

Polynomielle Kopplungen mit Mischtermen

Das PTCNN mit Mischtermen erweitert das oben beschrieben Modell zusätzlich um Produkte zwischen Potenzen der Zelleingänge und Ausgänge. Damit wird das bestehende Prinzip aus getrennten Pfaden für Vorwärts- und Rückkopplungen verlassen und die Komplexität des Netzwerks nimmt weiter zu. Die allgemeine Zellzustandsgleichung für ein PTCNN mit Mischtermen lautet:

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{|k|,|l|\leq 1} \sum_{p=0}^{P} \sum_{q=0}^{Q} c^{(p),(q)}_{kl} \cdot y^p_{i+k,j+l}(t) \cdot u^q_{i+k,j+l} \,, \tag{2.32}$$

mit den Kopplungsparametern $c^{(p),(q)}_{kl}$. Gleichung (2.32) lässt sich wiederum in (2.4) überführen, wenn gilt:

$$\sum_{|k|,|l|\leq 1} c^{(0),(0)}_{kl} = z,\ c^{(1),(0)}_{kl} = a_{kl},\ c^{(0),(1)}_{kl} = b_{kl} \text{ und } c^{(p),(q)}_{kl} = 0, \text{ sonst.} \tag{2.33}$$

In [55] wurde gezeigt, dass die XOR-Funktion mit diesem Modell in einer einzigen CNN-Operation berechnet werden kann.

2.5. Lern- und Optimierungsverfahren

Das Verhalten von CNNs wurde umfangreich studiert und mittlerweile existieren ganze Bibliotheken mit Sammlungen von Templates für verschiedene Operationen, meist aus dem Bereich der Bildverarbeitung [65]. Dies schließt auch verschiedene Erweiterungen des Chua-Yang-Modells, wie PTCNN und das FSR-Modell, mit ein. Trotzdem müssen für die Lösung neuer Problemstellungen häufig auch neue Netzwerkparameter gefunden werden. Eine Möglichkeit dazu bietet das „Training“ des Netzwerks in einem überwachten Lernprozess anhand von Eingabe- und Referenzdaten [71]. Auf diese Weise können neue CNN-Operationen ohne spezifisches Wissen über die jeweiligen Kopplungseigenschaften im Netzwerk entwickelt werden, wodurch vor allem bei Netzwerken mit nichtlinearen Gewichtsfunktion eine sehr aufwändige analytische Untersuchung vermieden wird [40, 129].

Wird eine Operation auf einem CNN-Chip implementiert, so kann unter Umständen eine Anpassung der Netzwerkparameter erforderlich sein [59]. Vor allem bei analogen Designs können Fertigungstoleranzen, die begrenzte Darstellungsgenauigkeit und Rauschen zu anderen Ergebnissen als reine Software-Simulationen führen. Durch die Ausführung einer Parameteroptimierung auf dem Schaltkreis können dessen spezifische Eigenschaften berücksichtigt und das Template an die Hardware angepasst werden [28, 50].

Für das Training der Parameter werden häufig Optimierungsverfahren verwendet, die auf einem genetischen Algorithmus oder dem Simulated Annealing basieren [28, 54, 110, 129, 141]. Obwohl inzwischen eine Vielzahl von Beiträgen mit unterschiedlichen Algorithmen für CNN-Lernverfahren publiziert wurde, ist das prinzipielle Vorgehen jedoch immer gleich. Zunächst werden problemangepasste und möglichst redundanzfreie Trainingsdaten für Eingänge und Anfangszustände sowie Referenzdaten für das gewünschte Ergebnis der Operation erstellt. Als Ausgangspunkt wird mindestens ein Parametersatz generiert (zufällig oder anhand einer bereits bekannten Operation) und der Ausgang des Netzwerks durch Anwendung dieser Parameter wird bestimmt. Danach wird eine Kostenfunktion ermittelt, die die Abweichung des erzielten Ergebnisses von den gegebenen Referenzdaten repräsentiert. Anschließend werden die Parameter verändert und der Vorgang solange wiederholt, bis der Fehler eine festgelegte Grenze unterschreitet oder ein anderes Abbruchkriterium erfüllt wird. Gegebenenfalls erfolgt im Anschluss eine umfangreiche Verifikation des ermittelten Parametervektors anhand verschiedener Testdaten.

2.6. Anwendungsbeispiele

2.6.1. Hochgeschwindigkeits-Bildverarbeitung

Aufgrund ihrer Topologie sind CNNs prädestiniert für viele Anwendungen der zweidimensionalen Bildverarbeitung, der damit eine herausragende Bedeutung für den

praktischen Einsatz der Netzwerke zukommt. Das Standard-CNN besteht aus einer regulären, zweidimensionalen Anordnung von Zellen, die direkt mit den einzelnen Punkten eines Bildes korrespondieren und damit eine vollständig parallele Signalverarbeitung für jedes Pixel ermöglichen (vgl. Kapitel 2.3.2). Ein ungekoppeltes CNN mit $a_{kl} = 0, \forall k, l$ und $z = 0$ ist äquivalent zu einem Filter mit endlicher Impulsantwort, wobei der Faltungskern dem um 180° gedrehten **B**-Template entspricht. Nach [19] ist die Struktur gekoppelter Netzwerke sogar äquivalent zu einem Filter mit unendlicher Impulsantwort und erlaubt damit die Berechnung deutlich komplexerer Funktionen.

Die massiv parallele Verarbeitung des CNNs ist der Schlüssel für eine Hochgeschwindigkeits-Bildverarbeitung, die die Leistungsfähigkeit herkömmlicher, nach der Von-Neumann-Architektur arbeitender Digitalprozessoren deutlich übersteigt. Voraussetzung dafür sind allerdings schaltungstechnische Realisierungen, die eine direkte Eingabe von Bilddaten erlauben. Im Kapitel 3 werden verschiedene Systeme vorgestellt, die vor allem in der Bild- und Videosignalverarbeitung eingesetzt werden können und damit den Grundstein für einen praktischen Einsatz von CNNs in industriellen und akademischen Anwendungen bilden.

Unter Verwendung eines CNN-basierten Kamerasystems konnte erstmals die Einschweißtiefe eines Laserschweißprozesses in Echtzeit geregelt werden [1]. Die enorme Geschwindigkeit des dabei verwendeten analogen, zellularen Rechners zur Vorverarbeitung von Aufnahmen der Schweißnaht ermöglicht die Regelung der Laserleistung bei wechselnden Prozessparametern (Vorschubgeschwindigkeit, Blechdicke, Richtung, etc.) mit Bildwiederholraten bis zu 14 kHz. Dadurch wird eine Zerstörung des Unterblechs verhindert und die Entstehung unerwünschter Spritzer detektiert [107]. Mit der gleichen Technologie erfolgte die erfolgreiche Qualitätskontrolle eines Drahtziehprozesses, um Defektstellen mit einer Größe von ca. 100 µm bei einer Vorschubgeschwindigkeit von 10 m/s zu detektieren. Die dafür erforderliche Bildwiederholrate von mindestens 3,5 kHz konnte nicht mit herkömmlichen Kamera- und Bildverarbeitungssystemen erreicht werden [11].

Weitere Beispiele für den praktischen Einsatz von CNNs in der Bildverarbeitung sind die Gesichtserkennung [10], die automatische Hinderniserkennung bei Fahrzeugen [41], die Erkennung bewegter Objekte [132], die Bildkodierung [137] sowie die Überwachung von Produktionsprozessen [158].

2.6.2. Lösung partieller Differentialgleichungen

Bereits in [22] wurde die enge Verbindung zwischen CNNs und partiellen Differentialgleichungen am Beispiel der Wärmeleitungsgleichung erläutert. Tatsächlich lässt sich die dynamische Entwicklung einer Vielzahl linearer und nichtlinearer partieller Differentialgleichungen auf einem CNN darstellen. So wurden bereits Reaktions-Diffusions-Systeme [17], Modelle zur Wellenausbreitung [23] und verschiedene nichtlineare Differentialgleichungen 2. Ordnung [113, 114, 127] auf unterschiedlichen Netzwerkstrukturen untersucht.

Für jede nicht-analytische Berechnung einer partiellen Differentialgleichung ist eine örtliche Diskretisierung des zugrundeliegenden räumlich und zeitlich kontinuierlichen Systems erforderlich. Betrachtet man die partielle Differentialgleichung nur an einer endlichen Zahl diskreter Punkte, so liefert eine Finite-Differenzen-Approximation [100] ein System gewöhnlicher Differentialgleichungen, das äquivalent zur Zellzustandsgleichung (2.4) ist.

Zwar können mit dem Chua-Yang-Modell einige Systeme, wie z. B. die Wärmeleitungsgleichung und Laplace-Gleichung, dargestellt werden, häufig ist jedoch die Erweiterung des ursprünglichen Netzwerkmodells unumgänglich. So erfordert bereits die Simulation der Wellengleichung die Verwendung mehrschichtiger Netzwerke [136] und die Simulation nichtlinearer partieller Differentialgleichungen die Verwendung nichtlinearer Kopplungsterme. Letzteres wird nachfolgend am Beispiel der viskosen Burgers-Gleichung beschrieben.

Burgers-Gleichung

Die homogene Burgers-Gleichung[6]:

$$\frac{\partial\varphi(s,t)}{\partial t} = -\frac{1}{2}\frac{\partial\varphi(s,t)^2}{\partial s} + \frac{1}{R}\frac{\partial^2\varphi(s,t)}{\partial s^2} \tag{2.34}$$

ist eine nichtlineare partielle Differentialgleichung die zur Simulation eindimensionaler, viskoser Strömungsvorgänge verwendet werden kann. In [62] wurde eine analytische Lösung der Burgers-Gleichung hergeleitet. Wie in [113] und [127] beschrieben, lässt sich die Burgers-Gleichung durch räumliche Diskretisierung auf ein System gewöhnlicher nichtlinearer Differentialgleichungen und damit auf ein PTCNN abbilden.

Durch Bilden des zentralen Differenzenquotients erhält man:

$$\frac{\partial\varphi(s,t)}{\partial s} = \frac{\varphi(s+\Delta s,t) - \varphi(s-\Delta s,t)}{2\Delta s} \tag{2.35}$$

und damit:

$$\frac{\partial\varphi^2(s,t)}{\partial s} = \frac{\varphi^2(s+\Delta s,t) - \varphi^2(s-\Delta s,t)}{2\Delta s}\,. \tag{2.36}$$

Aus der Taylor-Entwicklung folgt der Differenzenquotient 2. Ordnung:

$$\frac{\partial^2\varphi(s,t)}{\partial s^2} = \frac{\varphi(s-\Delta s,t) - 2\varphi(s,t) + \varphi(s+\Delta s,t)}{(\Delta s)^2}\,. \tag{2.37}$$

Betrachtet man nun die Funktion lediglich an diskreten Stützstellen $i \in [1, N_\mathrm{v}]$ eines regulären Gitters, so erhält man mit $\varphi(i\Delta s,t) \equiv x_i(t)$:

$$\dot{x}_i(t) = \frac{x_{i-1}(t) - 2x_i(t) + x_{i+1}(t)}{R(\Delta s)^2} - \frac{x_{i+1}^2(t) - x_{i-1}^2(t)}{4\Delta s}\,. \tag{2.38}$$

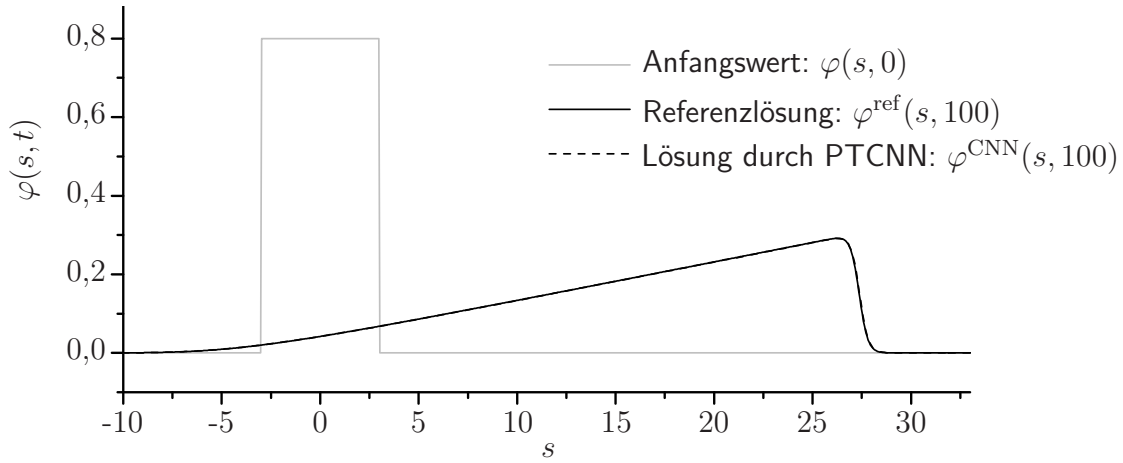
[6] Um Verwechslungen mit dem Zustand und Eingang der CNN-Zustandsgleichung zu vermeiden, wird hier entgegen der häufig verwendeten Nomenklatur $u(x,t)$ durch $\varphi(s,t)$ ersetzt.

Diese räumlich diskretisierte Form der Burgers-Gleichung lässt sich auf ein PTCNN unter der Annahme $y_{ij}(t) = x_{ij}(t)$ abbilden. Ein Koeffizientenvergleich mit (2.28) liefert das Template:

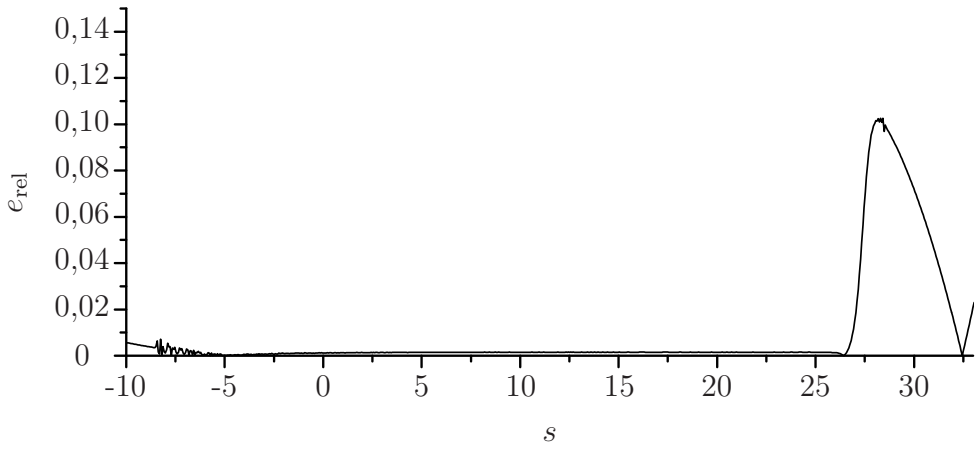
$$\mathbf{A}^{(1)} = \begin{array}{|c|c|c|}\hline \frac{1}{R(\Delta s)^2} & \frac{R(\Delta s)^2-2}{R(\Delta s)^2} & \frac{1}{R(\Delta s)^2} \\ \hline \end{array} \qquad \mathbf{A}^{(2)} = \begin{array}{|c|c|c|}\hline -\frac{1}{4\Delta s} & 0 & \frac{1}{4\Delta s} \\ \hline \end{array}$$

$$\mathbf{B}^{(0)} = 0,\ \mathbf{B}^{(1)} = 0,\ \mathbf{z} = 0\,. \tag{2.39}$$

Für $\Delta s \to 0$, entsprechend einem Netzwerk mit unendlich vielen Zellen, tritt ein Grenzübergang zum räumlich kontinuierlichen Modell auf.



(a) Vergleich von PTCNN mit Referenzlösung. Die Lösung für das PTCNN ist von der Referenzlösung überdeckt.



(b) Relativer Fehler.

Abbildung 2.5.: Berechnung der Burgers-Gleichung mit einem PTCNN.

Abbildung 2.5 zeigt das Simulationsergebnis zur Lösung der Burgers-Gleichung mit den in (2.39) gegebenen Netzwerkparametern gegenüber einer Referenzlösung $\varphi^{\mathrm{ref}}(s,t)$ zum Zeitpunkt $t = 100$ für $R = 30$. Die Referenzlösung wurde mit einem Runge-Kutta-Verfahren 4. Ordnung bei einer Schrittweite von $h = 10^{-3}$ ermittelt. Für die Berechnung mit dem PTCNN wurde ein Netzwerk mit $N_{\mathrm{v}} = 2000$ Zellen mit dem Polynomgrad $P = 2$ und Dirichlet-Randbedingungen verwendet. Die Diskretisierungsschrittweite beträgt $\Delta s = 0{,}06$; damit umfasst der Definitionsbereich für s den Bereich $[-60{,}60]$. Die Schrittweite des Euler-Verfahrens ist $h = 0{,}01$ und die Anfangsbedingung lautet:

$$\varphi(s{,}0) = \begin{cases} 0{,}8 & , -3 \leq s \leq 3 \\ 0 & , \text{sonst} \end{cases} . \tag{2.40}$$

Zum Vergleich der beiden Lösungen dient der Betrag des relativen Fehlers:

$$e_{\mathrm{rel}} = \left| \frac{\varphi^{\mathrm{CNN}}(s,t) - \varphi^{\mathrm{ref}}(s,t)}{\varphi^{\mathrm{ref}}(s,t)} \right| . \tag{2.41}$$

Der relative Fehler nimmt im dargestellten Intervall $[-10, 30]$ einen Maximalwert von etwa 10 % an und kann durch eine Verringerung von Δs und h weiter verkleinert werden. Das Beispiel zeigt, dass die Burgers-Gleichung mit einem PTCNN gelöst werden kann (siehe auch [127] und [114]). Welchen Einfluss die gerinigere Rechengenauigkeit einer praktischen Hardware-Realisierung auf die Lösung hat, wird in Kapitel 8.5.1 untersucht.