Untersuchung von Synchronisationsphänomenen in dynamischen Systemen mit Zellularen Neuronalen Netzen

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Robert Sowa

geb. in

Malapane

Bonn 2004

Anfertigung mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Referent: PD Dr. K. Lehnertz

2. Referent: Prof. Dr. K. Maier

Tag der Promotion: 29.Juni 2004

Inhaltsverzeichnis

Einleitung

Das Wort *synchron* wird häufig sowohl in der alltäglichen als auch in der wissenschaftlichen Sprache verwendet. Es setzt sich ursprünglich zusammen aus den beiden griechischen Wörtern $\sigma\acute{\nu}\nu$ (*syn*, das Selbe, Gleiche) und $\chi\rho\acute{o}\nu o\varsigma$ (*chronos*, Zeit) und bedeutet direkt übersetzt *die gemeinsame Zeit nutzen* bzw. *zur gleichen Zeit geschehen*.

Das Wort synchron, aber auch Abwandlungen wie *synchronisiert* und *Synchronisation*, beschreiben in fast allen Zweigen der Naturwissenschaften, Ingenieurwissenschaften und im alltäglichen Leben Phänomene, die sehr unterschiedlich sind, aber häufig universellen Gesetzen unterworfen sind.

Im 17. Jahrhundert erschien der Begriff Synchronisation erstmals in der naturwissenschaftlichen Literatur. Christian Huygens ([Hu1673]) beschrieb mit ihm reibungsfreie harmonische Oszillatoren (zwei Pendel, die an einem horizontalen Balken aufgehängt sind). Er beobachtete, dass die *Phasendifferenz* zwischen zwei näherungsweise harmonischen Oszillatoren nach einem gewissen Einschwingvorgang immer null wird. Huygens beschrieb mit Synchronisation eine strenge Beziehung zwischen den *Phasen* der Oszillatoren, deshalb wird dieser spezielle Fall *Phasensynchronisation* genannt.

Die verstärkte Untersuchung von nichtlinearen dynamischen Systemen seit den 1980er Jahren ([Ot93], [Sch94]) warf die Frage auf, ob Synchronisation auch in diesen Systemen gefunden bzw. beschrieben werden kann. In der Literatur über nichtlineare dynamische Systeme finden sich mehrere Synchronisationsbegriffe, die allerdings nur *Teilaspekte* beschreiben. Die einfachste Variante ist die *vollständige Synchronisation*. Sie beschreibt die Identität zweier *Zeitreihen*. Rosenblum und Kollegen ([RPK96]) führten den Begriff *lag synchronization* ein. Hier sind die Zeitreihen bis auf einen konstanten zeitlichen Versatz gleich. Unabhängig von diesen Definitionen führten Afraimovich und Kollegen ([AVR86]) die *generalized synchronization* ein. In diesem Fall sind zwei Zeitreihen synchronisiert, falls ein Funktional existiert, das sie in Beziehung zueinander setzt.

Wird das Huygenssche Konzept der Phasensynchronisation auf deterministische chaotische Systeme angewendet, so ergibt sich das Problem der Definition von Phasenvariablen aus den Zeitreihen der untersuchten Systeme. Die Nutzung der *Hilbert-Transformation* hilft bei der Definition eines solchen Phasenbegriffs. Durch die Konstruktion eines *analytischen Signals* ordnet sie jedem Zeitpunkt einer skalaren Zeitreihe eine *instantane Phase* zu. Gabor ([Ga46]) hat dieses Verfahren ursprünglich auf dem Gebiet der Kommunikationstheorie eingeführt.

Die Anwendungen des Konzepts der Phasensynchronisation sind vielfältig. Die Untersuchung von Synchronisationsphänomenen gekoppelter chaotischer Systeme spielt eine bedeutende Rolle unter anderem auf den Gebieten der Laserdynamik ([FCRL93], [RT94]),

Festkörperphysik ([PYW95]), Elektronik ([HCP94]), Biologie ([HKK95]) sowie der Nachrichtentechnik ([KP95]). Auch physiologische Daten wie z.B. Herzschlag und Atmung ([SDEAS96], [SRKA98]) wurden in letzter Zeit auf Synchronisationsphänomene hin untersucht.

Eine große Herausforderung stellt die Analyse von Synchronisationsphänomenen in *Zeitreihen hirnelektrischer Aktivität* dar. Das *Elektroenzephalogramm* (EEG) liefert die gemessene Aktivität einer Vielzahl von Neuronen des offenen dynamischen System „menschliches Gehirn“. Die Anzahl der beobachteten Freiheitsgrade übersteigt dabei die im Rahmen der Beobachtungsdauer mögliche Auflösbarkeit. Daraus resultieren Signale mit einem eher stochastischen und nichtstationären Erscheinungsbild. Es gibt aber auch pathologische Phänomene, wie z.B. epileptische Anfallsaktivität, bei denen Neuronen in Verbänden synchronisieren und in ausgeprägten Mustern über den scheinbar stochastischen Hintergrund dominieren.

Das geschilderte Phänomen der epileptischen Anfallsaktivität ist von großem Interesse. Könnte diese Aktivität vorhergesagt werden, könnte sie durch kurzfristig eingesetzte Medikation abgemildert oder sogar direkt verhindert werden. Vielfältige Ansätze zu Analyse und Vorhersage (vgl. [LL02] für einen Überblick) wurden entwickelt, u.a. auch basierend auf der Nutzung von Zellularen Neuronalen Netzen ([TKAW99], [KTW00], [LPKH02]). Es hat sich herausgestellt, dass *bivariate* Analysetechniken der *nichtlinearen Zeitreihenanalyse* eine Sensitivität und Spezifität erreichen, die mit *univariaten* Analysetechniken schwer erreicht werden können ([EMKARSFDL02], [LMKARDE03]).

Ein prominentes Beispiel für ein solches bivariates Maß ist das Synchronisationsmaß *mittlere Phasenkohärenz R*. Durch Nutzung der Hilbert-Transformation und der zirkularen Varianz auf der zu den Signalen gehörigen Phasendifferenzverteilung wurde dieses Maß von Mormann und Kollegen ([MLDE00]) entwickelt. Studien ([MAKRDEL03], [MKADLE03]) zeigten, dass Langzeit-Voranfalls-Zustände (*prä-iktal*, bis zu mehreren Stunden andauernd) anhand der zeitlichen Entwicklung von R definiert werden können. Diese zeichnen sich durch niedrige Synchronisationswerte R aus, welche sich signifikant von den Werten des anfallsfreien Intervalls (*interiktal*) unterscheiden.

Obwohl die mittlere Phasenkohärenz R und andere auf der Hilbert-Transformation basierende Maße einfach berechnet werden können (*Fast-Fourier-Transformation* (FFT), es wird nur eine Hin- und eine Rücktransformation benötigt), werden Echtzeit-Berechenbarkeitsgrenzen durch alle möglichen Kombinationen der vorhandenen Sensoren (bis zu 256 Sensoren in klinischen oder neurowissenschaftlichen Systemen) gesetzt. Um ein unhandliches Verteiltes-Rechnen-System zu vermeiden (wegen Fehleranfälligkeit der Geräte und hohem Platz- und Energieverbrauch), müssen andere Möglichkeiten und andere Architekturen für die Aufgaben der nichtlinearen Zeitreihenanalyse evaluiert werden, die eine entsprechende Rechenkapazität bieten.

Zellulare Neuronale Netze (CNN) ermöglichen eine hohe Rechengeschwindigkeit, bei gleichzeitig niedrigem Energie- und Platzverbrauch. Durch die nur lokal vorhandene Kopplung ihrer *Neuronen*, können CNN als VLSI[1] Implementationen realisiert werden und somit zur Entwicklung von miniaturisierten Analysesystemen führen.

[1] Very-Large-Scale-Integrated

Das Ziel der vorliegenden Dissertation ist die Entwicklung von CNN, die es erlauben, den Synchronisationsgrad zwischen zwei Zeitreihen möglichst optimal zu approximieren. Es wird gezeigt, dass aufwändige Algorithmen, wie die Berechnung der mittleren Phasenkohärenz R, auf solche *neuen* Architekturen übertragen werden können.

In Kapitel 1 wird zunächst ein Überblick über die unterschiedlichen Formen von Synchronisation und die Möglichkeiten der Berechnung des Synchronisationsgrades durch das Maß der mittleren Phasenkohärenz R gegeben. Dabei wird sowohl auf künstlich generierte Daten, als auch auf gemessene Daten von realen Systemen eingegangen. Kapitel 2 führt in die Konzepte der Neuronalen und Zellularen Neuronalen Netze ein. In Kapitel 3 wird auf die beiden vorhandenen Werkzeuge SCNN (universeller Software-Simulator für CNN) und das Aladdin System (schaltungstechnische VLSI-Realisation eines CNN) eingegangen. Es werden erste Voruntersuchungen am Aladdin System besprochen. Im folgenden Kapitel 4 werden CNN entwickelt, mit denen die Approximation der mittleren Phasenkohärenz R von Modellzeitreihen und intrakraniell abgeleiteten Zeitreihen hirnelektrischer Aktivität möglich ist. Es erfolgt eine genaue Auswertung, wobei auch auf Einflussfaktoren wie z.B. Toleranzen von schaltungstechnischen Realisationen im CNN eingegangen wird. Es schließt sich im Kapitel 5 eine Untersuchung der Fähigkeiten der CNN an Langzeit-EEG an. Dabei wird aber nicht nur die Langzeit-Stabilität der Approximation untersucht, sondern auch weitere Generalisierungsverhalten, wie z.B. die Approximation der Daten, die an einer anderen Elektrodenkombination abgeleitet wurden. Kapitel 6 skizziert ein mögliches Konzept eines Analysesystems von CNN zur Untersuchung der Synchronisation in EEG-Aufzeichnungen. Die Dissertation schließt mit einer Diskussion in Kapitel 7 und einer Zusammenfassung in Kapitel 8.

Kapitel 1

Synchronisation

Es ist schwierig, Synchronisation in einer einheitlichen Definition zu beschreiben. Nach Pikovsky und Kollegen ([PRK01]) bedeutet Synchronisation *die Anpassung von Frequenzen periodischer Oszillatoren aufgrund von schwachen Wechselwirkungen*. Dies entspricht der klassischen Definition von Synchronisation. In den letzten Jahren wurden weitere verschiedene Definitionen entwickelt. Trotz der unterschiedlichen Meinungen können bestimmte Sonderfälle von Synchronisationsphänomenen definiert werden, die sich durch gewisse Eigenschaften auszeichnen.

Diese Sonderfälle werden nach einer Einführung in die Theorie der dynamischen Systeme beschrieben. Es folgt die Darstellung des Synchronisationsmaßes der *mittleren Phasenkohärenz* R als ein Werkzeug, mit dem die Synchronisation gekoppelter Systeme gemessen werden kann.

1.1 Dynamische Systeme

Ist ein *dynamisches System* durch d Variablen zur Zeit t vollständig beschrieben, so kann die Dynamik durch Konstruktion eines zeitabhängigen Vektors in einem d-dimensionalen *Zustandsraum* beschrieben werden mit

$$\vec{x}(t) = (x_1(t), ..., x_d(t)) \quad \text{mit} \quad \vec{x}(t) \in \Re^d \tag{1.1}$$

Jeder Zustand ist eindeutig durch einen Punkt $\vec{x}(t)$ definiert. Die Bahn aller Punkte in der zeitlichen Entwicklung wird *Trajektorie* genannt. Existiert ein *Generator* (eine Abbildung F bzw. *Bewegungsgleichung*) zwischen allen möglichen Zuständen und der zukünftigen Entwicklung

$$\frac{\partial \vec{x}(t)}{\partial t} = F(\vec{x}(t)) \quad \text{mit} \quad f : \Re^d \mapsto \Re^d \tag{1.2}$$

so wird dieses dynamische System als *deterministisch* definiert.

Lineare (*nichtlineare*) Dynamiken werden durch die Linearität (Nichtlinearität) von F definiert. Für *stationäre* Dynamiken gilt $F \neq F(t)$, d.h. F ist nicht explizit von der Zeit abhängig. *Konservative* Dynamiken genügen $divF = 0$, d.h. die Zustandsraumvolumina bleiben im Fortlauf der Zeit erhalten. Für *dissipative* Dynamiken gilt $divF < 0$, d.h. die

Zustandsraumvolumina kontrahieren auf Strukturen (*Attraktoren*) und verbleiben dort. *Stetige* Dynamiken genügen $\vec{x}(t) \to \vec{x}(t)^* \longrightarrow F(\vec{x}(t)) \to F(\vec{x}(t)^*)$, d.h. benachbarte Segmente sind zu einander ausgerichtet und es entsteht ein *deterministischer lokaler Fluss*. Für *stochastische* dynamische Systeme existiert kein Zusammenhang zwischen Gegenwart und Zukunft, so dass sich die Trajektorie selbst schneiden kann.

Durch nichtlineare Bewegungsgleichungen können *chaotische* Dynamiken entstehen. Ein Beispiel dafür ist das Rössler-System (siehe Anhang D). Die Darstellung des dreidimensionalen Zustandsraums stellt einen komplizierten Attraktor dar, welcher durch den Generator, drei gekoppelte gewöhnliche Differentialgleichungen erster Ordnung, beschrieben wird. Solch ein *seltsamer Attraktor* ist eine Eigenschaft von chaotischen Dynamiken.

Die *Zeitreihenanalyse* bietet sich als Instrumentarium zur Analyse dynamischer Systeme an. Mit ihr können Verbindungen zwischen der Theorie bekannter dynamischer Systeme und den gemessenen realen Daten von Systemen unbekannter Dynamik hergestellt werden. Aufzeichnungen von *Observablen* realer dynamischer Systeme sind meist mit zusätzlichen Störsignalen durchsetzt. Diese Störsignale können oft als Rauschen, als Artefakte oder als die hohe Komplexität des zugrunde liegenden Systems wahrgenommen werden. Die Entwicklung der *nichtlinearen Zeitreihenanalyse* ([Ot93], [Sch94],[KS97]) ermöglichte es, die Theorie nichtlinearer dynamischer Systeme auf reale Daten anzuwenden.

1.1.1 Vollständige Synchronisation

Zwei Systeme X und Y gelten als vollständig synchronisiert, wenn die Zustandsvariablen $\vec{x}(t)$ und $\vec{y}(t)$ im Grenzwert unendlicher Zeit t vollständig identisch werden. Dann gilt

$$\lim_{t\to\infty}[\vec{x}(t) - \vec{y}(t)] = \vec{0} \tag{1.3}$$

Dies stellt einen Spezialfall dar.

1.1.2 Lag synchronization

Rosenblum und Kollegen ([RPK97]) führten den Begriff *lag synchronization* ein. Zwischen zwei Systemen X und Y herrscht lag synchronization, wenn die Zustandsvariablen $\vec{x}(t)$ und $\vec{y}(t)$ bis auf eine zeitliche Verschiebung τ identisch sind. Es gilt also

$$\vec{x}(t+\tau) = \vec{y}(t) \tag{1.4}$$

Für den Grenzwert $\tau \to 0$ geschieht der Übergang zur vollständigen Synchronisation.

Der Vollständigkeit halber sei darauf hingewiesen, dass u.a. mit der *maximalen linearen Kreuzkorrelation* C_{max} der Grad der lag synchronization gemessen werden kann. Wenn $C_{max} = 1$ gilt, erfüllen beide Systeme die Eigenschaft der lag synchronization (siehe Gleichung 1.4); liegt der Wert nahe bei Null, sind sie unsynchronisiert. Dabei gilt für C_{max} die folgende Gleichung:

$$C_{max} = \max_{\tau}\{C(x,y)(\tau)\} \tag{1.5}$$

mit

$$C(x,y)(\tau) = \left| \frac{corr(x,y)(\tau)}{\sqrt{corr(x,x)(0)corr(y,y)(0)}} \right| \tag{1.6}$$

als normierte Kreuzkorrelations-Funktion und

$$corr(x,y)(\tau) = \int_{-\infty}^{+\infty} x(t+\tau)y(t)dt \tag{1.7}$$

als lineare Kreuzkorrelations-Funktion mit zeitlicher Verschiebung τ.

Für weitere Untersuchungen an Modellsystemen und Zeitreihen hirnelektrischer Aktivität mit der maximalen linearen Kreuzkorrelation C_{max} sei auf [Mor98], [MKADLE03] und [MAKRDEL03] verwiesen.

1.1.3 Phasensynchronisation

Phasensynchronisation ist die älteste Definition von Synchronisation und wurde von Christian Huygens ([Hu1673]) zum ersten Mal wissenschaftlich beschrieben. Er beobachtete, dass die *Phasendifferenz* zwischen zwei näherungsweise harmonischen Oszillatoren, zwei Pendeln, die an einem horizontalen Balken aufgehängt sind, nach einem gewissen Einschwingvorgang immer null wird.

Rosenblum und Kollegen ([RPK96]) wiesen dieses Verhalten auch bei nichtlinearen und sogar chaotischen Zeitreihen nach. Dieses Konzept wurde auch auf biologische Zeitreihen wie z.B. MEG (Magnetoenzephalographie) und EMG (Elektromyographie) Daten von Parkinson Patienten ([TRWKPVSF98]) und EEG Daten von Epilepsiepatienten ([MLDE00], [MAKRDEL03]) angewendet. Üblicherweise wird Phasensynchronisation (mit φ =Phasenvariable des Systems) folgendermaßen definiert:

$$n\varphi_x(t) - m\varphi_y(t) = const \quad \text{mit } n \text{ und } m \text{ ganze Zahlen} \tag{1.8}$$

Erweitert auf gekoppelte chaotische Systeme ergibt sich eine Abschwächung von Gleichung 1.8, die lediglich eine *Beschränktheit* der Phasendifferenz zweier Systeme

$$n\varphi_x(t) - m\varphi_y(t) < const \tag{1.9}$$

oder als weitere Einschränkung nur die *Gleichheit der über die Zeit gemittelten Frequenzen* fordert

$$n \left\langle \frac{d}{dt}\varphi_x(t) \right\rangle = m \left\langle \frac{d}{dt}\varphi_y(t) \right\rangle \tag{1.10}$$

1.1.4 Generalized synchronization

Der Vollständigkeit halber wird auf die letzte gebräuchliche Art von Synchronisation eingegangen. Afraimovich und Kollegen ([AVR86]) führten die *generalized synchronization* ein.

Zwischen zwei Systemen X und Y herrscht generalized synchronization, wenn zwischen beiden Systemen ein Funktional ψ existiert, für das die folgende Bedingung gilt:

$$\vec{y}(t) = \psi[\vec{x}(t)] \tag{1.11}$$

Dabei werden die Eigenschaften von ψ (Glattheit, Differenzierbarkeit, ...) in der Literatur unterschiedlich definiert und kontrovers diskutiert (z.B. [PCJMH97], [AGLE99]).

1.2 Mittlere Phasenkohärenz R

Das Synchronisationsmaß der *mittleren Phasenkohärenz R* wurde von Mormann und Kollegen ([MLDE00], [Mor98], [Mor03]) vorgeschlagen und erfolgreich zur Klassifikation von Modelldaten und Zeitreihen hirnelektrischer Aktivität angewendet.

Um Aussagen über den Synchronisationszustand zweier Systeme und damit über ihre Phasen treffen zu können, muss ein Phasenbegriff sinnvoll definiert werden. Zunächst wird mit Hilfe der *Hilbert-Transformation* ein Verfahren dargestellt, mit dem einer beliebigen Zeitreihe (bzw. einem abgetasteten Signal) $s(t)$ eine Phase $\varphi(t)$ zugeordnet werden kann. Dann wird das hierauf basierende Synchronisationsmaß der mittleren Phasenkohärenz R vorgestellt.

Hilbert-Transformation und instantane Phase

Traditionell wird die Phase einer Variablen eines physikalischen Systems mit Hilfe einer periodischen Funktion, wie z.B.

$$s(t) = A(t)\cos(\varphi(t)) \tag{1.12}$$

dargestellt. Dabei wird $A(t)$ als Amplitude und $\varphi(t)$ als Phase bezeichnet. Mit Hilfe der Darstellung durch komplexe Zahlen lassen sich die Amplitude und Phase folgendermaßen bestimmen:

$$z(t) = A(t)e^{i\varphi(t)} = A(t)[\cos(\varphi(t)) + i\sin(\varphi(t))] \tag{1.13}$$

mit
$$s(t) = Re(z(t)) \tag{1.14}$$

und
$$A(t) = \sqrt{[Re(z(t))]^2 + [Im(z(t))]^2} \tag{1.15}$$

und
$$\varphi(t) = arctan(\tfrac{Im(z(t))}{Re(z(t))}) \tag{1.16}$$

Der Imaginärteil von $z(t)$ ist dabei gegenüber dem Realteil um $\frac{\pi}{2}$ verschoben.

Sei $s(t)$ ein beliebiges reales Signal, so gilt nach Durchführung der *Hilbert-Transformation*:

$$\tilde{s}(t) = s(t) \otimes \frac{1}{\pi t} = \frac{1}{\pi}\wp \int_{-\infty}^{+\infty} \frac{s(\tau)}{t-\tau} d\tau \tag{1.17}$$

$\tilde{s}$ ist die Hilbert-Transformierte des Signals $s(t)$, $\wp$ ist der Cauchysche Hauptwert des Integrals ([BS95]).

Über den Faltungssatz, mit FT (FT^{-1}) als Fourier-Transformation (inverse FT),

$$f(t) \otimes g(t) = FT^{-1}[FT[f(t)]FT[g(t)]] \tag{1.18}$$

kann $\tilde{s}(t)$ folgendermaßen geschrieben werden

$$\tilde{s}(t) = FT^{-1}[FT[s(t)](-i)sign(\omega)] \tag{1.19}$$

Das gesamte Fourierspektrum des Ausgangssignals erfährt also durch die Hilbert-Transformation eine Phasenverschiebung um $\frac{\pi}{2}$, während das Leistungsspektrum unverändert bleibt.

Damit ist das sogenannte *analytische Signal* ([Ga46])

$$z(t) = s(t) + i\tilde{s}(t) \tag{1.20}$$

als Verallgemeinerung der Gleichung 1.13 zu sehen.

Entsprechend gilt dann für die *instantane Amplitude*

$$A(t) = \sqrt{[s(t)]^2 + [\tilde{s}(t)]^2} \tag{1.21}$$

und für die *instantane Phase*

$$\varphi(t) = arctan(\frac{\tilde{s}(t)}{s(t)}) \tag{1.22}$$

Zu beachten ist, dass der Arcus Tangens in Gleichung 1.22 auf den Wertebereich $[0, 2\pi]$ beschränkt ist. Damit führt die ansonsten stetige Funktion $\varphi(t)$ an den Intervallgrenzen einen Sprung um 2π durch. Um die Differenzierbarkeit dieser Funktion zu erreichen, wird die Phase durch Anhebung oder Absenkung um 2π an den jeweiligen Sprungstellen *entfaltet*, damit sie als stetige Funktion $\varphi(t)^*$ abgeleitet werden kann. Die zeitliche Ableitung der entfalteten instantanen Phase wird als *instantane Frequenz* bezeichnet:

$$\omega(t) = \frac{d}{dt}\varphi^*(t) \tag{1.23}$$

Die Hilbert-Transformation ist hier ein Werkzeug, das einem *beliebigen* realen Signal $s(t)$ zu jedem Zeitpunkt eine instantane Phase und Amplitude zuordnet. Dies geschieht in Übereinstimmung mit dem üblichen Phasenbegriff der Physik gemäß Gleichung 1.12.

Eigenschaften und Probleme

Neben der Notwendigkeit der Entfaltung der instantanen Phase, um die instantane Frequenz problemlos berechnen zu können, gibt es noch weitere Eigenschaften, die bei Verwendung der Hilbert-Transformation zu beachten sind. Im Folgenden werden die einzelnen Eigenschaften kurz dargestellt (vgl. auch [Mor98]). Experimentell abgeleitete Signale liegen als endliche und diskrete Zeitreihen vor. Die Lösungen nichtlinearer Differentialgleichungssysteme sind hingegen kontinuierlich. Auf diesem Unterschied basieren diese folgenden Eigenschaften und Probleme.

Fehlen von Nulldurchgängen Gleichung 1.12 enthält keine additive Konstante (Offset) und ist darauf angelegt, Schwingungen um die Nulllinie zu beschreiben. Wird also ein Signal, welches Nulldurchgänge hat, Hilbert-transformiert, wie z.B. im Fall des monofrequenten Signals $s(t) = A\cos(\omega_0 t) + C$ mit $C > \frac{A}{2}$, so stimmt die errechnete instantane Frequenz nicht mit ω_0 überein.

Der nichtschwingende Anteil bei oszillierenden Systemen (Frequenzkomponente $\omega = 0$) ist eine Ursache für den Unterschied zwischen ω_0 und der instantanen Frequenz. Deshalb wird zunächst ein Mittelwertabgleich durchgeführt. Von allen Abtastpunkten der Zeitreihe wird der Mittelwert aller Amplitudenwerte der Zeitreihe subtrahiert und somit eine neue Zeitreihe erzeugt. Dies entspricht dem Nullsetzen der Frequenzkomponente $\omega = 0$.

Ein erneutes Anwenden der Hilbert-Transformation und Berechnung der instantanen Frequenz liefert eine wesentlich bessere Übereinstimmung zu ω_0. Allerdings gibt es immer noch Abweichungen an den Rändern sowie Oszillationen um ω_0. Diese Effekte werden in den folgenden Abschnitten erläutert.

Fehlen der periodischen Fortsetzbarkeit Für die Berechnung der Hilbert-Transformation wird die diskrete *Fourier-Transformation* (DFT) genutzt. Für diskrete Signale endlicher Dauer kann die diskrete DFT nicht problemlos eingesetzt werden, da sie unendlich viele Abtastwerte voraussetzt. Um dennoch die DFT nutzen zu können, muss das Signal an den Randwerten periodisch fortsetzbar sein (Stetigkeit beim Übergang vom letzten zum ersten Punkt). Dadurch wird eine Sprungstelle vermieden, für die im Spektrum viele hochfrequente Anteile erforderlich wären, die im ursprünglichen Signal nicht vorhanden sind.

Dieses Problem kann umgangen werden, indem die Technik des *tapering* genutzt wird. D.h. das Signal wird an den Rändern mit einer Fensterfunktion ([Ha78]) multipliziert. Eine mögliche Fensterfunktion ist das *Hanning-Fenster*. Hier wird das Signal mit der linken und rechten Hälfte des positiven Anteils einer Cosinus-Halbwelle multipliziert. Allerdings ist zu beachten, dass nach dem tapering die berechnete Hilbert-Transformierte nicht mehr als Hilbert-Transformierte des ursprünglichen Signals angesehen werden darf.

Durch Nutzung der maximalen Länge der Cosinus-Halbwelle stimmt ω_0 mit der berechneten Frequenz mit Ausnahme der Ränder gut überein.

Randeffekte aufgrund endlicher Datenlängen Nach Gleichung 1.17 muss weiterhin beachtet werden, dass für die Hilbert-Transformation die gesamte (unendliche) Vergangenheit und Zukunft des Signals bekannt sein müssen. Aufgrund der Faltung mit $\frac{1}{\pi t}$ werden die Beiträge zum Integral mit wachsendem Abstand zu einem beliebigen Zeitpunkt t_0 schnell vernachlässigbar. In Signalen von endlicher Dauer macht sich der auftretende Fehler nur an den Rändern der Hilbert-Transformierten bemerkbar. Daher kommt es zu den Randeffekten bei der instantanen Phase.

Dieses Problem kann durch Verwerfen von 10% der Datenpunkte an beiden Rändern der instantanen Phase gelöst werden. Durch Einführen eines solchen Überlagerung von 20% wird eine sehr gute Übereinstimmung von ω_0 und der errechneten instantanen Phase erreicht.

Signallänge Nach dem *Abtasttheorem* wird die maximal auflösbare Frequenz (*Nyquist-Frequenz*) eines diskreten Signals durch die halbe Abtastrate ω_s bestimmt:

$$\omega_{max} = \frac{\omega_s}{2} \tag{1.24}$$

Nach [RK98] sollte die Abtastrate so gewählt werden, dass mindestens 20 Abtastwerte pro Periode vorhanden sind. In praktischen Anwendungen muss ein Mittelweg zwischen der verlangten Stationarität der Zeitreihensegmenten und dem Rechenaufwand zur Berechnung der Hilbert-Transformierten gefunden werden (bei N Datenpunkten ist die Berechnungsdauer proportional zu $N \log(N)$).

Phasensynchronisationsmaß R

Nachdem mit der Hilbert-Transformation und der instantanen Phase eine Darstellungsform beschrieben wurde, mit der jeder beliebigen skalaren jedoch oszillierenden Zeitreihe $s(t)$ eindeutig eine Phase $\varphi(t)$ und Amplitude $A(t)$ zugeordnet werden kann, kann nun das Phasensynchronisationsmaß R definiert werden.

Es liegt nahe, den zirkulären Charakter der Phase zu berücksichtigen ([Ma72]). Mit Hilfe der zirkulären Statistik können für eine Winkelverteilung statistische Momente wie z.B. die Hauptrichtung und die zirkuläre Varianz bestimmt werden. Um dies für die instantane Phase durchführen zu können, werden die Differenzen der instantanen Phasen der beiden Zeitreihen

$$\Delta\varphi_j = \varphi_2(j\Delta t) - \varphi_1(j\Delta t) \tag{1.25}$$

auf dem Einheitskreis in der komplexen Zahlenebene abgebildet ($j \in [0, N-1]$ mit N als Anzahl der Datenpunkte). Mit

$$z_j = \cos(\Delta\varphi_j) + i\sin(\Delta\varphi_j) = e^{\Delta\varphi_j} \tag{1.26}$$

wird jedem Phasendifferenzwert $\Delta\varphi_j$ eine komplexe Zahl z_j zugeordnet. Mit dem Mittelwert dieser Zahlen

$$Z = \frac{1}{N}\sum_{j=0}^{N-1} z_j \tag{1.27}$$

läßt sich die *mittlere Phasendifferenz*

$$\overline{\Delta\varphi} = arctan\left(\frac{Im(Z)}{Re(Z)}\right) \tag{1.28}$$

und die *mittlere Phasenkohärenz*

$$R = |Z| = \sqrt{[Re(Z)]^2 + [Im(Z)]^2} = 1 - CV \tag{1.29}$$

bestimmen. Dabei gibt CV die *zirkuläre Varianz* an.

R ist auf $[0, 1]$ beschränkt. Für den Fall der vollständigen Synchronisation (siehe Gleichung 1.8) nimmt R den Wert 1 an, für eine Gleichverteilung der Phasendifferenzen (zu erwarten bei nicht synchronisierten Zeitreihen) den Wert 0.

Im Folgenden werden Anwendungsbeispiele der mittleren Phasenkohärenz R vorgestellt. Das erste Beispiel ist eine kurze Darstellung an zwei gekoppelten nichtlinearen Modellsystemen. Das zweite Beispiel beschäftigt sich mit Zeitreihen *hirnelektrischer Aktvität*, die durch *intrakranielle Aufzeichnungen* in der prächirurgischen Epilepsiediagnostik gewonnen wurden.

1.3 Synchronisation in Modellsystemen

Als Beispiel einer *nichtlinearen* Dynamik wurden zwei gekoppelte Rösslersysteme (siehe Anhang D) mit dem Phasenkohärenzmaß R untersucht. Für weitere Untersuchungen, auch an anderen Modellsystemen, sei auf [Mor98] und [Mor03] verwiesen.

Im Folgenden werden nur die x-Komponenten zweier gekoppelter Rössler-Systeme betrachtet. Die Betrachtung der y- und z-Komponenten würde keine weiteren Erkenntnisse bringen. Die Daten wurden wie in Kapitel 1.2 beschrieben vorverarbeitet, um unerwünschte Seiteneffekte zu vermeiden.

Es wurden 300 Zeitreihensegmentpaare mit je 4096 Datenpunkten erzeugt. Die Kopplungsstärke ϵ wurde linear für jeden Datenpunkt von $\epsilon = 0$ um $\Delta\epsilon = 0.04/(300 * 4096)$ erhöht. Danach wurde R für die einzelnen Zeitreihensegmentpaaren berechnet.

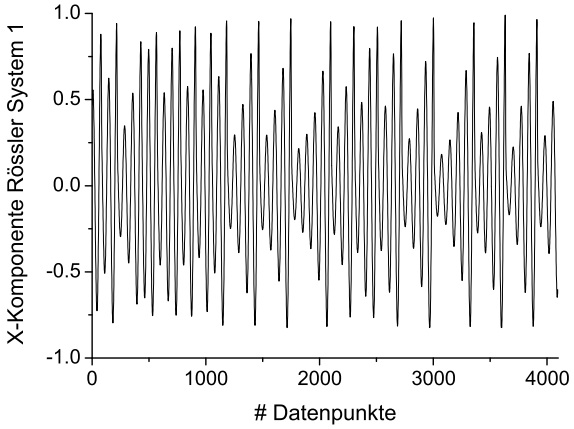
Die Abbildungen 1.1 a.) bis d.) stellen beispielhaft jeweils Zeitreihensegmente der x-Komponenten der beiden Rösslersysteme dar. Aus den oberen beiden Zeireihen errechnete sich ein Phasenkohärenz-Wert von 0.3 ($\epsilon = 0.011$), aus den unteren beidenZeitreihen ein Wert von 0.9 ($\epsilon = 0.036$).

Die Abbildungen 1.2 a.) und b.) stellen R in Abhängigkeit von der Kopplungsstärke dar. Das Phasenkohärenzmaß R folgt der Steigerung der Kopplungsstärke und erreicht bei einer Kopplung von 0.04 hohe Werte nahe 1. Werden die Signale mit weißem Rauschen mit einem niedrigen *Signal-zu-Rausch-Verhältniss* (SNR) (für genauere Betrachtungen sei auf [Mor98] und [MLDE00] verwiesen) versetzt (vgl. 1.2 b.)), ist der Anstieg immer noch deutlich zu erkennen. Damit qualifiziert sich R als robustes Maß, das auch auf reale Daten angewendet werden kann, die mit Rauschen versetzt sind.

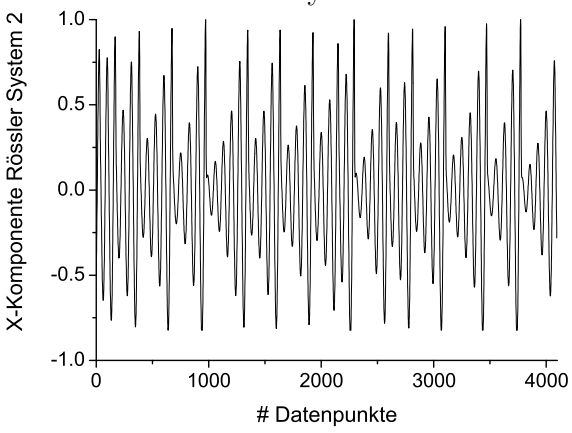
1.4 Synchronisation in Zeitreihen hirnelektrischer Aktivität

Die Anwendung der nichtlinearen Zeitreihenanalyse auf physiologische Daten wie z.B. Herzschlag und Atmung ([SDEAS96] und [SRKA98]) erschließt neue Möglichkeiten, die Dynamik der zugrundeliegenden Prozesse zu analysieren. Die Zeitreihen sind allerdings nicht mehr als ideale Daten anzusehen, sondern werden mit einer endlichen Aufzeichnungsdauer und endlichen Abtastfrequenz aufgezeichnet.

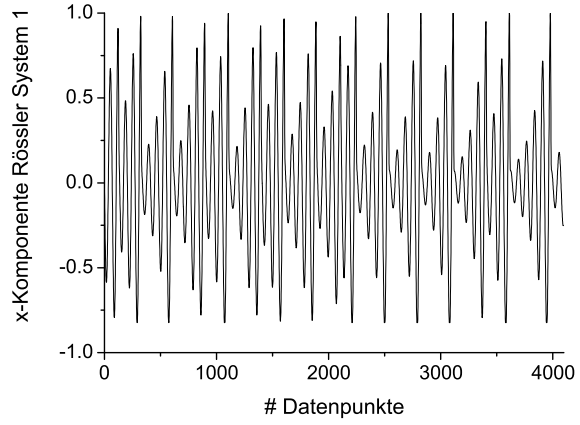
a.) Zeitreihe der x-Komponente des ersten Rösslersystem



b.) Zeitreihe der x-Komponente des zweiten Rösslersystem



c.) Zeitreihe der x-Komponente des ersten Rösslersystem



d.) Zeitreihe der x-Komponente des zweiten Rösslersystem

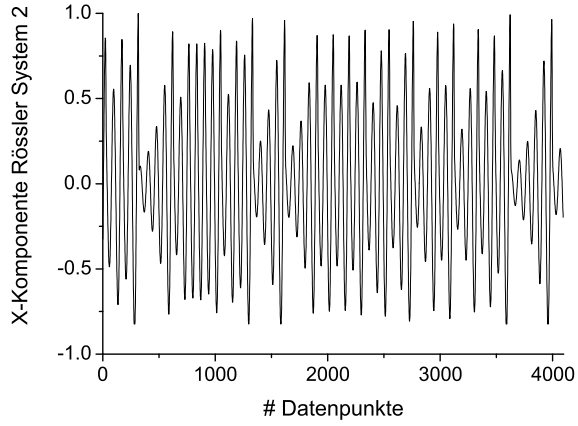


Abbildung 1.1: Die Zeitreihen a.) und b.) entsprechen einem niedrigen Phasenkohärenz Wert R von 0.3 ($\epsilon = 0.011$), die Zeitreihen c.) und d.) einem hohen von 0.9 ($\epsilon = 0.036$).

Dabei wird versucht, Einblick in die zugrundeliegende Dynamik zu gewinnen, um einerseits Aussagen über den aktuellen Zustand des Systems, aber auch Vorhersagen über künftige Zustände geben zu können. Eine große Herausforderung stellt dabei die Analyse von Synchronisationsphänomenen in Zeitreihen hirnelektrischer Aktivität in der prächirurgischen Epilepsiediagnostik dar.

1.4.1 Epilepsie

Die Krankheit Epilepsie wird durch plötzliche und wiederkehrende Funktionsstörungen (*Anfälle*) des Gehirns charakterisiert. Diese Anfälle entsprechen einer starken synchronen Aktivität der Neuronen des Gehirns. Epilepsien können in zwei Hauptklassen unterteilt werden. Gehen die Anfälle von einem abgegrenzten Areal (*Fokus*) aus, so wird von einer

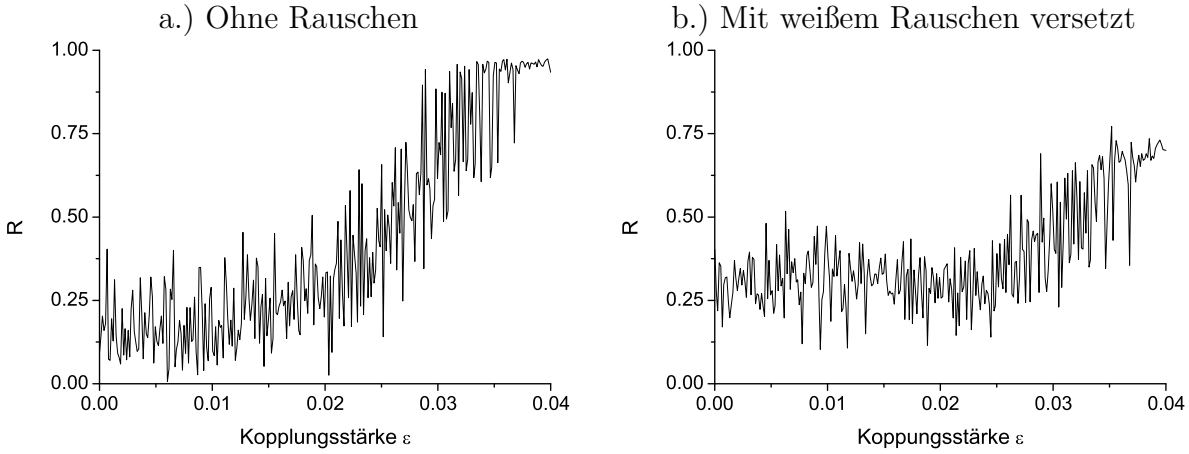


Abbildung 1.2: Entwicklung von R bei Steigerung des Kopplungfaktors ϵ bei den x-Komponenten zweier Rössler-Systeme (siehe Anhang D ohne Rauschen (a.)), mit weißem Rauschen (b.), Rausch-zu-Signal-Verhältnis 75%))

partiellen oder *fokalen* Epilepsie gesprochen. Wird allerdings fast das gesamte Gehirn bei Anfallsbeginn vereinnahmt, handelt es sich um eine *generalisierte* Epilepsie.

Ungefähr ein Prozent der Bevölkerung leidet an Epilepsie. Etwa zwei Dritteln der betroffenen Menschen kann durch anfallshemmende Medikamente geholfen werden. Weiteren acht Prozent kann durch operative Eingriffe geholfen werden. Den Übrigen kann durch keine bisher bekannte Therapie ausreichend geholfen werden.

Im Fall einer fokalen Epilepsie erfordert eine erfolgreiche chirurgische Behandlung die exakte Lokalisierung des epileptischen Fokus und die genaue Abgrenzung zu weiteren wichtigen Bereichen im Gehirn. Dafür werden verschiedene elektro physiologische Methoden genutzt ([EP97]) und durch bildgebende Verfahren ergänzt. Für die exakte Lokalisierung muss die hirnelektrischen Aktivität während eines Anfälls registriert werden. Diese Aufzeichnungen (*Elektroenzephalogramme*) können auf dem Skalp oder sogar direkt aus verschiedenen Gehirnstrukturen durchgeführt werden. Dabei zeichnet jede Elektrode die Aktivität einer Vielzahl von Neuronen auf. Da das Auftreten eines Anfalls bisher nicht vorhersehbar ist, können diese Aufzeichnungen durchaus mehrere Tage lang dauern und sind nicht ungefährlich für den Patienten. Deshalb stellt sich die Frage, ob eine Lokalisierung des epileptischen Fikus schon durch die Analyse der anfallsfreien (*interiktalen*) Zeiträume möglich ist. Diese Fragestellung wird u.a. in [LE95], [MLDE00] und [AGLE99] behandelt.

Eine weitere Fragestellung ist die Möglichkeit der Vorhersage der Anfälle. Könnte die Plötzlichkeit eines eintreffenden Anfalls durch eine Vorhersage entschärft werden, so könnten sich die therapeutischen Möglichkeiten drastisch wandeln ([El01]). Einerseits könnte z.B. ein einfaches Warnsystem die Patienten vor den Folgen eines Anfalls wie z.B. einem Sturz schützen, andererseits könnten Langzeit-Medikamententherapien durch eine kurzfristige, bedarfsgerechte Medikation ersetzt werden. Solch ein hypothetischer Zustand des Gehrins vor Anfällen (*prä-iktaler* Zustand) ist in neuerer Zeit Gegenstand zahlreicher Publikationen (Überblick in [LL02]).

1.4.2 Das Elektroenzephalogramm

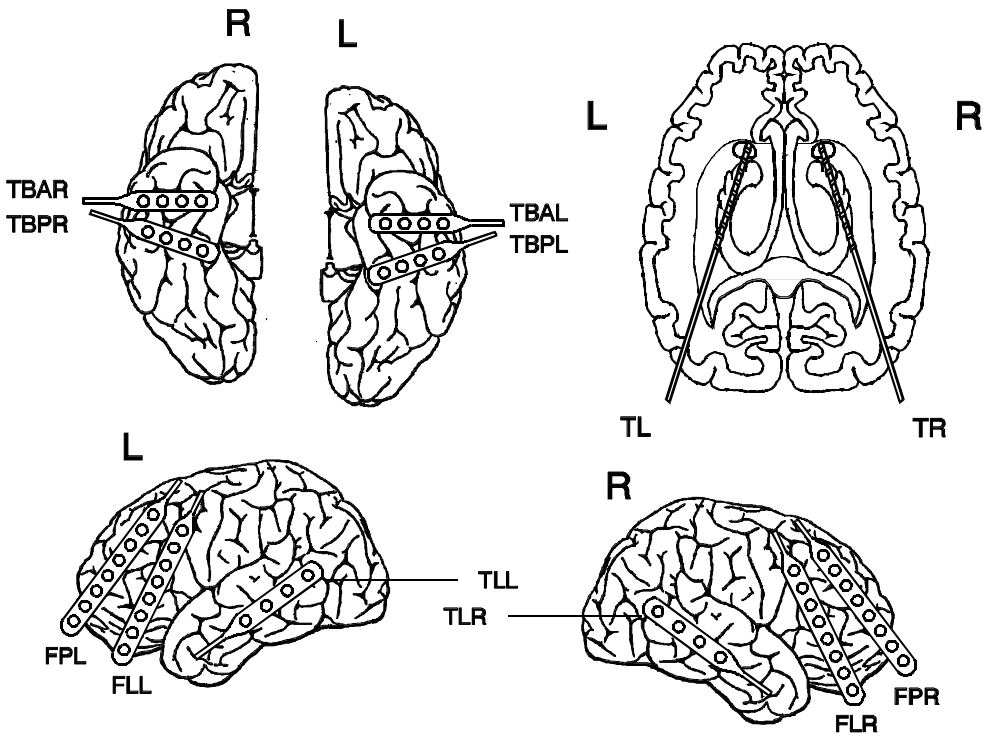


Abbildung 1.3: Implantationsschema zur Ableitung des ECoG (an den Elektroden TBAR, TBAL, TBPR, TBPL, TLR, TLL, FPL, FLL und FPR) und des SEEG (an den Elektroden TL und TR). Die einzelnen Kontakte , die durch Kreise dargestellt sind (bei TL und TR als Ringe um die Elektrode), werden durchnummeriert, wobei die höchste Zahl sich jeweils am schmalen Ende der Elektrode befindet (bei TL und TR sind die Kontakte TL10 und TR10 jeweils zur Bezeichnung TL/TR hin zu finden, an den beiden Spitzen finden sich jeweils die Kontakte TL01 und TR01).

In der Klinik für Epileptologie des Universitätsklinikums Bonn werden Epilepsiepatienten im Rahmen der prächirurgischen Epilepsiediagnostik intrakranielle Elektroden zur Ableitung der hirnelektrischen Aktivität implantiert. Die intrakranielle Ableitung hat gegenüber der Oberflächen-Ableitungen auf dem Kopf den Vorteil, dass ein besseres Signal-Rausch-Verhältnis und eine bessere räumliche Spezifität und Sensitivität besteht. Abbildung 1.3 zeigt ein typisches Implantationsschema. Hier wird zwischen dem *Electrocortigogramm* (*ECoG*, die Elektroden liegen subdural dem Cortex auf) und dem *Stereo-Elektroenzephalogramm* (*SEEG*, die Elektroden werden in bestimmten Hirnstrukturen, z.B. Hyppokampus, positioniert) unterschieden.

Ableitungen sind mit bis zu 128 Kanälen möglich. Die Abtastfrequenz der in der vorliegenden Arbeit analysierten Zeitreihen beträgt 173.61 Hz mit einer Auflösung von 12 Bit (bzw. 200 Hz und 16 Bit) (Frequenzbereich 0.03 bis 85 Hz, Tiefpaß mit einer Steilheit von 12 dB/oct). In der vorliegenden Arbeit werden nur von den *Tiefenelektroden TL* und *TR*

abgeleitete SEEG-Signale analysiert. Dabei werden nur die Zeitreihenpaare der benachbarten Elektroden betrachtet. Dies führt zu 36 Kanalkombinationen. Werden symmetrische Synchronisationsmaße wie die mittlere Phasenkohärenz R verwendet, so verbleiben 18 Kanalkombinationen. Auf beiden Elektroden TL und TR sind 10 zylindrische Messsonden, mit einer Länge von 2.5mm, einem Durchmesser von 1mm und einem Abstand von 4mm zueinander angebracht. Die Abbildung 1.4 zeigt SEEG-Ableitungen aus dem anfallsfreien Zustand (a.)) und zu Beginn eines epileptischen Anfalls (b.)).

Im Folgenden wird nur noch allgemein vom EEG gesprochen. Der Vollständigkeit halber sei erwähnt, dass der Zeitraum um den Anfall (*iktus*) herum, also vor (*prä-iktal*), während (*iktal*) und nach dem Anfall (*post-iktal*), als *peri-iktaler* Zustand bezeichnet wird.

a.) Interiktales SEEG b.) Iktales SEEG

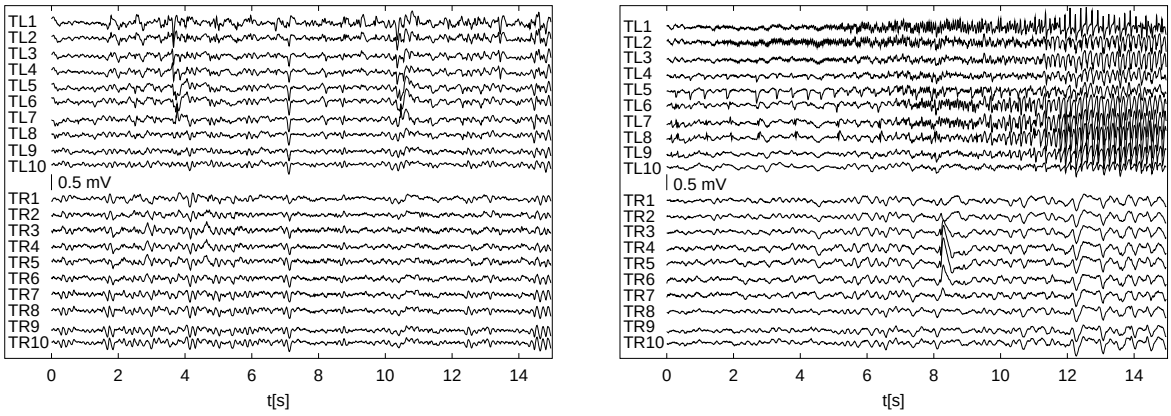


Abbildung 1.4: Die dargestellten Zeitreihen, aus den interiktalen (links) und iktalen (rechts) Zeiträumen wurden an den Messsonden TL1 bis TL10 und TR1 bis TR10 abgeleitet.

1.4.3 Zeitliche Entwicklung von charakterisierenden Kenngrößen

Im Hinblick auf eine Vorhersage von epileptischen Anfällen ist es interessant, die zeitliche Entwicklung von EEG charakterisierenden Maße zu betrachten, um Veränderungen in den Dynamiken der verschiedenen Gehirnareale während oder auch vor einem Anfall zu entdecken. In Untersuchungen, die z.B. das zeitliche Verhalten der Korrelationsdimension ([LE98]) oder Interpendenzen zwischen Gehirnarealen ([MLDE00], [MAQBCRV98], [AGLE99]) analysierten, war es in bestimmten Fällen möglich, dem Anfall vorhergehende Veränderungen in der dem EEG zugrundeliegenden Dynamik festzustellen. Damit konnte ein prä-iktaler Zustand definiert werden, der bis zu mehrere Stunden vor dem Anfall beginnen kann.

Die Abbildung 1.5 zeigt die zeitliche Entwicklung des Synchronisationsmaßes mittlere Phasenkohärenz R für eine feste Elektrodenkombination beispielhaft für einen interiktalen und peri-iktalen Datensatz eines Patienten.

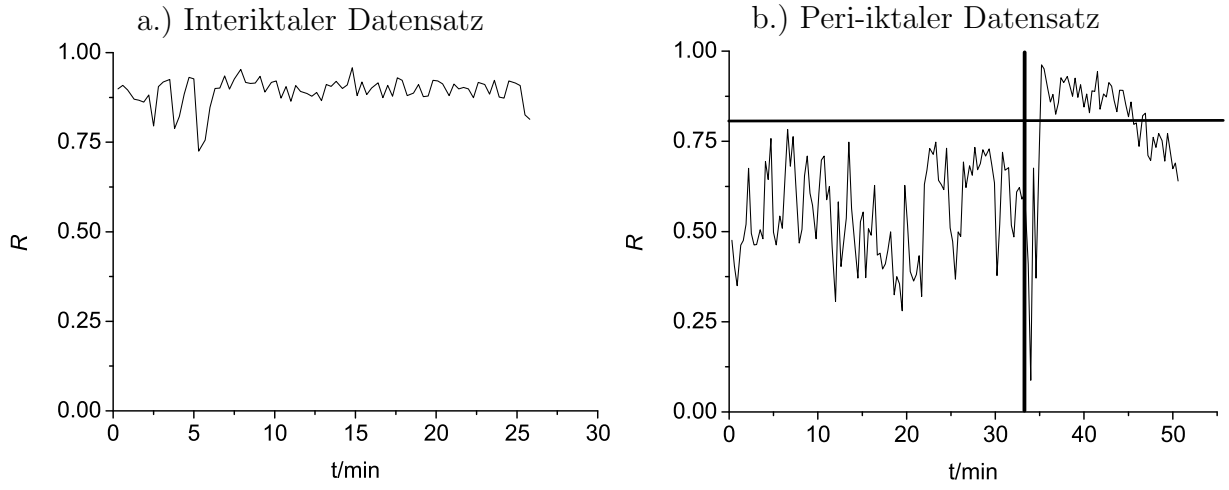


Abbildung 1.5: Zeitliche Entwicklung der mittleren Phasenkohärenz R bei einem interiktalen und einem peri-iktalen Datensatz. In b.) ist der elektrische Anfallsbeginn durch eine senkrechte Linie und der Mittelwert von R, berechnet aus allen analysierten interiktalen Abschnitten, durch eine waagerechte Linie dargestellt.

Die Abbildung 1.5 a.) zeigt eine typische zeitliche Entwicklung von R für einen interiktalen Datensatz. Es dominieren hohe R Werte aus dem Bereich $[0.8, 0.9]$. Abbildung 1.5 b.) stellt die zeitliche Entwicklung von R für einen peri-iktalen Datensatz dar. Vor dem durch eine senkrechte Linie markierten Anfall ist ein Absinken von R zu erkennen ($R \in [0.35, 0.75]$) an. Diese Werte sind deutlich geringer als die der R Werte aus den interiktalen Bereichen. Das wird durch die waagerechte Linie, die den Mittelwert von R aller zur Verfügung stehenden interiktalen Abschnitte darstellt, unterstrichen. Im Anfallzeitraum steigt R wieder an und übertrifft sogar das durchschnittliche Niveau der interiktalen Bereiche.

In [MKADLE03] und [MAKRDEL03] berichteten Mormann und Kollegen von Abnahmen der Synchronisation vor epileptischen Anfällen. Dieser Verlust der Synchronisation konnten in allen peri-iktalen Datensätzen der untersuchten 17 bzw. 18 Patienten in einigen wenigen der 18 möglichen Kanalkombinationen der Elektrodenen TL und TR (siehe Kapitel 1.4.2) gezeigt werden. Dabei wurden verschiedene Synchronisationsmaße verwendet, unter anderem auch die mittlere Phasenkohärenz R. Der durch den Abfall der Synchronisation definierte prä-iktale Zustand konnte eine Dauer von wenigen Minuten bis zu mehreren Stunden erreichen. Lagen für einen Patienten mehrere peri-iktale Datensätze vor, so konnte mindestens eine Kanalkombination gefunden werden, in der der Verlust der Synchronisation zu beobachten ist.

Mormann und Kollegen konnten zeigen, dass mit diesem Phänomen des Verlustes der Synchronisation ein hypothetischer prä-iktaler Zustand definiert und somit auch prinzipiell gegen das Auftreten von Anfällen vorgegangen werden kann.

Dieses Phänomen des Verlustes der Synchronisation erscheint unlogisch und widerspricht der Intuition. Ein Erklärungsversuch für dieses Verhalten ist die Idee der *Rekrutierung* von Neuronen für das Anfallsgeschehen. Ein Gehirnareal *koppelt* sich vom restlichen Gehirn ab und rekrutiert Neuronen, die im Anfallsgeschehen *mitwirken* sollen. Sind die implantierten

Elektroden so plaziert, dass Kontakte innerhalb und außerhalb des Areals liegen, kann dieser Prozess durch die Nutzung von Synchronisationsmaßen untersucht werden.

Zusammenfassend läßt sich sagen: Das Maß mittlere Phasenkohärenz R ist ein nichtlineares Synchronisationsmaß, mit dem nicht nur in (idealen) gekoppelten dynamischen Modellsystemen Synchronisation nachgewiesen werden kann, sondern es ist auch robust genug, um auf reale Zeitreihen wie EEG, die aus Aufzeichnungen der prächirurgischen Epilepsiediagnostik stammen, angewendet werden zu können. Dieses Maß und weitere Synchronisationsmaße ermöglichen es, einen hypothetischen Voranfallszustand zu definieren, so dass prinzipiell gegen das Auftreten von Anfällen vorgegangen werden kann.

R wird in seiner Berechnungszeit durch die Berechnung der Fourier-Transformation dominiert (siehe Gleichung 1.19). Die Nutzung der Fast-Fourier-Transformation schafft einen Proportionalitätsfaktor von $N \log(N)$ (mit N als Anzahl der Datenpunkte pro Datenfenster) zur Rechenzeit, welche aber mit heutigen Personal-Computern in Echtzeit zu berechnen ist. Dieser wird allerdings durch die Anzahl der möglichen Kanalkombinationen von z.B. $\frac{L(L-1)}{2}$, bei L möglichen Kanälen (in klinischen Applikationen bis zu $L = 256$) drastisch erhöht.

Personal-Computer eignen sich nicht, um ein Anfalls-Vorhersagesystem zu etablieren, da sie meist nur im Verbund die benötigte Rechenleistung bieten und viel Platz und viel Energie benötigen. Außerdem sind sie fehleranfällig, da z.B. Netzteile und sich bewegende Teile wie Festplatten und Lüfter unter Volllast im Dauerbetrieb ausfallen können. Natürlich dürfen auch nicht die Personalkosten für die Service-Techniker in der Kalkulation vergessen werden. Um miniaturisierte oder sogar tragbare Systeme aufbauen zu können, müssen andere *Arten* von Rechnerarchitekturen untersucht werden. Dabei muss gewährleistet werden, dass die Algorithmen möglichst exakt auf diese *neuen* Architekturen übertragbar sind.

Das folgende Kapitel beschäftigt sich mit solchen *neuen Architekturen*, die eine hohe Rechenkapazität bei gleichzeitig kleinem Platz- und Energieverbrauch bieten. Es handelt sich dabei um *neuronale Netze* und im speziellen *Zellulare Neuronale Netze*.

Kapitel 2

Künstliche Neuronale Netze

2.1 Neuronale Netze

2.1.1 Motivation

Die heute zur Verfügung stehende Rechenleistung von *Personal-Computern* (PC) (*von Neumann Maschinen*) bis hin zu wissenschaftlich genutzten *Super-Computern* oder *Rechner-Verbünden* (Cluster-Systeme) bietet genug Ressourcen für die verschiedensten Applikationen. Vom hoch qualitativen *nichtlinearen Videoschnitt* auf PCs, die privat angeschafft werden, bis hin zur Wettervorhersage auf Super-Computern sind heute viele Anwendungen möglich. Auch die Bildung von heterogenen Rechennetzen verbunden durch das Internet (*GRID*-Technologie), die entsprechend der Problemstellung konfiguriert und genutzt werden, ist keine Zukunftsmusik mehr. Die Idee von der *Rechenleistung aus der Steckdose* scheint greifbar nahe.

Trotz der steigenden *konventionellen* Computer-Rechenleistung wird auch auf dem Gebiet der künstlichen neuronalen Netze geforscht. Werden das menschliche Gehirn, das neuronale Netz schlecht hin, und Computer miteinander verglichen, so fällt auf, dass Computer in der Datenspeicherung und Verarbeitung schneller sind als das Gehirn. Transistoren, Schaltkreise in Computern, arbeiten heute im Gigahertzbereich. Komplexe arithmetische Aufgaben können schnell und in hoher Genauigkeit von Computern bearbeitet werden. Das menschliche Gehirn kann diese Geschwindigkeit nicht erreichen. Dafür kann das menschliche Gehirn andere Aufgaben lösen, die der Computer nicht einmal ansatzweise ausführen kann. Tätigkeiten wie z.B. Gehen, Sprechen, Erkennen von Personen (auch Jahre nach dem letzten Treffen) oder aus Erfahrungen in Situationen handeln sind Aufgaben, die von Computern kaum gelöst werden können.

Die Leistungsfähigkeit des Gehirns basiert auf seinem *massiv parallelen Aufbau*. Die *Neuronen*, die Bauelemente aus denen das Gehirn besteht, arbeiten im Millisekundenbereich. Sie sind viel langsamer als elektronische Bauteile. Trotzdem können komplexe Tätigkeiten wie z.B. das Sehen (Erkennen und Auswerten von Bildern) effizient und schnell bewältigt werden. Durch die höchst komplexe *Verkabelung* im Gehirn existiert ein Werkzeug, mit dem die Neuronen Aufgaben in einigen hundert Schritten bewältigen, die Computer in einigen

Millionen oder noch mehr Schritten kaum schaffen.

Die oben genannte Fähigkeiten, aus Erfahrungen heraus Situationen zu meistern oder fehlende Bruchstücke einer Aufgabenlösung durch Intuition zu erkennen, können mit dem Computer nicht simuliert werden. Die unterschiedliche Art und Weise der Speicherung wird als ein wichtiger Grund hierfür genannt. Das Wissen im Gehirn wird in einer *vagen* und vielschichtigen Weise gespeichert. Erinnerungen an Bilder, Töne oder Lösungswege können sehr schnell, ohne nachzudenken, zur richtigen Zeit verwendet werden. Erinnerungen werden assoziativ genutzt.

Weiterhin ist das Gehirn plastisch. Es verändert sich, um Probleme effizient zu lösen, z.B. das Erlernen einer neue Sprache.

Der Einsatz von Neuronalen Netzen und Computern hat dennoch häufig große Vorteile. Es gibt Aufgaben, die nur mit neuronalen Netzen bzw. Computern sinnvoll gelöst werden können. Tabelle 2.1 stellt die Eigenschaften eines menschlichen Gehirns und eines Computers gegenüber:

Eigenschaft	Gehirn	Computer
Parallelität	+	-
Präzision	-	+
Fehlertoleranz	+	-
Speicherzugriff	global	lokal
Mustererkennung	+	-
Fehlerloses Speichern	-	+
Lernfähigkeit	+	-
Rekonstruktion	+	-
Verallgemeinerung	+	-
Selbstorganisation	+	-

Tabelle 2.1: Vergleich Leistungen von Gehirn und Computer, + (-) bezeichnet eine hohe (niedrige) Leistung

2.1.2 Grundlagen

Neuronen

Wie in realen neuronalen Netzen sind in künstlichen neuronalen Netzen die Neuronen (*Nervenzellen*, siehe Abbildung 2.1) die Grundbausteine. Sie sind dem biologischen Vorbild nachempfunden. Sie bestehen aus einem *Zellkörper*, aus den *Dendriten* (Eingabeleitungen), die die Eingabe an die Zelle weiterleiten und dem *Axon* (Ausgabeleitung), das die Ausgabe der Zelle weiterleitet.

Ein Axon verzweigt sich und tritt mit anderen Dendriten über *Synapsen* in Kontakt. Die Stärke der Verbindung durch die Synapsen wird durch *Verbindungsgewichte* geregelt.

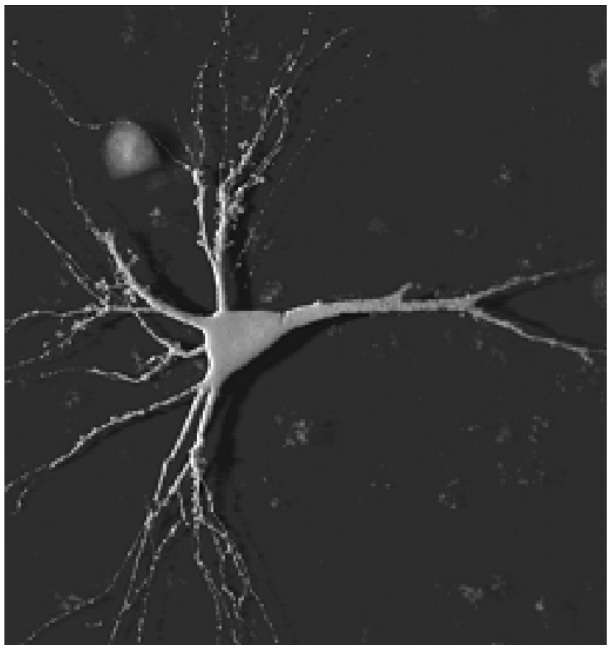


Abbildung 2.1: Neuron

Das menschliche Gehirn besteht aus etwa 10^{11} Neuronen mit jeweils 10^3 bis 10^4 Synapsen. Diese Komplexität kann nicht vollständig nachgebaut werden.

Ein künstliches Neuron (siehe Abbildung 2.2) wird üblicherweise durch ein n-Tupel $(\vec{x}, \vec{\omega}, f_a, f_o, o)$ beschrieben, wobei die Komponenten des Tupels folgende Bedeutungen haben:

- *Eingabe* $\vec{x} = (x_1, ..., x_n)$
- *Gewichte* $\vec{\omega} = (\omega_1, ..., \omega_n)$, die mit den dazugehörigen Eingaben multipliziert werden
- *Aktivitätsfunktion* $f_a : \Re^n \times \Re^n \rightarrow \Re$
- *Ausgabe-* bzw. *Transfer-* oder *Aktivierungsfunktion* $f_o : \Re \rightarrow \Re$
- *Ausgabe* o

Eine häufige Wahl für die Aktivitätsfunktion ist die gewichtete Summe aller Eingaben:

$$f_a(\vec{x}, \vec{\omega}) = \sum_{i=1}^{n} x_i \omega_i \tag{2.1}$$

Für die Ausgabe des Neurons gilt:

$$o = f_o(f_a(\vec{x}, \vec{\omega})) \tag{2.2}$$

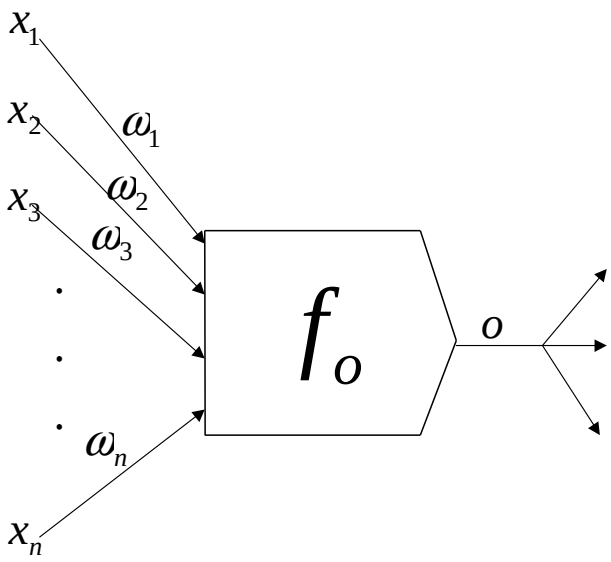


Abbildung 2.2: Künstliches Neuron

Bei biologischen Neuronen muss für das Auslösen eines *Aktionspotentials* o ein bestimmter *Schwellenwert* S überschritten werden. Damit kann z.B. f_o durch Verwendung eines binären Schwellenwertes modelliert werden:

$$o = f_o(\sum_{i=1}^{n} x_i\omega_i) = \begin{cases} 1 & : \quad \sum_{i=1}^{n} x_i\omega_i \geq S \\ 0 & : \quad \text{sonst} \end{cases} \tag{2.3}$$

Da in diesem Modell nicht die Intensität aufeinander folgender Aktionspotentiale biologischer Neuronen berücksichtigt wird, werden lineare Ausgabefunktionen verwendet.

Der zeitliche Abstand, in dem Aktionspotentiale durch Nervenzellen weitergereicht werden, ist nach unten beschränkt (er liegt im Millisekunden Bereich). Deshalb sollte im formalen Neuronenmodell eine beschränkte und differenzierbare Ausgabefunktion verwendet werden. *Sigmoidale* Funktionen f_s erfüllen diese Voraussetzungen:

- $f_s : \Re \rightarrow [0, 1]$
- monoton wachsend
- differenzierbar
- $\lim_{x\rightarrow-\infty} f_s = Lg$, mit Lg als kleinsten Wert, den F_s annehmen kann
- $\lim_{x\rightarrow+\infty} f_s = Rg$, mit Rg als größten Wert, den F_s annehmen kann
- $Lg < Rg$

Die Reizschwelle, die in einem biologischen Neuron überschritten werden muss, damit es *feuern* kann, wird durch den *Schwellenwert* oder *Bias* dargestellt. Dieser Schwellenwert

kann in f_o durch einen zusätzlichen Parameter oder als zusätzliches Gewicht ω_{Bias} realisiert werden. Abbildung 2.3 stellt noch einmal exemplarisch ein künstliches Neuron dar, mit den Eingaben $\vec{x}$ und ihren Gewichten $\vec{\omega}$ und mit der Eingabe 1, die mit dem dazugehörigen Gewicht ω_{Bias} den Schwellwert des Neurons darstellt.

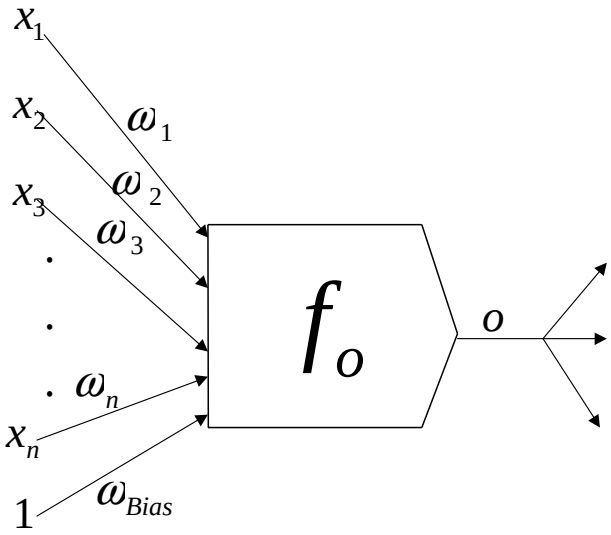


Abbildung 2.3: Künstliches Neuron mit Schwellenwert

2.1.3 Neuronenverbände

Werden mehrere Neuronen miteinander verbunden, so entsteht ein neuronales Netz. Die folgende Auflistung stellt einige grundsätzliche Eigenschaften eines neuronalen Netzes dar:

- N Neuronen sind Knoten des Netzes
- Kanten sind Verbindungen
- jedes Neuron hat beliebig viele Eingänge
- jedes Neuron sendet über eine beliebige Anzahl von Verbindungen genau eine Ausgabe
- $L \leq N$ Neuronen bekommen von außen Eingaben (Eingabeschicht)
- $M \leq N$ Neuronen geben nach außen Ausgaben (Ausgabeschicht)

Die Abbildung 2.4 stellt beispielhaft ein künstliches Neuronales Netz mit drei Eingabe-Neuronen (x_1 bis x_3) auf der linken Seite in der *Eingabeschicht* (*Input-Layer*), acht Neuronen in den mittleren Schichten (*Verborgene-Schicht*, *Hidden-Layer*) und vier Ausgabe-Neuronen (o_1 bis o_4) auf der rechten Seite in der *Ausgabeschicht* (*Output-Layer*) dar. Dieses Netz ist vorwärtsgerichtet und besitzt keine Rückkopplungen.

Die Verbindungen von einem Neuron i zu einem Neuron j werden durch die Gewichte ω_{ij} beschrieben. Damit kann die *Topologie* oder *Verbindungsstruktur* mit der *Verbindungs-* oder

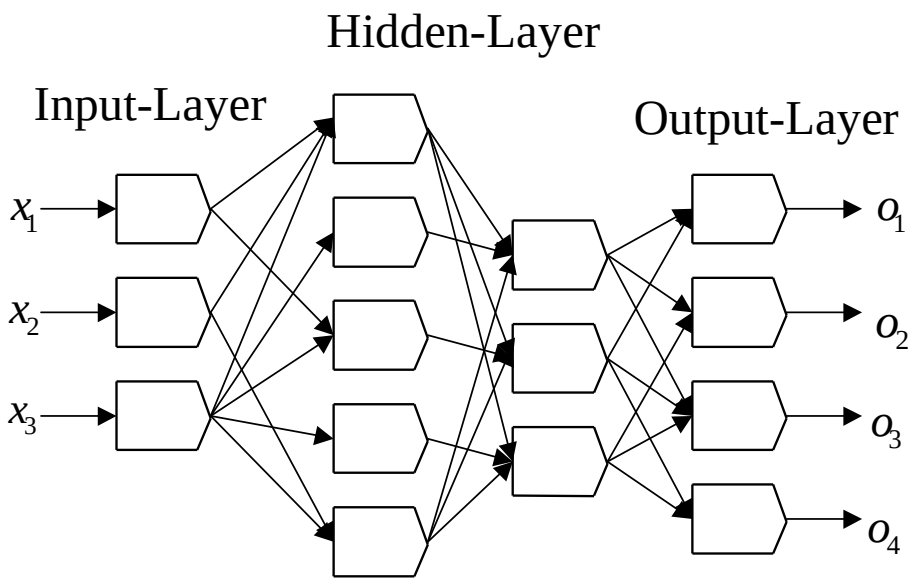


Abbildung 2.4: Vorwärts gerichtetes neuronales Netz ohne Rückkopplung, nicht existierende Verbindungen haben das Gewicht $\omega_{ij} = 0$

Gewichts-Matrix W beschrieben werden. Gilt $\omega_{ij} = 0$, so existiert keine Verbindung zwischen den Neuronen i und j. Gilt $\omega_{ij} < 0$ ($\omega_{ij} > 0$), so existiert eine hemmende (anregende) Verbindung zwischen den Neuronen i und j.

Topologien

Neuronale Netze lassen sich in Netze mit und ohne Rückkopplung unterteilen:

- Rückkopplungfreie Netze (*vorwärts gerichtete* Netze)
  1. Ebenenweise verbunden
     - Netze sind in mehrere Schichten unterteilt
     - Verbindungen nur von Schicht zur Schicht
  2. Allgemeine
     - Verbindungen über Schichten hinweg sind erlaubt
- Rückgekoppelte Netze (*rekurrente* Netze)
  1. direkte Rückkopplung
     - Neuronen können ihre eigene Aktivität über eine Verbindung vom eigenen Ausgang zum eigenen Eingang verstärken oder hemmen
  2. indirekte Rückkopplung
     - Neuronen höherer Schichten (zur Ausgabeschicht hin) sind mit Neuronen niedriger Schichten (zur Eingabeschicht hin) verbunden

3. laterale Rückkopplung
   - Rückkopplungen innerhalb einer Schicht sind möglich
4. vollständig verbunden
   - Beispiel hierfür sind sog. *Hopfield-Netze* ([RR93]) mit einer symmetrischen Gewichts-Matrix W, deren Einträge auf der Diagonalen 0 sind.

Rückgekoppelte Netze werden eingesetzt, um *Zeitabhängigkeiten* in Daten modellieren zu können. Die bereits verarbeiteten Daten werden mit der Eingabe als neue Eingabe genutzt.

2.1.4 Lernen

Neuronale Netze können i. Allg. nicht sofort die gestellte Aufgabe erledigen. Analog zu einem biologischen System müssen sie ein *Training* absolvieren und die nötigen Fähigkeiten *lernen*. Das Lernen erfolgt durch *Selbstmodifikation* nach einer Lernregel. Die folgenden Modifikationen können durchgeführt werden:

- Modifikation der Gewichte ω_{ij}
  - Verbindungen ändern
  - neue Verbindungen setzen (vorher $\omega_{ij} = 0$, nachher $\omega_{ij} \neq 0$)
  - Verbindungen löschen (vorher $\omega_{ij} \neq 0$, nachher $\omega_{ij} = 0$)
- Modifikation der Schwellenwerte
- Modifikation der Transferfunktion
- Modifikation des Netzwerkes
  - Neuronen löschen
  - Neuronen der Topologie hinzufügen

Die erste und zweite Modifikation wird häufig im Training verwendet.

Das Lernen kann in drei Varianten unterteilt werden:

- Überwachtes Lernen
  - die Differenz zwischen der aktuellen Ausgabe und der idealen Ausgabe fließt ins Training ein
  - Trainingsdaten, bestehend aus Eingabedaten und idealen Ausgabedaten, müssen vorhanden sein
- Bestärktes Lernen
  - nur *richtig* oder *falsch* (keine Differenz) fließt ins Training ein

  - Trainingsdaten, bestehend aus Eingabedaten und idealen Ausgabedaten müssen vorhanden sein

- Unüberwachtes Lernen (*selbst organisert*)
  - nur Trainingsdaten, bestehend aus Eingabedaten und idealen Ausgabedaten sind vorhanden

Für Details zum Thema Lernen sei auf den Anhang B und auf das Kapitel 3.2.3 verwiesen.

Zusammenfassend läßt sich sagen: Technische neuronale Netze ermöglichen eine völlig neue Herangehensweise an Probleme, die mit herkömmlichen Computern mit einem hohen Zeitbedarf gelöst werden können[1]. Erforderlich dafür ist die Loslösung von der Darstellung durch programmierte Algorithmen, die auf Computern zur Lösung der Probleme eingesetzt werden. Die Probleme müssen durch entsprechende Wahl der Neuronen und der Topologie des Netzes abgebildet werden. Es gibt keine Programmierung mehr, statt dessen muss durch Training die Problemstellung dem künstlichen neuronalen Netz *beigebracht* werden.

Technische neuronale Netze haben den Vorteil, dass bekannte Eigenschaften von natürlichen neuronalen Netzen, wie dem menschlichen Gehirn, integriert werden können. So können Lösungswege der Natur in die technische Welt übertragen werden. Gleichzeitig kann aber auch durch Modellierung in technischen neuronalen Netzen versucht werden, die Natur, in diesem Fall den Aufbau der dem Problem unterliegendene Strukturen, zu verstehen.

Technische neuronale Netze bieten die Möglichkeit, Höchstleistungsberechnungen durchzuführen und dabei einen Bruchteil der Energie und des notwendigen Raumes, wenn sie als *VLSI-Realisationen* (*Halbleiter-Bausteine*) verfügbar sind, zu verbrauchen. Das folgende Kapitel beschäftigt sich mit einer Klasse solcher Netze, den *Zellularen Neuronalen Netzen.*

Weitere Informationen zum Thema neuronale Netze sind z.B. in [RR93] zu finden. Im Anhang E findet sich eine Auflistung über einige wichtige Daten in der Entwicklung künstlicher neuronaler Netze in den letzten 60 Jahren. Dabei wird auf das *Modell von McCulloch und Pitts*, die *Hebb'sche Lernregel* und auf das *Perzeptron* genauer eingegangen.

[1] Probleme die ein polynomielles Laufzeitverhalten mit hohem Polynomgrad oder sogar ein exponentielles Laufzeitverhalten aufweisen.

Kapitel 3

Software und Hardware CNN

3.1 Zellulare Neuronale Netze (CNN)

Zellulare Neuronale Netze, kurz *CNN*, sind eine Klasse von künstlichen neuronalen Netzen mit bestimmten Eigenschaften. Sie bestehen aus einer Menge von Neuronen, die allerdings nur lokal miteinander verbunden sind. Dises Paradigma wurde zum ersten Mal von Chua und Yang im Jahr 1988 ([CY88], [Ch98]) formuliert:

```
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.
```

Diese Definition enthält die drei wichtigsten Punkte, die für jedes CNN gelten:

1. Die Neuronen sind beliebig räumlich angeordnet, der Geometrie sind keine Grenzen gesetzt.

2. Jedes Neuron ist nur mit Neuronen aus der nächsten Nachbarschaft verbunden.

3. Jedes Neuron ist ein dynamisches System, welches durch seine Dynamik und drei Zugänge (der Eingabe, dem Status und der Ausgabe) beschrieben wird.

Aufgrund der lokalen Kopplung können CNN auch als nichtlineare analoge Schaltungen in Form von *Halbleiter-Bauteilen* (*ICs* bzw. *Chips*) zur massiv parallelen Verarbeitung von Signalen in Echtzeit gebaut werden. Allgemeine technische neuronale Netze können dagegen aufgrund der i. Allg. hochkomplexen Verkabelung nicht als Geräte realisiert werden.

3.1.1 Konzept

In diesem Unterkapitel werden CNN anhand einer zweidimensionalen Struktur erklärt, dem *Starndard-CNN*. Theoretisch ist jede geometrische Form denkbar.

Abbildung 3.1 zeigt ein zweidimensionales CNN. Um die einzelne markierte Zelle ist eine 3×3 und eine 5×5 Umgebung zu erkennen. Diese Umgebungen zeigen zwei Möglichkeiten der Nachbarschaftswahl, die 1er Sphäre und die 2er Sphäre. Diese Wahl wird für *alle* Zellen

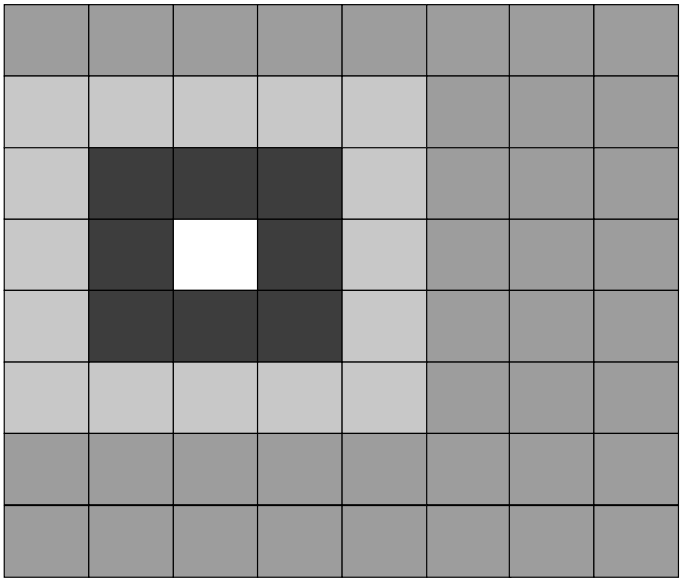


Abbildung 3.1: Zweidimensionales CNN mit Zelle (weißes Quadrat) und dazugehöriger 3×3 (schwarze Zellen) und 5×5 (hell graue Zellen) Umgebung

getroffen. D.h. jede Zelle kann nur auf diese Art und Weise mit der Nachbarschaft verbunden werden.

Um Probleme mit Randzellen und deren Nachbarschaften (*virtuelle* Zellen, falls sie außerhalb des CNN liegen) zu vermeiden, können folgende *Randbedingungen* definiert werden (vgl. Abbildung 3.2):

- Die Randzellen werden auf einen konstanten Wert gesetzt. Ist dieser Wert 0, werden die Randbedingungen *neutral* genannt.
- Die Randzellen werden mit den Randzellen auf der anderen Seite verbunden (*wrap around* bzw. periodische Randbedingung). Es entsteht eine geschlossene Struktur.
- Die Randzellen werden mit einem Versatz um eine Zeile mit den Randzellen auf der anderen Seite verbunden (*closed spiral* bzw. geschlossene Spirale).

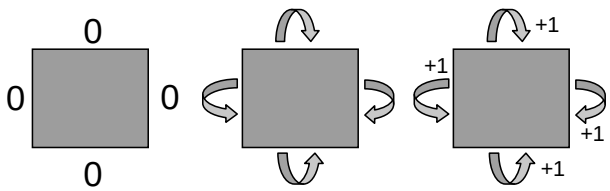


Abbildung 3.2: Möglichkeiten zur Handhabung der Randbedingung (links - *neutrale* Randbedingung, Mitte - *wrap around*, rechts - *closed spiral*)

Es sind beliebige Konfigurationen denkbar, um die virtuellen Zellen zu beschreiben. Die drei oben genannten sind die Gebräuchlichsten.

Jede Zelle (siehe Abbildung 3.3) besteht formal gesehen aus einem Eingang u, einem Status x, einem Schwellenwert z und einem Ausgang y.

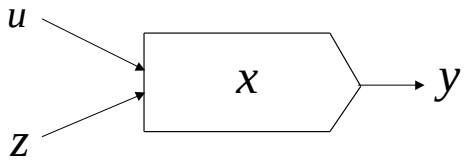


Abbildung 3.3: Schematische Darstellung der Zelle i

Das Verhalten eines CNN kann über ein gekoppeltes Differentialgleichungssystem (*Zustandsgleichung*) beschrieben werden. Im vorliegenden Fall eines zweidimensionalen CNN, mit $M \times N$ Zellen, gilt für jede Zelle ij ($0 < i < M$ und $0 < j < N$):

$$C_{ij}\frac{d}{d\tau}x_{ij}(\tau) = \frac{-x_{ij}(\tau)}{R_{ij}} + \sum_{kl \in S_A} a_{kl}y_{kl}(\tau) + \sum_{op \in S_B} b_{op}u_{op} + z_{ij} \tag{3.1}$$

mit $x_{ij}(\tau)$ als Status der Zelle ij, $y_{kl}(\tau)$ als Ausgabe der Zelle kl, u_{op} als Eingabe der Zelle op, z_{ij} als Schwellenwert der Zelle ij und den Gewichten (*Templates*) a_{kl} (*Feedback*) und b_{op} (*Feedforward*) mit den Elementen kl und op aus den Sphären S_A und S_B. C_{ij} und R_{ij} entsprechen der Kapzität und dem Widerstand einer jeden Zelle ij und werden ohne Beschränkungung der Allgemeinheit $R = C = 1$ gesetzt.

Die Ausgabe einer jeden Zelle ij wird durch die *Kennlinie* (*Ausgabefunktion*) bestimmt:

$$y_{ij}(\tau) = f(x_{ij}(\tau)) = \frac{1}{2}(|x_{ij}(\tau) + 1| - |x_{ij}(\tau) - 1|) \tag{3.2}$$

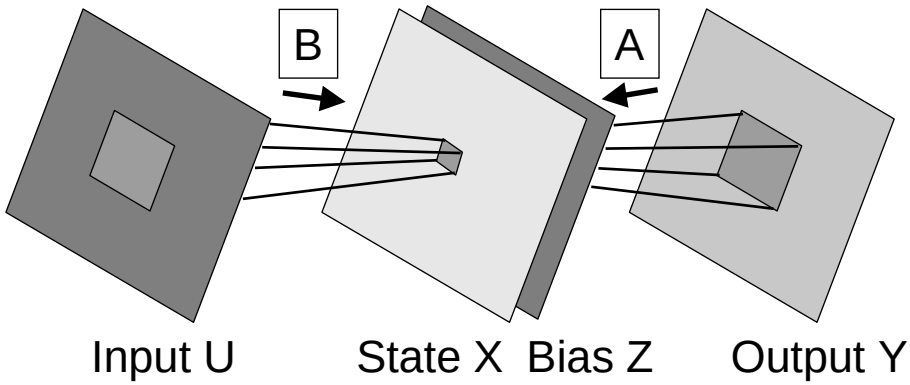


Abbildung 3.4: Schematische Darstellung der Verkabelung eines CNN

Auch diese *stückweise lineare* Funktion kann bei Bedarf gegen eine andere ausgetauscht werden. Sie wird aber im Standard-CNN genutzt. Die Werte, die jede Zelle in der Eingabe und in der Ausgabe annehmen kann, liegen im Bereich $[-1, 1]$. Sie können also alle *Graustufen* zwischen *weiß* ($\cong -1$) und *schwarz* ($\cong 1$) annehmen. Werden beim Voreinstellen des CNN auch die Werte des Status gesetzt, so werden auch hier die Grenzen $[-1, 1]$ eingehalten. Abbildung 3.4 stellt die Verkabelung eines CNN dar. Diese bildliche Darstellung entspricht der Gleichung 3.1. Ein CNN kann als ein System von vier Ebenen (Eingabe U, Status X, Schwellenwert Z und Ausgabe Y) verstanden werden. Durch die Definition der Größe der Nachbarschaft wird die Verbindungsstruktur zwischen allen Zellen und deren Ebenen festgelegt.

Eine mögliche Konfiguration kann folgendermaßen aussehen:
Wird zwischen der Eingabe und dem Status eine *invariante* 1er Nachbarschaft festgelegt und zwischen der Ausgabe und dem Status eine *invariante* 2er Nachbarschaft festgelegt, sowie ein *invarianter* Schwellenwert z gesetzt, so gilt Folgendes:

1. Nur die 3×3 umgebenen Zellen haben über ihre Eingabe und das Feedforward-Template Einfluss auf den Status einer jeden Zelle.

2. Nur die 5×5 umgebenen Zellen haben über ihre Ausgabe und das Feedback-Template Einfluss auf den Status einer jeden Zelle.

3. Der Schwellenwert $z_{ij} = z$ gilt für alle Zellen ij und damit global für das gesamte Netz.

Somit kann diese Konfiguration mit zwei Matrizen und einem Schwellenwert beschrieben werden. Eine Matrix mit $3 * 3 = 9$ Einträgen beschreibt das Feedforward-Template B, eine weitere mit $5 * 5 = 25$ Einträgen das Feedback-Template A. Dadurch, dass beide Templates invariant sind, gelten sie für alle Zellen gleichermaßen. Zu bemerken ist, dass das zentrale Element von A und B für jede Zelle eine Kopplung auf sich selbst ist. Somit wird das Verhalten des CNN in diesem Fall durch $3 * 3 + 5 * 5 + 1 = 35$ Parameter vollständig beschrieben. Bei einem CNN mit z.B. $64 * 64 = 4096$ Zellen sind 35 Parameter eine sehr geringe Zahl. Im Gegensatz dazu benötigt ein Hopfield-Netz dieser Größe $4096 * (4096 - 1) = 16773120$ zu bestimmende Parameter.

3.1.2 Aufgabenfelder und Beispiele

CNN werden in vielen wissenschaftlichen Aufgabenfeldern eingesetzt.

Komplexe Aufgaben aus der *Bildverarbeitung* können mit CNN gelöst werden. Dabei werden spezielle Eigenschaften von Objekten extrahiert und klassifiziert, eine Kollisionserkennung durchgeführt und die Objektanzahl und Größe bestimmt. Aber auch die *Analyse von dreidimensionalen Oberflächen* kann mit CNN durchgeführt werden. Dabei können z.B. Extremwerte erkannt werden oder Flächen, welche nicht vorgegebenen Gradienten entsprechen. Ebenso können *partielle Differentialgleichungen* mit CNN gelöst werden. Dabei werden u.a. thermische Verläufe, die Abstrahlung von Antennen und medizinische Probleme betrachtet. CNN können auch zur Mustererkennung in skalaren Signalen genutzt werden ([Ca03]).

Diese Aufzählung enthält nur wenige Beispiele von Problemstellungen, die mit CNN zu realisieren sind. Weitere Beispiele sind in z.B. [Ch98], [FTFE01], [KTW00], [PTW96] zu finden.

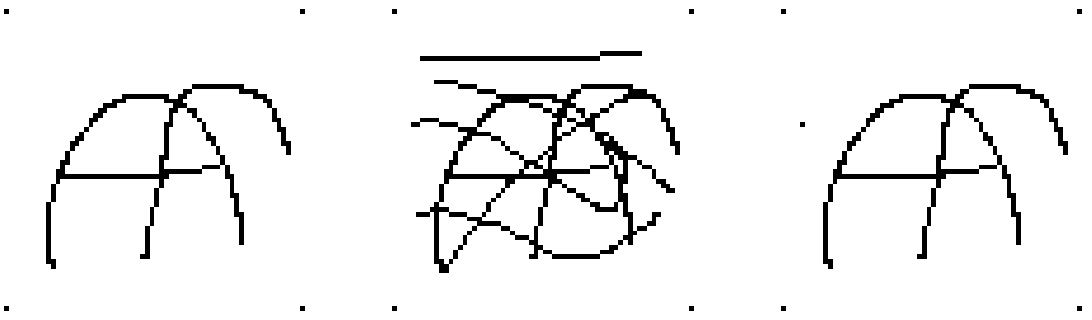
Im Folgenden werden einige Beispiele aufgeführt, die die Mächtigkeit des Werkzeugs CNN illustrieren. Wegen der zweidimensionalen Topologie wird von *Bildern*, z.B. Eingabe-Bild, geredet. Die Ecken von Bildern mit großen weißen Flächen werden durch schwarze Punkte dargestellt, damit sie von der Seite unterschieden werden können. Weitere Informationen zu den im folgenden Text beschriebenen CNN-Typen sind in [RK99] und [Ch98] zu finden.

AND-CNN

Aufbau des CNN Das AND-CNN wird durch folgende Gewichte A, B und den Schwellenwert Z bestimmt:

$$A = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{Bmatrix} \quad B = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{Bmatrix} \quad Z = \{ \ -1 \ \} \tag{3.3}$$

Aufgabe bzw. Ablauf Gegeben sind zwei statische binäre Bilder (Bilder die nur schwarze und weiße Bildpunkte enthalten) P_1, P_2.
Das Eingabe-Bild U wird mit dem ersten binären Bild geladen, $U = P_1$.
Das erste Status-Bild wird mit dem zweiten binären Bild geladen, $X(0) = P_2$.
Für die Ausgabe gilt nach einer zeitlichen Entwicklung ($Y(t) \Rightarrow Y(\infty)$):
Das Ausgabe-Bild ist ein binäres, das der logischen Operation UND zwischen P_1 und P_2 entspricht. Dabei entsprechen die schwarz eingefärbten Bildpunkte dem Bool'schen Wert *wahr* und die weiß eingefärbten dem Bool'schen Wert *falsch*.



a.) Eingabe-Bild U b.) Status-Bild $X(0)$ c.) Ausgabe-Bild $Y(\infty)$

Abbildung 3.5: Beispielanwendung des AND-CNN

Die Abbildungen 3.5 a.) bis c.) demonstrieren die Ausführung der logischen Operation UND. Weitere Operationen wie NOT, OR, XOR oder Maskierung können durchgeführt

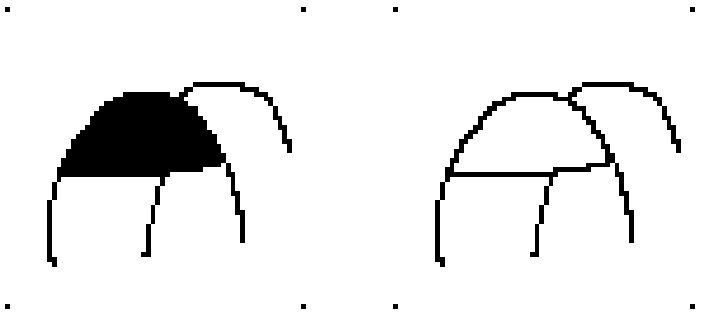
werden, so dass durch Kombination von mehreren CNN, bzw. einer Kaskadierung mehrerer CNN-Operationen, mathematische Operationen durchgeführt werden können.

Kanten-Finder-CNN

Aufbau des CNN Das Kanten-Finder-CNN wird durch folgende Gewichte A, B und den Schwellenwert Z bestimmt:

$$A = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{Bmatrix} \quad B = \begin{Bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{Bmatrix} \quad Z = \{ -1 \} \tag{3.4}$$

Aufgabe bzw. Ablauf Gegeben ist ein statisches binäres Bild P.
Das Eingabe-Bild wird mit dem binären Bild geladen, $U = P$.
Das erste Status-Bild $X(0)$ kann beliebig gewählt werden oder per Standardeinstellung $X(0) = 0$ sein.
Die Randwerte, die virtuellen Zellen, werden auf -1 (weiß) gesetzt.
Für die Ausgabe gilt nach einer zeitlichen Entwicklung ($Y(t) \Rightarrow Y(\infty)$):
Das Ausgabe-Bild ist ein binäres, welches nur noch die Ränder aller schwarzen Strukturen enthält.



a.) Eingabe-Bild U b.) Ausgabe-Bild $Y(\infty)$

Abbildung 3.6: Beispielanwendung des Kanten-Finder-CNN

Kanten finden (siehe Abbildungen 3.6 a.) und b.)), genauso wie z.B. Lücken in Strukturen finden ist ein komplexes Problem in der Bildverarbeitung, das von CNN in einer hohen Geschwindigkeit gelöst werden kann. Auch hier bieten CNN einen weiteren interessanten Ansatz als Ergänzung oder gar als Ersatz zu computerbasierten Lösungen.

Game-of-Life-CNN

Das *Spiel des Lebens* (*Game of Life*) ist ein sehr prominentes Beispiel für einen *endlichen Automaten*[1]. Das Spiel entwickelt sich in diskreten Zeitschritten. Eine Zelle, die schwarz

[1]Ein endlicher Automat ist eine *Turing Maschine*, ein Konstrukt aus der Informatik. Jeder Computer, der mit einem festen Programm arbeitet, ist durch eine Turing Maschine beschreibbar. Umgekehrt gibt es

eingefärbt ist lebt, eine Zelle, die weiß eingefärbt ist, ist tot. Das Spiel wird auf einer unendlich großen Fläche gespielt.

Das Überleben einer jeden Zelle hängt von der Konfiguration der acht Zellen in der unmittelbaren Umgebung (3×3 Umgebung) ab. Die folgenden lokalen Regeln gelten für das Spiel:

1. Geburt: Eine tote Zelle zum Zeitpunkt t wird lebendig zum Zeitpunkt $t+1$, wenn genau drei der acht Nachbarn am Leben sind.
2. Tod bei Überbevölkerung: Eine lebendige Zelle zum Zeitpunkt t stirbt zum Zeitpunkt $t+1$, wenn mehr als drei der acht Nachbarn am Leben sind.
3. Tod durch Vereinsamung: Eine lebendige Zelle zum Zeitpunkt t stirbt zum Zeitpunkt $t+1$, wenn weniger als zwei der acht Nachbarn am Leben sind.
4. Überleben: Eine lebendige Zelle zum Zeitpunkt t bleibt zum Zeitpunkt $t+1$ am Leben, wenn genau zwei oder drei der acht Nachbarn am Leben sind.

Dieses Problem kann in ein Game-of-Life-CNN überführt werden. Dadurch, dass dieses Problem nicht linear separierbar ist, muss es in eine Kaskade von drei CNN aufgeteilt werden, welche in einer bestimmten Reihenfolge angesprochen werden.

Aufbau des CNN Das Game-of-Life-CNN wird durch zwei CNN K_1, K_2 und dem in 3.1.2 genannten AND-CNN realisiert.

Für K_1 gilt:

$$A = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{Bmatrix} \quad B = \begin{Bmatrix} -1 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & -1 \end{Bmatrix} \quad Z = \{ \ -1 \ \} \tag{3.5}$$

Für K_2 gilt:

$$A = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{Bmatrix} \quad B = \begin{Bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{Bmatrix} \quad Z = \{ \ 4 \ \} \tag{3.6}$$

Das AND-CNN wurde bereits in 3.1.2 detailiert beschrieben.

Aufgabe bzw. Ablauf Gegeben ist ein statisches binäres Bild P.

Zunächst wird P in K_1 bearbeitet:
Das Eingabe-Bild wird mit dem binären Bild geladen, $U_1 = P$.
Das erste Status-Bild $X_1(t)$ kann beliebig gewählt werden oder per Standardeinstellung $X_1(t) = 0$ sein.

---

zu jeder Turing Maschine ein Computerprogramm, das diese simuliert. Für weitere Informationen sei auf [Ch98] und die Referenzen darin verwiesen.

Parallel dazu (oder danach) wird P in K_2 bearbeitet:
Das Eingabe-Bild wird mit dem binären Bild geladen, $U_2 = P$.
Das erste Status-Bild $X_2(t)$ kann beliebig gewählt werden oder per Standardeinstellung $X_2(t) = 0$ sein.

Die beiden Ausgaben $Y_1(\infty)$ und $Y_2(\infty)$ werden mit einer AND-CNN-Operation verknüpft. Die resutierende Ausgabe $Y_{AND-CNN}(\infty)$ enthält den ersten Schritt des Game-of-Life. Jeder weitere Schritt erfordert die Abarbeitung der oben beschriebenen Kaskade von CNN-Operationen.
Das Game-of-Life-CNN ist ein Beispiel für ein CNN, welches durch Aneinanderreihung von CNN-Operationen die ihm gestellte Aufgaben löst.

Das Game-Of-Life-CNN ist ein Beispiel für die Möglichkeit, ein Problem aus der Welt der Computer in die Welt der künstlichen neuronalen Netze zu überführen. Dadurch, dass der *Spiel des Lebens* Automat eine *universelle Turing Maschine* ist ([BCG82]) und *jedes* geschriebene Programm durch eine universelle Turing Maschine dargestellt werden kann, kann also auch *jedes* geschriebene Programm auf ein Zellulares Neuronales Netz übertragen werden. Diese Tatsache ist der Nachweis, dass jedes Computer-Problem mit CNN gelöst werden kann. Durch vorhandene schaltungstechische Realisationen erscheinen CNN als Alternative zur herkömmlichen Hochgeschwindigkeits-Datenverarbeitung.

Zusammenfassend läßt sich sagen: Zellulare Neuronale Netze stellen eine besondere Klasse von künstlichen neuronalen Netzen dar. Durch die lokale Kopplung der Zellen verringert sich die Komplexität der technischen Realisation. CNN sind als schaltungstechnische Realisationen erhältlich. Damit ist die Nutzung eines CNN als Koprozessor zu einem bestehenden System oder sogar als eigenständige Recheneinheit möglich.

Interessant ist an dieser Stelle der Vergleich zwischen erzielbarer Geschwindigkeit, bei gleichzeitiger Betrachtung des verbrauchten Volumens für das Bauteil und der verbrauchten Energie im Betrieb. Handelsübliche Computer werden deutlich übertroffen. Ein $25mm^2$ großer 64×64 CNN-Chip mit linearen 3×3 A und B Templates ist bei z.B. Bildverarbeitungsoperationen, wie *Halftoning* (aus Grauwerten wird ein Schwarz-Weiß Raster, ähnlich wie beim Faxen, erzeugt), etwa 2000 mal schneller als ein aktueller Pentium4 2GHz Computer. Gleichzeitig verbraucht er nur einige mW Energie, gegen die 90 W des Prozessors (unter Volllast, der Energieverbrauch für zusätzliche Komponenten eines Computers ist nicht eingerechnet). Diese hohe Geschwindigkeit wird durch die *massive parallele Struktur* der CNN erreicht. Jede Zelle arbeitet parallel (gleichzeitig) mit den anderen Zellen des CNN am gestellten Problem.

Da endliche Automaten als CNN implementiert werden können, sind den Aufgabenstellungen, die mit CNN gelöst werden können, keine Grenzen gesetzt. Allerdings erfordert die *Programmierung* eines CNN ein Umdenken. Die Konfigurationen der wenigsten CNN, die für spezielle Probleme genutzt werden können, lassen sich analytisch berechnen, d.h. die A und B Templates und der Schwellenwert Z brauchen nur noch aus der Lösung der Differentialgleichung 3.1 abgelesen werden. Statt dessen muss hier fast immer, wie bei den allgemeinen künstlichen Neuronalen Netzen, die Technik der Optimierung genutzt werden (siehe Kapitel 3.2.3 und Anhang B). Natürlich muss vorher das Problem erst in eine

Darstellungsform gebracht werden, die im CNN abbildbar ist. Die folgenden Abschnitte beschäftigen sich mit zwei CNN-Realisationen, einer Software-Implementation, *SCNN*, und einer schaltungstechnischen Implementation, *Aladdin-ACE4K*.

3.2 SCNN - ein universeller Software Simulator

In diesem Abschnitt wird eines der zwei hier genutzten Werkzeuge, das *SCNN Software System* besprochen.

Um Ideen und Aufgabenstellungen durch CNN zu verwirklichen, wird in der Regel zunächst ein allgemeiner Simulator genutzt. Solch ein Simulator sollte alle Möglichkeiten bieten, die das System theoretisch erfüllen kann. Idealerweise sollte er im Quelltext vorliegen, so dass mögliche negative Seiteneffekte abgefangen und auf die Realisation des Simulators oder der Aufgabenstellung zurückgeführt werden können. Ein solches System für CNN wird durch SCNN, einen Software Simulator, dargestellt. Es wurde über Jahre hinweg im Institut für Angewandte Physik der Universität Frankfurt ([Ku96], [KTW96], [KLT00], [KTAGFLPSW00]) entwickelt und erweitert.

3.2.1 Konzept

SCNN ist ein universeller Simulator für Zellulare Neuronale Netze. Die Software wurde in portablem C[2] unter AIX [3] entwickelt. SCNN wurde aber auch für weitere Betriebssysteme wie Linux und Windows portiert[4].

SCNN kann sowohl per Benutzeroberfläche, als auch per Skript gesteuert werden. Die Berechnungen wurden nicht auf dem Arbeitsplatz-Computer durchgeführt, da SCNN, je nach Konfiguration, durchaus mehrere Tage braucht, um Endergebnisse zu liefern.

Im folgenden werden die *Haupteigenschaften* der globalen Struktur des SCNN beschrieben. SCNN kann beliebig viele Zell-Schichten mit beliebig vielen Zellen pro Schicht simulieren. Dabei kann die Topologie einer jeden Schicht ein-, zwei- oder dreidimensional Verbindungen zwischen den eigenen Zellen erlauben. Genauso flexibel ist die Verbindung zwischen den Schichten selbst. Weiterhin kann jeder Schicht eine andere Ausgabefunktion zugeordnet werden. Damit können die Schichten als verschiedene CNN angesehen werden, die miteinander verknüpft werden können.

Folgende Ausgabefunktionen (*Kennlinien*) stehen zur Verfügung (mit m als Nummer der Schicht):

- $f(x_{ij}^m(\tau)) = \frac{1}{2}(|x_{ij}^m(\tau)+1| - |x_{ij}^m(\tau)-1|)$, Standardfunktion stückweise linear
- $f(x_{ij}^m(\tau)) = \frac{2}{1+exp(-\beta * x_{ij}^m(\tau))} - 1$, sigmoidale Funktion mit Steigungsparameter β
- $f(x_{ij}^m(\tau)) = sgn(x_{ij}^m(\tau))$

[2] C ist eine prozedurale Programmiersprache, eine Standardsprache, die für viele verschiedene Betriebssysteme verfügbar ist.

[3] AIX ist ein Unix-ähnliches Betriebssystem.

[4] Diese Portierungen war für die vorliegende Dissertation notwendig und wurde ebenfalls im Rahmen derselben durchgeführt, da die Arbeitsgruppe nur über Windows und Linux PCs verfügt.

- $f(x_{ij}^m(\tau)) = x_{ij}^m(\tau)$
- tabellarisch, mit stückweise linearer oder kubischer Spline-Interpolation

Zusätzlich kann jeder Schicht eine weitere Vorverarbeitung-Ausgabefunktion g^m zugewiesen werden, mit $y_{ij}^m(\tau) = f(g^m(x_{ij}^m(\tau)))$. Diese Vorverarbeitung-Ausgabefunktion wird in tabellarischer Form angegeben und kann stückweise linear oder mit kubischen Splines interpoliert werden.

Die Simulation erfolgt durch Integration der Zustandsgleichung 3.1. Für die Integration stehen das Eulersche Integrationsverfahren und das Integrationsverfahren Runge-Kutta 4ter Ordnung zur Verfügung. Sie werden ausführlich im Anhang A besprochen.

Die folgenden Randbedingungen (siehe Kapitel 3.1.1) können für Netzwerke mit translationsinvarianten Gewichten gewählt werden. Einerseits können alle virtuellen Zellen auf $+1$, 0 (auch *neutral* oder *Zero-Flux* genannt) oder -1 gesetzt werden. Weiterhin kann auch eine periodische Bedingung oder eine periodische Bedingung mit einem Versatz um $+1$ beim Wechsel von der rechten Rand zum linken Rand gesetzt werden (*geschlossene Spirale in x-Richtung*). Die letzte Möglichkeit besteht aus der Wahl der *von Neumann Bedingung*. Die virtuellen Zellen werden mit den Werten der entsprechenden Randzellen gleichgesetzt.

Die Zellen können mit invarianten oder varianten Widerständen R und Kondensatoren C initialisiert werden. Üblicherweise werden diese Werte in der Zustandsgleichung auf den Wert 1 gesetzt.

Alle Zellenwerte werden als Fließkommazahlen mit doppelter Genauigkeit gespeichert und verarbeitet. Für die Simulation eines Hardware CNN können jedem Zelleingang, Zellstatus, Widerstand und Kondensator Zufallswerte (gaußverteilt) additiv überlagert werden, um damit mögliche Bauteiltoleranzen nachzuvollziehen.

Die Templates A und B können translationsinvariant oder translationsvariant gewählt werden. Jedes Template kann durch ein Polynom mit beliebiger Ordnung oder tabellarisch mit stückweise linearer oder kubischer Spline-Interpolation beschrieben werden. Für ein polynomiales A Template mit Ordnung Q gilt

$$a_{kl}^{(Q)}(y_{kl}(\tau)) = \sum_{q=1}^{Q} a_{q,kl}(y_{kl}(\tau))^q \tag{3.7}$$

Damit entspricht $a_{kl}y_{kl}(\tau)$, aus der Zustandsgleichung 3.1, der Notation $a_{kl}^1(y_{kl}(\tau))$. Für ein polynomiales B Template mit Ordnung V gilt entsprechend

$$b_{op}^{(V)}(u_{op}) = \sum_{v=1}^{V} b_{v,op}(u_{op})^v \tag{3.8}$$

SCNN sichert Template- und Ausgabefunktion-Strukturen in Standard-ASCII Dateien. *Bilder* (gebräuchliche Bezeichnung bei Nutzung von zweidimensionalen CNN-Topologien), mit denen die Eingabe, der Status, der Schwellenwert, die Referenz bei der Optimierung, die Ausgabe und die Werte der Widerstände und Kondensatoren beschrieben werden, können in eigenen Fließkommaformaten (ASCII oder binär) oder in Bildformaten (PBM und PGM) gespeichert und geladen werden. Dabei ist den Fließkommaformaten Vorzug zu geben, da

die Bildformate nur schwarz-weiß (2 Bit Informationen) oder Graustufenbilder (8 Bit Informationen, 256 Stufen) erzeugen. Dies führt zum Verlust der Genauigkeit, die durch die Auslegung der inneren Stukturen auf doppelte Fließkommagenauigkeit gegeben ist.

3.2.2 Simulation

Die Simulation mit SCNN bedeutet die Topologie des CNN und dessen Templates A und B, den Schwellenwert Z, eine Eingabe U und einen Status $X(0)$ festzulegen, die Einstellungen des Simulators wie z.B. den gewünschten Integrationsalgorithmus zu wählen und die Zustandsgleichung 3.1 numerisch zu lösen. Das Ergebnis nach einer gewählten Integrationszeit (*Berechnungsdauer*) τ_{trans} steht dann in der Ausgabe $Y(\tau_{trans})$. τ_{trans} wird durch die Schrittweite h (siehe Anhang A) des Integrationsverfahrens und die Anzahl der Simulationsschritte N_{sim} bestimmt:

$$\tau_{trans} = h * N_{sim} \tag{3.9}$$

Dieses Verhalten entspricht dem der schaltungstechnischen Realisationen von CNN. Hier wird auch nach einer Zeit $\tau_{transChip}$ der Chip *angehalten* und die Ausgabe $Y(\tau_{transChip})$ ausgelesen.

Im Anhang F.2 wird detailiert auf ein Beispiel zur Simulation eines AND-CNN (siehe auch Kapitel 3.1.2) eingegangen.

3.2.3 Optimierung

Die Konfigurationen der wenigsten CNN, die für spezielle Probleme genutzt werden können, lassen sich analytisch lösen, so dass die A und B Templates und der Schwellenwert Z nur noch aus der Lösung der Zustandsgleichung 3.1 abgelesen werden müssen. Statt dessen muss hier fast immer, wie bei den allgemeinen künstlichen neuronalen Netzen, die Technik der Optimierung (Training) genutzt werden (siehe Anhang B). Natürlich muss vorher das Problem in eine Darstellungsform gebracht werden, die im CNN abbildbar ist.

SCNN bietet die Möglichkeit des Trainings. Aus einer Ausgangskonfiguration (eine Eingabe U, ein Status $X(0)$ und der dazugehörigen *Referenz* Y^{Ref}) können durch Nutzung von Optimierungsverfahren die *Parameter* (die Templates A und B und der Schwellenwert Z) bestimmt werden. Dabei können auch einige der Template-Werte festgehalten und nur eine Untermenge trainiert werden. Zusätzlich können auch zu erhaltende Symmetrien (z.B. Punkt, Stern, Kreuz) im Template festgesetzt werden. Dabei ist das Training nicht nur auf Templates und den Schwellenwert beschränkt, sondern kann auch die tabellarisch abgelegten Koeffizienten der Ausgabefunktion und/oder der Templates bestimmen. Training bedeutet die Minimierung der folgenden Funktion:

$$Err^{global} \stackrel{!}{=} \min \left[FCompare(Y(\tau_{trans}), Y^{Ref}) \right] \tag{3.10}$$

$FCompare$ bewertet die gewonnene Ausgabe $Y(\tau_{trans})$ und die Referenz Y^{Ref}. Diese *Bewertungsfunktion* liefert den Wert null, wenn Ausgabe und Referenz identisch sind und den Wert eins, wenn sie völlig unterschiedlich sind (siehe auch Gleichung B.9). Err^{global}, auch *Systemfehler* genannt, stellt den Verlauf des Trainings dar: Für jeden Trainingsschritt

n_{train} (mit N_{train} als maximale Anzahl von Trainingsschritten) rekonfiguriert das Optimierungsverfahren die Parameter um ein möglichst kleines Err^{global} zu finden. Also bedeutet Training die Suche nach einem Minimum in einem *Parameterraum* (*Fehlergebirge*), dessen Dimension durch die Anzahl der zu optimierenden Parameterwerte bestimmt wird.

Im Anhang B werden die in der vorliegenden Dissertation genutzten Optimierungserfahren ausführlich besprochen. Es handelt sich um *Iterative-Annealing*, *Powell's Linienminimierung*, das *Downhill-Simplex* und das *Evolutionäre* Optimierungsverfahren.

Im Anhang F.3 werden die in SCNN zur Verfügung stehende Bewertungsverfahren ausführlich besprochen. Im Rahmen dieser Dissertation wurde hauptsächlich die Bewertungsfunktion $MSEn$ (normierter Mean-Square-Error) genutzt

$$MSEn(Y(\tau_{trans}), Y^{Ref}) = \frac{1}{4N_{Bildpunkte}} \sum_{i=1}^{N_{Bildpunkte}} \left(y_i(\tau_{trans}) - y_i^{Ref} \right)^2 \tag{3.11}$$

mit $N_{Bildpunkte}$ als Anzahl der Zellen bzw. *Bildpunkte* die im CNN vorhanden sind. Diese Bewertungsfunktion verwendet die Mittelung über die quadratischen Abstände der Grauwerte aller Bildpunkte. Problematisch ist diese Bewertungsfunktion, wenn Strukturen in der Ausgabe und Referenz verglichen werden sollen. Die Mittelung verwischt diese Strukturen, so dass der Vergleich als gut bewertet wird, auch wenn einige Bildpunkte einen sehr großen Grauwertabstand haben. Dieser Effekt wirkt noch stärker, wenn die Bildgröße erhöht wird.

Das oben genannte Training (Minimierung der Gleichung 3.10) berücksichtigt nur eine Trainingsmenge (siehe Definition im Anhang B.1) mit einem Element und damit nur eine *Eigenschaft*, die erlernt werden soll. Kann das Problem, welches dem CNN beigebracht werden soll, nur durch eine Trainingsmenge mit Mächtigkeit > 1 dargestellt werden, so muss das Trainingskonzept folgendermaßen verändert werden: Es gibt nicht mehr nur eine Ausgangskonfiguration (bestehend aus der Eingabe U, dem Status $X(0)$ und der Referenz Y^{Ref}), sondern N_m (Mächtigkeit der Trainingsmenge). Also liefert jeder Trainingsschritt N_m *lokale* Bewertungen Err_i^{lokal} ($i \in [1, N_m]$) mit

$$Err_i^{lokal} = FCompare(Y_i(\tau_{trans}), Y_i^{Ref}) \tag{3.12}$$

Mit einer *Zusammenführungsfuktion* $FCompose(\forall i \in [1, N_m], Err_i^{local})$ können so alle Err_i^{lokal} zu einem einzigen Wert Err^{global} zusammengeführt werden (siehe Gleichung 3.13), so dass in diesem erweiterten Training der Optimierungsalgorithmus für alle N_m Elemente der Trainingsmenge *gleichzeitig* nach den optimalen Paremeter A, B und Z sucht.

$$Err^{global} \stackrel{!}{=} \min \left[FCompose(\forall i \in [1, N_m], Err_i^{local}) \right] \tag{3.13}$$

Im Anhang F.4 werden die in SCNN zur Verfügung stehende Zusammenführungsfunktionen ausfürlich besprochen. Im Rahmen dieser Dissertation wurde die Zusammenführungsfunktionen *Mittelwert* und *Median* genutzt. Der Mittelwert als Zusammenführungsfunktion verwischt alle Fehlergebirge durch eine Mittelung. Das kann problematisch werden, wenn ein Minumum eines Elements der Trainingsmenge durch die Mittelung angehoben und dadurch bei der Optimierung übersehen wird. Der Median als Zusammenführungsfunktion springt von dem Fehlergebirge eines Elements der Trainingsmenge in das des nächsten.

Das Optimierungsverfahren kann u.U. keinen gleichmässigen Abstieg schaffen, wenn die Fehlergebirge sich stark unterscheiden.

Im Anhang F.5 wird detailiert auf zwei Beispiele zum Thema SCNN Training eingegangen.

Zusammenfassend läßt sich sagen: Das SCNN-System ist ein hervorragendes Werkzeug, mit dem die verschiedensten CNN getestet werden können. Dem Aufbau der CNN scheinen keine Grenzen gesetzt, allerdings bilden die Speicherkapazität des Rechners und die Geschwindigkeit des Prozessors *natürliche* Grenzen bei der Lösung von Problemstellungen.

Das SCNN-System bietet die Möglichkeit, die verschiedensten Parameter durch implementierte Optimierungsverfahren zu trainieren. Dabei kann eine Trainingsmenge mit Mächtigkeit > 1 trainiert werden.

Ausführliche *aktivierbare* Ausgabedateien ermöglichen eine detailgenaue Beobachtung der Simulations- und Trainingsszenarien.

3.3 Aladdin CNN System

Aufgrund des Paradigmas der lokalen Kopplung konnte ein 64×64 CNN mit 3×3 A und B Templates und variantem/invariantem Schwellenwert, in eine schaltungstechnische Realisation (Chip) überführt werden, den *ACE4K*.

Das Alladin CNN System ([AC00],[ACS00]), das den ACE4K enthält, ist die erste Realisation einer *CNN Universal Machine* (*CNN-UM*), eines CNN mit erweiterten Fähigkeiten dieser Topologie. Das Konzept der CNN wurde mit der Möglichkeit, Bilddaten, Templates und Verarbeitungs-Anweisungen lokal im System abzuspeichern, erweitert. Dadurch entstand eine leistungsfähige Chip-Umgebung, ein vollständiges Mikroprozessorsystem, das CNN-UM. Nachdem Daten und Anweisungen geladen wurden, kann es selbständig die gestellten Aufgaben erfüllen, die aus mehreren CNN-Operationen bestehen können.

Die folgenden Abschnitte schaffen einen Überblick über das System und stellen erste Aufgaben und Ergebnisse vor. Natürlich bleibt die Entwicklung nicht stehen. Mittlerweile stehen das *ACE16K* System mit 128×128 Zellen und das *ACE8K* System mit 64×64 Zellen und zwei Lagen, also ein dreidimensionales CNN, zur Verfügung (siehe Homepage von *Analogic Computers Ltd*, www.analogic-computers.com). Informationen zu älteren CNN-Chips sind z.B. in [Sch02] zu finden.

3.3.1 Konzept

Das zur Verfügung stehende System ist der *Aladdin Visual Computer Stack*. Er besteht aus mehreren Platinen, die in Sandwich-Bauweise übereinander angeordnet in den PCI-Steckplatz eines Standard Pentium-PCs eingebaut wurden. Dieses System ist eine Hochgeschwindigkeits-Verarbeitungseinheit, bestehend aus zwei Prozessoren, dem ACE4K und einem *Digitalen Signal Prozessor* (*DSP*). Der ACE4K kann durch seine parallele Architektur

bis zu 1000 CNN-Operationen pro Sekunde verarbeiten. Die Weiterverarbeitung und die Ein- und Ausgabe-Operationen werden vom DSP durchgeführt.

Der ACE4K ist ein in CMOS (0.8μ) gefertigter Chip. Er enthält ein CNN mit einer 64×64 Topologie. Durch die zweidimensionale Topologie der ACE4K werden die Daten in Form von Bildern verarbeitet. Die Werte jeder Zelle (Bildpunkt) haben eine Auflösung von etwa 7.6 Bit (128 bis 256 Stufen). Der Chip enthält 32 lokale Templatespeicher und 4 weitere für Graustufen- und Binärbilder. Damit ist eine hohe Ausführungsgeschwindigkeit von CNN-Operationen gewährleistet. Zusätzlich ist in dem System auch ein Software-CNN vorhanden. D.h. alle Operationen können vorher auf einem *idealen* CNN getestet werden.

Tabelle 3.1 listet weitere technische Eigenschaften auf. Die zu bearbeitenden Daten, die Bilder, können entweder von einem Datenträger oder direkt von einem optional erhältlichen *Frame-Grabber* eingelesen werden. Die letztere Quelle ermöglicht eine Echtzeitverarbeitung der Daten.

System	Desktop PC
Bus	PCI, 32 Bit Datentransfer
DSP Taktfrequenz	40 MHz
OnBoard Speicher	16 MB
Transferraten in Bildern pro Sekunde	
Bild laden (analog)	2273
Bild ausgeben (analog)	2300
Bild laden (digital)	21550
Bild ausgeben (digital)	22520
Ausführungsgeschwindigkeiten	
Template Operation	9 ms
Logik Operation	3.8 ms

Tabelle 3.1: Technische Daten ACE4K Visual Stack System

3.3.2 Berechnung und Simulation

CNN-Edit ist ein universeller Editor, in dem Programme für den Stack geschrieben und auch übersetzt werden können. CNN-Run führt dann die gewünschten Operationen auf dem Stack aus. Dabei kann gewählt werden, ob der ACE4K (Berechnung der Ausgabe) oder der Software-Simulator (Simulation der Ausgabe) als CNN genutzt werden soll. Zur Programmierung und Nutzung von CNN-Operationen steht eine umfangreiche Template-Bibliothek zur Verfügung ([RK99]). Im Anhang G.3 ist die Programmierung des Aladdin Systems anhand eines Beispieles ausführlich dargestellt.

Das Bild 3.8 verdeutlicht einige Operationen, die mit dem Stack ausgeführt werden können. Die Geschwindigkeit, mit der diese Operationen durchgeführt werden, liegt etwa 2000 mal über der eines aktuellen Pentium4 Systems mit 2 GHz Taktfrequenz. Der ACE4K kann zunächst nur 64×64 Pixel große Bilder verarbeiten. Wird aber das implementierte

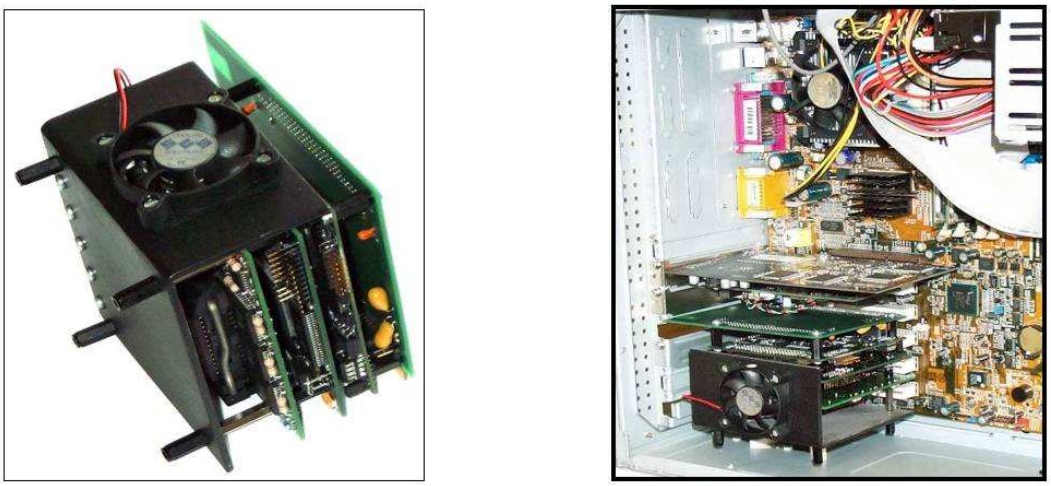


Abbildung 3.7: Aladdin Visual Computer Stack (siehe *http://www.analogic-computers.com*)

Kacheln (*tiling*) genutzt, so können viel größere Bilder verarbeitet werden. Sie werden dann in 64×64 großen Kacheln in den ACE4K übergeben und das Ergebnis-Bild wird automatisch zusammengesetzt.

Die in den Abbildungen 3.9 a.), b.) und c.) dargestellten Bilder verdeutlichen den Unterschied zwischen dem ACE4K und dem Software-Simulator CNN. Die Abbildung 3.9 b.) ist auf dem Simulator entstanden, es stellt eine Kantenextraktion der Abbildung 3.9 a.) dar. Die Abbildung 3.9 c.) zeigt das Ergebnis der Arbeit des ACE4K. Es sind kleine Unterschiede zwischen der Abbildung 3.9 b.) und der Abbildung 3.9 c.) zu entdecken, trotzdem erreicht die schaltungstechinsche Realisation ein gutes Ergebnis. Im Abschnitt 3.3.3 wird dieser Vergleich anhand eines Beispiels durchgeführt.
In Anbetracht der Tatsache, dass der ACE4K nur über 7.6 Bit Genauigkeit pro Bildpunkt verfügt und die 64×64 Matrix aufgrund der schaltungstechischen Realisation nicht homogen ist, ist das Ergebnis beachtlich.

3.3.3 Erster Test des ACE4K

Wie schon oben erwähnt, ist der ACE4K ein Chip, der Fertigungstoleranzen unterworfen ist. Die 64 × 64 Matrix ist nicht homogen, es kann nur eine ungefähre Auflösung von 7.6 Bit für jeden Bildpunkt angegeben werden.

Zunächst wurde ein *Wiederholungs-Test* durchgeführt, um die *Langzeitstabilität* des Stacks zu prüfen und erste Aussagen über die Güte der Ausgabebilder $Y(\tau_{transChip})$ machen zu können. Dafür wurde eine CNN-Operation, das Invertieren von Bildern, im Stack umgesetzt. Die folgende CNN-Konfiguration sowie das in Abbildung 3.10 dargestellte Eingabebild wurden dabei genutzt.

$$A = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & -1.5 & 0 \\ 0 & 0 & 0 \end{Bmatrix} \quad B = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & -1.5 & 0 \\ 0 & 0 & 0 \end{Bmatrix} \quad Z = \{ \ -0.4 \ \} \tag{3.14}$$

Abbildung 3.8: Beispiel-Operationen des Aladdin Visual Computer Stacks (siehe *http://www.analogic-computers.com*)

a.) Eingabe-Bild U

b.) Ausgabe-Bild $Y(\tau_{trans})$ Simulator

c.) Ausgabe-Bild $Y(\tau_{transChip})$ ACE4K

Abbildung 3.9: Vergleich Software-Simulator CNN und ACE4K des Aladdin Visual Computer Stacks (siehe *http://www.analogic-computers.com*)

Danach wurde diese Operation 4750 mal automatisch wiederholt und die resultierenden

a.) Eingabe-Bild U | b.) Negativ-Bild (Referenz) Y^{ref} | c.) Ausgabe-Bild $Y(\tau_{transChip})$ ACE4K

Abbildung 3.10: Invertieren mit CNN am Beispiel der Mona Lisa

Ausgabe-Bilder $Y(\tau_{transChip})$ gespeichert.
Es fällt auf, dass das Negativ-Bild Y^{ref} in Abbildung 3.10 b.) und das beispielhaft ausgewählte Ausgabe-Bild $Y(\tau_{transChip})$ in Abbildung 3.10 c.) nicht übereinstimmen. Das wurde mit dem Bildvergleichsmaß PE (siehe Anganf F.3) quantifiziert. Im nachfolgenden Schritt wurden alle gewonnenen Ausgabe-Bilder $Y(\tau_{transChip})$ mit dem Negativ-Bild Y^{ref} verglichen. Die Abbildung 3.11 stellt den Verlauf des Vergleichs über alle gewonnenen Ausgabe-Bilder dar. Zunächst bestätigt der Graph die Aussage, dass die 64×64 Matrix des ACE4K nicht homogen ist. Wäre dies der Fall, dann würde der Graph einer waagerechte Linie gleichen, die im Idealfall eines vollständig homogenen ACE4K konstant 0 ist. Zu erkennen ist eine Verschiebung (Offset$\neq$0). Dies hängt mit der geringen Auflösung zusammen. Gleichzeitig ist aber auch eine nicht unerhebliche Schwankung zu erkennen. Diese wiederum hängt mit der nicht genau bestimmbaren und fixierbaren Auflösung zusammen. Hier spielen Effekte, wie die *Wärmeentwicklung und Ausdehnung im Dauerbetrieb* eine große Rolle (siehe auch Kapitel 4.6).

Trotzdem ermutigt dieses Ergebnis, weitere Untersuchungen mit dem ACE4K durchzuführen. Denn ein wichtiges Resultat ist, dass der ACE4K mit *konstantem* Fehlerniveau und *gleichbleibender* Varianz durchgehend funktioniert. Die Schwankung im Graphen liegt im Bereich von etwa $\pm 0.1\%$. Damit liegt sie unter der Auflösungsgrenze einer Zelle.

3.3.4 Erste Ergebnisse

In einem ersten Test der drei CNN Typen, SCNN Simulator, ACE4K und der Simulator des Stacks wurden auch erste Erfahrungen mit den Unterschieden der untersuchten Systeme gesammelt. Die Ansteuerung des ACE4K und des Simulators geschah durch das SCNN System. Ausführliche Informationen dazu sind im Anhang G.2 zu finden.

Der Test bestand aus der CNN-Operation *Invertieren* (siehe Kapitel 3.3.3) und im zweiten Schritt aus einem Nachtrainieren der Parameter. Als Konfiguration für die Eingabe U wurde das Bild in Abbildung 3.10 a.) gewählt, im Status-Bild wurden alle Bildpunkte auf 0 gesetzt. Das Ausgabe-Bild $Y(\tau_{trans})$ bzw. $Y(\tau_{transChip})$ sollte dem Negativ-Bild Y^{ref} in

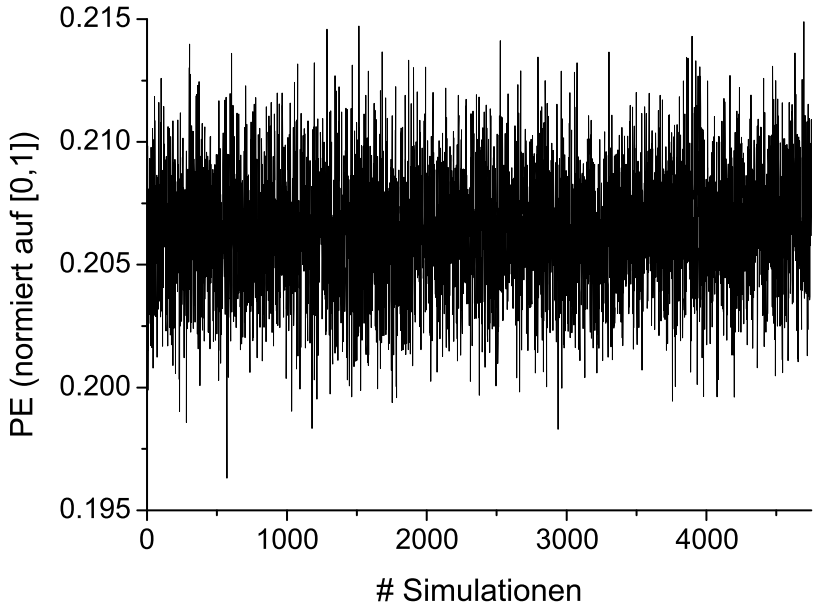


Abbildung 3.11: Ausgabeverhalten des ACE4K beim Wiederholungs-Test

Abbildung 3.10 b.) entsprechen.

Schritt 1: Simulation

Tabelle 3.2 verdeutlicht die Qualität der Ergebnisse der simulierten Ausgaben mit dem SCNN-System und dem Simulator des Stacks und die berechnete Ausgabe des ACE4K.

CNN-Typ	$PE(Y^{ref}, Y(\tau_{trans}))$ bzw.$PE(Y^{ref}, Y(\tau_{transChip}))$
SCNN-Simulaor	0.0457133
Simulator des Stacks	0.0627281
ACE4K	0.155947

Tabelle 3.2: Simulationsfehler der CNN-Operation Invertieren am Bild in Abbildung 3.10

Das Bildvergleichsmaß PE kann als durchschnittlicher prozentualer Fehler eines Bildpunktes im Vergleich zur Referenz gedeutet werden. Tabelle 3.2 zeigt, dass die beiden Simulatoren etwa dreimal bessere Ergebnisse lieferten als der Chip.

Die Templates und der Schwellenwert wurden aus der Template-Bibiliothek ([RK99]) entnommen. Damit wurden *generelle* Parameter genutzt, die noch nicht auf die einzelnen CNN optimiert waren. Auf diese Weise läßt sich der geringe Unterschied erklären.

Schritt 2: Nachtraining

Die generellen Parameter der CNN-Operation Invertieren (siehe Parameter 3.14) mit der Eingabe U (siehe Abbildung 3.10 a.)), den Status 0 sowie den Negativ-Bild (siehe Abbildung 3.10 b.)) als Referenz Y^{ref} wurden für die beiden Simulatoren und den ACE4K trainiert und damit speziell auf die Eigenschaften der Systeme angepaßt.

Das Training (siehe Kapitel 3.2.3) wurde auf 1300 bzw. 1500 Trainingsschritte eingestellt. Als Trainingsverfahren wurde Iterative-Annealing (siehe Anhang B.1), mit dem Startwert $T0 = 10$ und 100 Abkühlschritten gewählt. Alle $9 + 9 + 1 = 19$ Parameter standen dem Optimierungsalgorithmus zur Verfügung. Als Bewertungsfunktion wurde PE genutzt. Um das Problem des ACE4K mit seinen geringen Parametergrenzen im Bereich $[-3, 3]$ zu umgehen, wurde zunächst für das Iterative-Annealing das *Abschneiden hoher Werte* im SCNN aktiviert. Alle vom Optimierungsverfahren vorgeschlagenen Werte > 3 (bzw. < -3) wurden auf 3 (bzw. -3) gesetzt. Dieser Eingriff erlaubte es, in einem n-Kubus (n=Anzahl der Parameter) mit den oben genannten Grenzen nach einem globalen Minimum zu suchen. Zusätzlich wurde die Linienminimierung nach Powell's (siehe Anhang B.2) für das Training des ACE4K genutzt.

Die Tabelle 3.3 stellt die Trainingsergebnisse dar.

CNN-Typ	Err^{global}
SCNN-Simulator (Iterative-Annealing)	0.00274689
Simulator des Stacks (Iterative-Annealing)	0.0130005
ACE4K (Iterative-Annealing)	0.720709
ACE4K (Powell's)	0.0718055

Tabelle 3.3: Bewertung des Training der CNN-Operation Invertieren für verschiedenen CNN-Typen SCNN-Simulator, Simulator des Stacks und ACE4K

Zunächst ist im Vergleich zur Tabelle 3.2 zu erkennen, dass die durch das Training gewonnenen Paramter eine Verbesserung von Err^{global} erbracht haben. Die einzige Ausnahme ist der ACE4K in Verbindung mit Iterative-Annealing. Hier hat die Einschränkung der Parameterwerte das Optimierungsverfahren fehlgeleitet.

Die anderen Ergebnisse sind um so positiver, da der SCNN-Simulator jetzt ein zwanzigmal besseres Ergebnis liefert. Der Simulator des Stacks steigerte sich um den Faktor fünf und der ACE4K (in Verbindung mit der Powell's Linienminimierung) etwa um den Faktor zwei.

Auch der Unterschied zwischen den drei Implementierungen erwies sich größer als zuvor: Der SCNN-Simulator ist etwa sechsmal besser als der des Stacks und 26 mal besser als der ACE4K. Der Simulator des Stacks ist etwa sechsmal besser als der ACE4K.

Obwohl der Unterschied größer ausfiel, ist es ermutigend, dass die CNN-Operation auch auf dem ACE4K recht gut optimiert werden konnte. In diesen Trainings wurde nur ein invarianter Schwellenwert trainiert. Würde ein varianter Schwellenwert genutzt, so könnte das Ausgabeverhalten auch an die Inhomogenität der ACE4K Matrix angepaßt werden. Dies würde den Fehler noch weiter reduzieren. Allerdings würde sich die zu optimierende

Parameteranzahl auf $9+9+4096 = 4114$ erhöhen, da jede Zelle ihren eigenen Schwellenwert hätte.

Die resultierenden Parameterwerte (siehe Gleichungen 3.15 bis 3.17, 3.18 bis 3.20, 3.21 bis 3.23 und 3.24 bis 3.26) zeigen eine Struktur, die der der generellen Parameterwerte (siehe 3.14) ähnlich ist. Allerdings darf nicht vergessen werden, dass in diesem Fall an *einem* Bild-Tripel die Parameter für die jeweiligen CNN optimiert wurden. Da nur ein Bild-Tripel genutzt wurde, muss davon ausgegangen werden, dass das Invertieren mit diesen Parametern bei anderen Bildern viel schlechter von statten geht als mit den generellen Parametern. Es ist sogar wahrscheinlich, dass diese Operation bei vielen anderen Bildern fehlschlagen wird.

$$A = \left\{ \begin{array}{ccc} -0.151226404111 & -.287324220381 & 0.0421300010928 \\ -0.31330527433 & -1.29223418269 & -0.273865012244 \\ -0.0995505985185 & -0.295454676778 & 0.0642675368747 \end{array} \right\} \tag{3.15}$$

$$B = \left\{ \begin{array}{ccc} -0.187328455941 & -0.267683682896 & 0.0254604273816 \\ -0.308639070023 & -1.8726324183 & -0.248243681304 \\ -0.110463784769 & -0.261740220292 & 0.0409356515026 \end{array} \right\} \tag{3.16}$$

$$Z = \left\{ \begin{array}{c} -0.00939882 \end{array} \right\} \tag{3.17}$$

1. Trainingsergebnis für den SCNN-Simulator (Iterative-Annealing)

$$A = \left\{ \begin{array}{ccc} 0.381813945231 & 0.253586864869 & -0.640108931828 \\ -0.19502014447 & -2.00136430041 & 0.0387551896788 \\ -0.288983157304 & 0.211963723955 & 0.582308412675 \end{array} \right\} \tag{3.18}$$

$$B = \left\{ \begin{array}{ccc} 0.48943546662 & -0.214331844556 & -0.317087173289 \\ -0.123040465312 & -2.45006578121 & 0.0722200485813 \\ -0.497540006148 & 0.168982668549 & 0.205219097584 \end{array} \right\} \tag{3.19}$$

$$Z = \left\{ \begin{array}{c} -0.0381458 \end{array} \right\} \tag{3.20}$$

2. Trainingsergebnis für den Simulator des Stacks (Iterative-Annealing)

$$A = \left\{ \begin{array}{ccc} 3 & 1.0063 & -3 \\ 1.71814 & -1.78046 & 1.90594 \\ -1.39374 & -0.594087 & 2.82745 \end{array} \right\} \tag{3.21}$$

$$B = \left\{ \begin{array}{ccc} 3 & 3 & -3 \\ 2.54779 & 3 & -3 \\ -3 & 1.80129 & 3 \end{array} \right\} \tag{3.22}$$

$$Z = \left\{ \begin{array}{c} 0.22857 \end{array} \right\} \tag{3.23}$$

3. Trainingsergebnis für den ACE4K (Iterative-Annealing)

$$A = \left\{ \begin{array}{ccc} -1.053539 & 0.379260 & -0.069646 \\ -0.072061 & -1.984227 & -0.013008 \\ -0.071930 & 0.283356 & 0.074762 \end{array} \right\} \tag{3.24}$$

$$B = \left\{ \begin{array}{ccc} 0.002995 & -0.041262 & -0.003063 \\ 0.001054 & -1.463940 & -0.037191 \\ -0.005974 & 0.090776 & -0.033176 \end{array} \right\} \tag{3.25}$$

$$Z = \left\{ \begin{array}{c} -0.171643 \end{array} \right\} \tag{3.26}$$

4. Trainingsergebnis für den ACE4K (Powell's Linienminimierung)

Dies ist das *Hauptproblem* des Trainings von künstlichen neuronalen Netzen: Wird mit einer zu kleinen Trainingsmenge gearbeitet, kann es zu einer Überspezialisierung kommen. Das neuronale Netz kann nur diese Trainingsmenge optimal lösen. Eine unabhängige Testmenge bereitet große Probleme oder ist vielleicht sogar völlig unlösbar. Der Abstieg in den Fehlerverläufen (siehe Abbildungen 3.12 a.) bis d.) (Err^{global} ist im Bereich $[0, 1]$ definiert, wird hier aber der Übersichtlichkeit halber im Bereich $[0, 0.5]$ dargestellt)) geht langsam von statten. Bei der Wahl einer größeren Anzahl von Trainingsschritten würden bessere Ergebnisse entstehen, da dem Optimierungsverfahren mehr Möglichkeiten zur Suche geben werden. Allerdings sind auch hier Grenzen gesetzt. Diese Grenzen werden im ACE4K durch seine 7.6 Bit Auflösung und in den Simulatoren durch die Rechengenauigkeit und damit verbundene Steigerung der Rechenzeit des Prozessors gesetzt.

Zusammenfassend läßt sich sagen: Der ACE4K stellt ein interessantes Werkzeug dar. Er ist ein technisches neuronales Netz, als Koprozessor realisiert. Die im Rahmen dieser Dissertation erfolgte Anbindung an das SCNN-System ermöglichte eine einfache Nutzung in der Berechnung von CNN-Operationen und im Training. Allerdings können die vorgegebene Einschränkungen bzw. Probleme, wie die geringe Auflösung der Bildpunkte, die Inhomogenität der Bildmatrix, die Einschränkungen der Topologie des Netzes und der möglichen Parameterwahl die Nutzung beeinträchtigen. Diese Einschränkungen wurden am Beispiel des Ausgabeverhaltens bei wiederholten Anwendungen einer CNN-Operation am ACE4K demonstriert.

Dieses Kapitel zeigte auch die ersten Anwendungen des Parameter-Trainings. Dabei wurden die drei möglichen CNN im SCNN-System (der eigene Simulator, der ACE4K und der Simulator des Stacks) anhand der CNN-Operation Invertieren verglichen. Die oben genannten Probleme des ACE4K kamen dabei ein weiteres Mal zur Geltung. Es stellte sich heraus, dass der ACE4K Ergebnisse berechnete, die etwa eine Größenordung schlechter bewertet wurden als die der Simulatoren. Trotzdem sind Verbesserungen durch Trainingsmaßnahmen möglich. Natürlich können auch weitere schaltungstechnische Realisationen von CNN genutzt werden, wie z.B. das in [LPKH04] erwähnten CNN mit 72×72 Topologie und Template-Polynomen bis zur 3. Ordnung.

Die Aufgabenstellung der folgenden Kapitel, CNN zur Untersuchung von Synchronisationsphänomenen zu trainieren, wurde zunächst auf dem Simulator SCNN durchgeführt.

Die Untersuchung von Synchronisationsphänomenen wurde durch die Approximation der mittleren Phasenkohärenz R mit CNN durchgeführt. Durch seine Optionsvielfalt erlaubt das SCNN-System eine einfache Konfiguration von komplexen CNN-Typen, die im Training und der Simulation genutzt werden konnten. Seine Rechengenauigkeit erlaubte eine Abschätzung der Fehler, die durch die Approximation geschehen. Mit den so erstellten und getesteten CNN konnten Anforderungen an die schaltungstechnische Realisationen angegeben werden. Gleichzeitig konnten aber auch vorhandene Hardware-Realisationen (wie z.B. der ACE4K) im Simulator abgebildet werden und der Einfluss von Bauteiltoleranzen modelliert und untersucht werden.

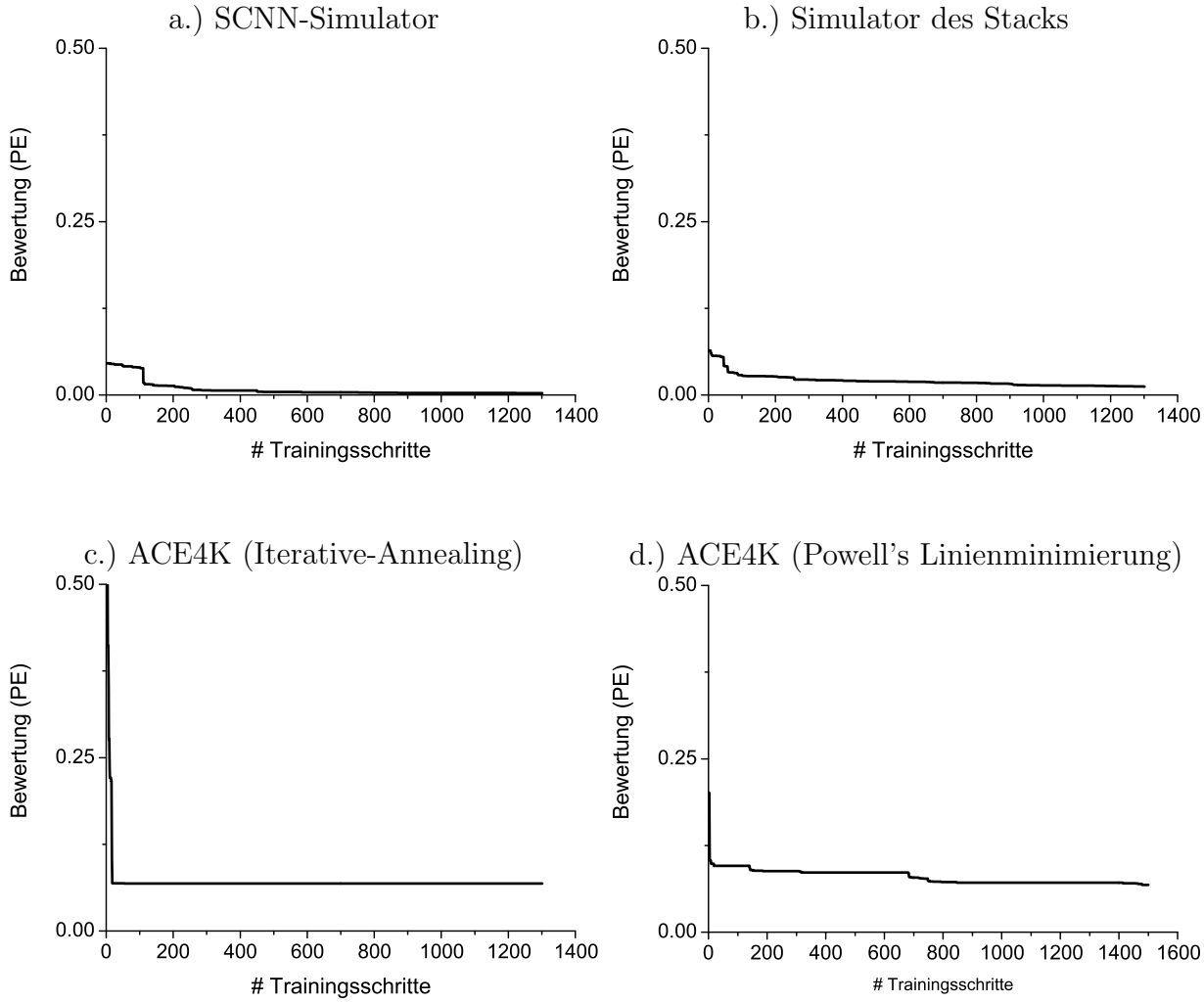


Abbildung 3.12: Err^{global}-Verläufe der vier CNN Realisationen

Kapitel 4

Synchronisationsmessungen mit CNN

In diesem Kapitel werden Synchronisationsmessungen mit CNN vorgestellt. Das Phasensynchronisationsmaß R (siehe Kapitel 1.2) wird dabei als Hilfsmittel zur Charakterisierung der Synchronisation von dynamischen Systemen verwendet. Es wird eine Methodik entwickelt, mit der Zeitreihen aus einer Trainingsmenge in ein CNN übertragen werden, das CNN trainiert wird und die so gewonnenen Parameter an einer Testmenge verifiziert werden. Hierfür werden Zeitreihen aus Modellsystemen und Zeitreihen hirnelektrischer Aktivität, sowie die entsprechenden Synchronisationswerte R, als Datenquellen verwendet. Zum Abschluss dieses Abschnitts wird eine Untersuchung des *Generalisierungsverhaltens* durchgeführt und die Robustheit des gewonnenen CNN gegen modellierte Bauteiltoleranzen analysiert. Im Rahmen der Untersuchung des Generalisierungsverhaltens wird die Approximationsqualität der gewonnenen CNN mit den Testmengen der drei anderen Datenquellen bestimmt.

4.1 Datenbasis

Die Datenbasis bestand aus vier Datenquellen. Die erste Datenquelle (GR) bestand aus synthetischen Zeitreihen, die aus gekoppelten Rössler-Systemen gewonnen wurden. Dieses Modellsytem wurde aufgrund seiner wohlbekannten Eigenschaften gewählt ([QAG00]). In Kapitel 1.3 sind beispielhaft in den Abbildungen 1.1 a.) bis d.) zwei Zeitreihenpaare abgebildet, die durch Lösung der Differentialgleichungssysteme (siehe Anhang D) von zwei verschieden stark untereinander gekoppelten Rössler-Systeme gewonnen wurden.

Die darauf folgende Untersuchung beschäftigte sich mit EEG-Daten. Hierfür wurden drei Datenquellen (P1, P2 und P3), EEG-Aufzeichnungen von drei Epilepsie-Patienten, verwendet. Diese Aufzeichnungen wurden im Rahmen der prächirurgischen Epilepsiediagnostik durchgeführt und wurden an den Elektroden TL und TR abgeleitet (siehe Kapitel 1.4). Dabei wurde die Elektrodenkombination gewählt, mit der der deutlichste Unterschied zwischen interiktalem und hypothetischen prä-iktalen Zustand ausgemacht werden konnte (vgl. Kapitel 1.4.3 und Abbildundungen 1.4 a.)und b.)).

Die Tabelle 4.1 enthält die wesentlichen Informationen zu den genannten Datenquellen. Die hirnelektrische Aktivität wurde in allen drei Fällenmit einer Frequenz von 173.611 Hz abgeleitet. Mit der Bezeichnung 0p00 bis 0p04 der Datenquelle GR wurde die Einstellung der Kopplungsstärke ϵ (Werte von $\epsilon = 0$ bis $\epsilon = 0.04$) benannt. In der Zeitreihe GRc

wurde die Kopplung linear von $\epsilon = 0$ (für den ersten Datenpunkt) bis $\epsilon = 0.04$ (für den letzten Datenpunkt) erhöht. Die mit einem * markierten Zeitreihen (P1f, P1j, P2d und P3f) enthielten die Aufzeichnung eines Anfalls. Die Zeitreihen P1e bis P1f, P1g bsP1j und P2b bis P2d wurden zeitnah bzw. hintereinander aufgezeichnet und enthielten einen hypothetischen prä-iktalen Zustand. Die interiktalen Zeitreihen wurden zum Teil an verschiedenen Tagen aufgenommen.

	gek. Rössler (*GR*)	Patient 1 (*P1*)		Patient 2 (*P2*)		Patient 3 (*P3*)	
	Art	Kanäle TL01-TL02		Kanäle TL04-TL05		Kanäle TL01-TL02	
		Art	Länge min:sek	Art	Länge min:sek	Art	Länge min:sek
a	0p00	interiktal	34:52	interiktal	15:13	interiktal	20:55
b	0p01	interiktal	29:52		20:56	interiktal	20:55
c	0p00-0p04	interiktal	19:54		19:55	interiktal	20:55
d	0p02	interiktal	29:43	peri-iktal *	20:45	interiktal	25:53
e	0p03		9:12			interiktal	25:53
f	0p04	peri-iktal *	44:40			peri-iktal *	50:38
g			29:53			interiktal	25:53
h			22:00				
i			49:49				
j		peri-iktal *	31:46				
k		interiktal	47:48				
		Gesamtlänge 349:29		Gesamtlänge 76:49		Gesamtlänge 191:02	

Tabelle 4.1: Informationen über die vier verwendeten Datenquellen GR und P1 bis P3

4.1.1 Datenvorverarbeitung

Entsprechend der Vorbereitung der Daten für die Berechnung des Synchronisationsmaßes mittlere Phasenkohärenz R (vgl. Kapitel 1.2) wurden die Daten von GR und P1 bis P3 in Datenfenster mit einer Breite von 4096 Datenpunkte (entspricht etwa 23.6 Sekunden) und 20% Überlagerung zum vorherigen Fenster aufgeteilt. Für jedes Fenster wurde ein Mittelwertabgleich durchgeführt. Die Abbildung 4.1 zeigt beispielhaft das Aufteilen in Datenfenster.

Eine CNN-Zelle kann nur einen Wertebereich von $[-1, 1]$ auflösen. Daher wurden die Daten im letzten Schritt der Vorverarbeitung skaliert. Die generierten Daten GR wurden von Anfang an auf den Wertebereich $[-1, 1]$ eingestellt. Die Datenquellen P1 bis P3 enthielten Werte mit einem Wertebereich von bis zu $[-300\ \text{mV}, 300\ \text{mV}]$. Für die folgende Analyse wurde gefordert, dass mindestens 90% der Datenpunkte einer Zeitreihe zur Verfügung stehen. Mit einer linearen Skalierung des Wertebereiches $[-150\ \text{mV}, 150\ \text{mV}]$ nach $[-1, 1]$ wurde diese Vorgabe für die Datenquellen P1 bis P3 erreicht. Werte größer/ kleiner 150 mV/-150 mV wurden auf -1 bzw. 1 abgebildet. Mit dieser Vorgehensweise wurden Aufzeichnungsartefakte in der Analyse ausgeschlossen. Abbildung 4.2 zeigt beispielhaft die

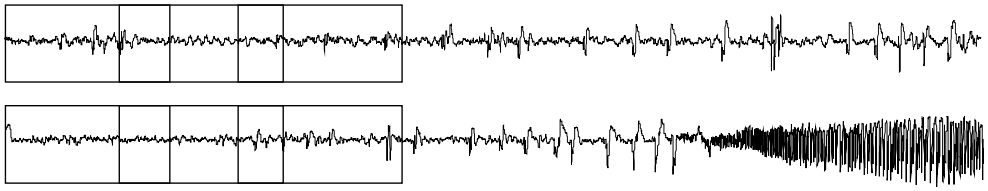


Abbildung 4.1: Beispielhafte Aufteilung zweier Zeitreihen in Datenfenster mit 20% Überlagerung

Amplitudenverteilung der EEG-Zeitreihe P1b, nach der Aufteilung in überlappende Fenster und nach Ausführung des Mittelwertabgleichs.

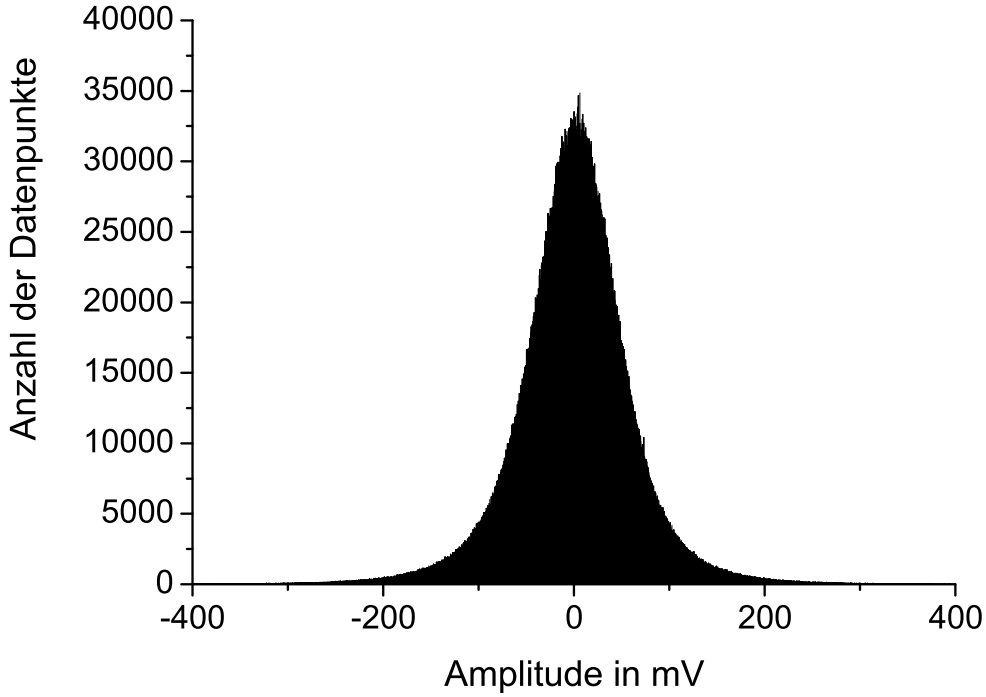


Abbildung 4.2: Amplitudenverteilung der EEG-Zeitreihe P1b

4.2 CNN-Einstellungen

Um Synchronisationsmessungen mit CNN durchführen zu können, wurden die Daten zunächst in ein CNN taugliches Format umgewandelt. Jede der Datenquellen wurde in eine Trainings- und eine Testmenge aufgeteilt. Zur Abschätzung des Synchronisationsgrades der Zeitreihenpaare wurde das Synchronisationsmaß mittlere Phasenkohärenz R als Vergleichsmaß genutzt. Der abgeschätzte Synchronisationsgrad wird im folgenden Text vereinfacht als *Synchronisationswert* bezeichnet. Um eine Synchronisationsmessung durchführen zu können,

wurde zunächst das CNN mit den Elementen aus der Trainingsmenge und den dazugehörigen Synchronisationswerten R trainiert. Durch dieses *überwachte* Training wurde ein CNN gewonnen, mit dem die Synchronisationswerte R approximiert werden konnten. Diese approximierten Synchronisationswerte werden im Folgenden mit R^{approx} bezeichnet. Anhand der Datenfensterpaare und der dazugehörigen Synchronisationswerte innerhalb der Testmenge konnte nun die Approximationsqualität des gewonnenen CNN bestimmt werden.

In den folgenden Abschnitten werden die in der vorliegenden Arbeit überwiegend genutzten Einstellungen der CNN dargestellt. Zunächst wird auf die grundlegenden Simulator- und Trainings-Einstellungen eingegangen. Waren andere Einstellungen notwendig, so werden sie an den entsprechenden Stellen in der Arbeit gesondert beschrieben. Es folgt die Darstellung der drei genutzten CNN-Topologien. Dabei wird detailiert auf den Aufbau der Datenfensterpaare im CNN-Format eingegangen. Zuletzt wird auf den Aufbau der Trainings- und Testmenge, der Synchronisationswerte R, der approximierten Synchronisationswerte R^{approx} und auf die in dieser Arbeit genutzten Evaluationstechniken zur Abschätzung der Approximationsqualität eingegangen.

4.2.1 Simulator

Die primären Einstellungen des Simulators des SCNN Systems betreffen die Kennlinie (Ausgabefunktion des CNN), die Wahl des Integrationsalgorithmus und die Einstellung der Integrationszeit (*Berechnungsdauer*) τ_{trans}. Als Kennlinie wurde eine sigmoide Funktion mit Steigungsparameter $\beta = 4$ genutzt (siehe auch Kapitel 3.2.1). Diese Kennlinie (siehe Abbildung 4.3) ähnelt in ihrem Verlauf den Kennlinien von Transistoren. Mit dieser Einstellung können CNN, die auf Simulatoren gewonnen wurden, leichter in schaltungstechnische Realisationen überführt werden.

Nach Gleichung 3.9 wird die Integrationszeit τ_{trans} durch die Schrittweite h des Integrationsalgorithmus und die Anzahl der Simulationsschritte N_{sim} festgelegt. In der vorliegenden Arbeit wurde $h = 0.2$ und $N_{sim} = 200$ und damit $\tau_{trans} = 40$ gesetzt. Diese Einstellungen wurden erfolgreich bei der Approximation der Korrelationsdimension mit CNN ([Ame98]) genutzt. In schaltungstechnischen Realisationen wird entsprechend vorgegangen, nach einer vorgegebenen Berechnungsdauer $\tau_{transChip}$ wird das Ergebnis $Y(\tau_{transChip})$ ausgelesen.

Als Integrationsalgorithmus wurde das Euler-Verfahren (siehe Anhang A.1) gewählt. Dieser Algorithmus gewährt im Vergleich zum Runge-Kutta Verfahren (siehe A.2) eine ausreichende Genauigkeit bei erheblicher Steigerung der Rechengeschwindigkeit. Für weitere Untersuchungen bezüglich der numerischen Integration der Zustandsgleichung (siehe Gleichung 3.1) sei auf [Ku96] verwiesen. Weitere Untersuchungen zur Wahl der Schrittweite und zur Auswirkung der Kennlinie der Zellen auf die stabilen Endzustände von CNN wurden in [Ge98] ausführlich besprochen. Hervorzuheben ist an dieser Stelle, dass die in [Ge98] untersuchten CNN bei Variation der Schrittweite im Bereich $h \in [0.01, 0.25]$ keine Unterschiede in deren Ausgaben $Y(\tau_{trans})$ enthielten.

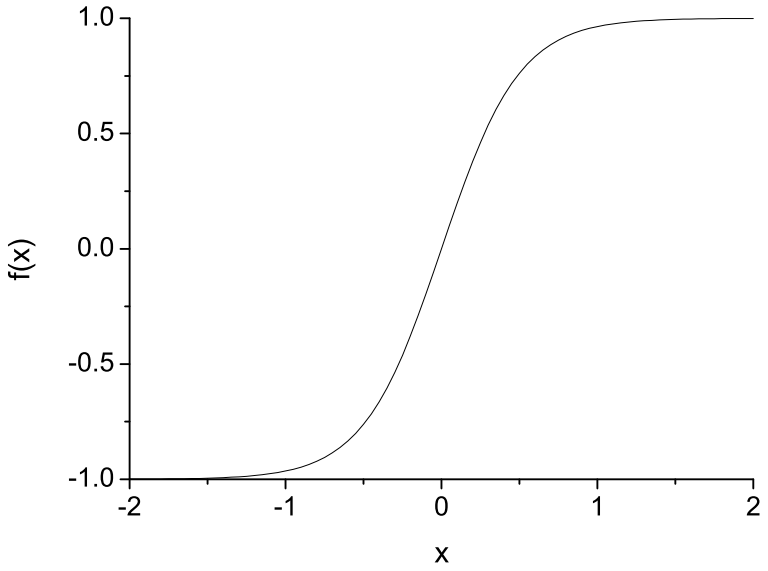


Abbildung 4.3: Sigmoide Ausgabefunktion des CNN mit Steigungsfaktor $\beta = 4$

4.2.2 Optimierung

Im Training wurde hauptsächlich die Bewertungsfunktion $FCompare = MSEn$ (siehe Gleichung 3.11) genutzt. Als Zusammenführungsfunktion wurde der Mittelwert eingesetzt. Zur Optimierung wurde das Iterative-Annealing (siehe Anhang B.1) mit Starttemperatur $T_0 = 50$ und Anzahl der Wiederholungen $j_{max} = 300$ eingesetzt. Insgesamt wurden $N_{train} = 14400$ Trainingsschritte erlaubt. Die optimalen Parameter, die während der N_{train} Trainingsschritte gefunden wurden, wurden am Ende des Trainings gesichert.

Für die A und B Templates wurden die Topologien 3×3, 5×5, 7×7 und 9×9 gewählt. Es wurden auch polynomielle Templates mit Ordnungen von bis zu $Q = V = 4$ und Punkt-, Stern- und Kreuz-Symmetrie zur Verkleinerung bzw. Veränderung des Fehlergebirges zugelassen. Diese Einstellungen wurden immer für beide Templates durchgeführt. Weiterhin wurden ein invarianter Schwellenwert Z sowie die Randbedingungen neutral, periodisch und als geschlossene Spirale in x-Richtung genutzt. Die einzelnen Einträge der Templates und der Schwellenwerts wurden vor dem Training mit mittelwertfreien, gaussverteilten Zufallszahlen mit einer Standardabweichung von $\sigma = 0.2$ initialisiert. In [Ame98] wurde gezeigt, dass sich beim Training von CNN mit nichtlinearen Templates das Trainingsergebnis deutlich verschlechtert, wenn die Standardabweichung größer als $\sigma = 0.2$ gesetzt wird.

4.2.3 CNN-Topologie

Für die vorliegende Arbeit wurden insgesamt drei verschiedene CNN-Topologien im Training verglichen.

In der Topologie 1 (siehe Abbildung 4.4) wurde das Zeitreihensegment der ersten Zeitreihe (*Zr 1*) zeilenweise von links oben nach rechts unten in die Eingabe U und das Zeitreihensegment der zweiten Zeitreihe (*Zr 2*) entsprechend in den Status $X(0)$ überführt. Durch

die Wahl eines 64×64 CNN fanden alle 4096 Datenpunkte eines jeden Segments Platz. In der Topologie 2 (siehe Abbildung 4.5) wurden die Datenpunkte beider Zeitreihensegmente abwechselnd zeilenweise in den Status $X(0)$ geschrieben. Die Eingabe U und das B Template wurden auf 0 gesetzt und nicht verändert. Dieses CNN war 64×128 Zellen groß. Die Topologie 3 (siehe Abbildung 4.6) war weitgehend mit der Topologie 2 identisch. Es wurde allerdings zusätzlich eine Kopie vom Status $X(0)$ in der Eingabe U abgelegt.

Alle drei Topologien richteten sich nach der Vorgabe, die Ergebnisse möglichst einfach auf schaltungstechnische Realisationen (z.B. ACE4K CNN mit 64×64 Topologie) übertragen zu können. Wird das Verfahren des Kachelns genutzt (siehe Kapitel 3.3.2), kann auch die 64×128 Topologie im ACE4K CNN genutzt werden.

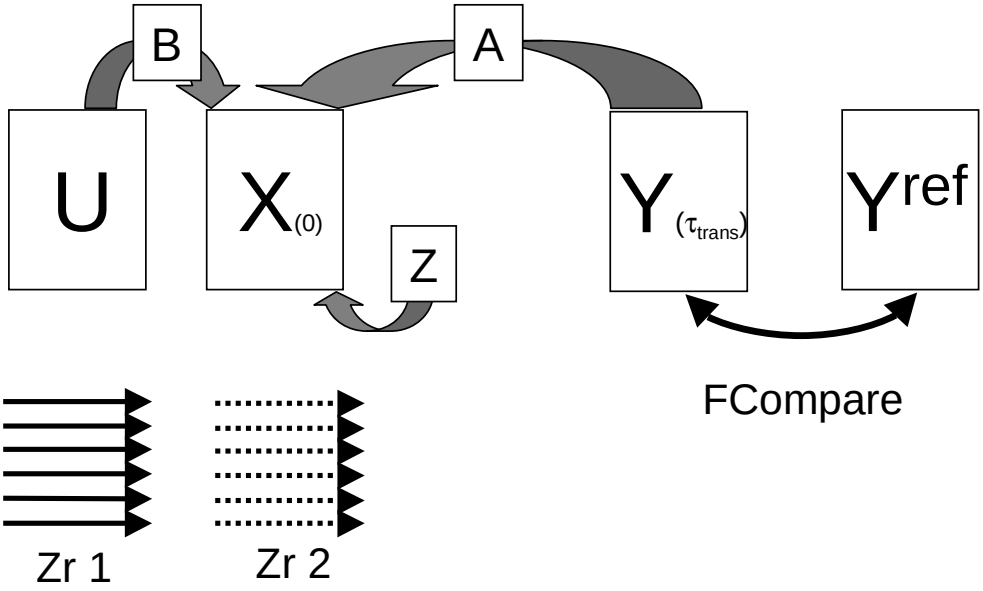


Abbildung 4.4: Topologie 1 mit einem 64×64 CNN

4.2.4 Zusammensetzung der Trainings-/Testmenge

Für das Training wurden zufällig Datenfensterpaare und die entsprechenden Synchronisationswerte R ausgewählt. Die Abbildungen 4.7 a.) bis d.) stellen Histogramme der Synchronisationswerte R der Datenquellen GR, P1, P2 und P3 dar. Für die Anwendung mit EEG-Daten (Datenquellen P1 bis P3), wurden Synchronisationswerte im Bereich $R \in [0.3, 0.9]$ betrachtet. Kleinere Werte können, aufgrund der geringen Datenpunktezahl von 4096 pro Datenfenster und der damit verbundenen Varianz der Synchronisationswerte, nicht sauber aufgelöst werden. Größere Werte sind nur in hoch synchronisierten Systemen vorhanden und waren damit in diesen Untersuchungen nicht von Interesse. Diese Synchronisationswerte entsprechen auch dem Bereich in dem typischerweise interiktale ($R \approx 0.9$)und prä-iktale ($R \in [0.3, 0.6]$) Aufnahmen bewegen. Da die exakten Synchronisationswerte 0.3 und 0.9 nicht häufig genug besetzt waren, wurde für jeden Wert eine Schwankung von maximal

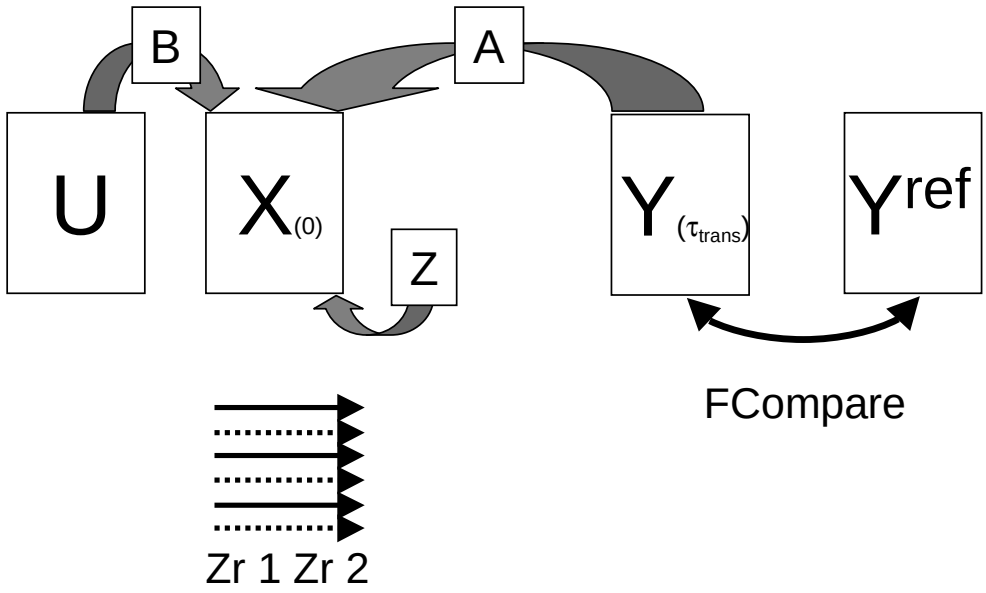


Abbildung 4.5: Topologie 2 mit einem 64×128 CNN

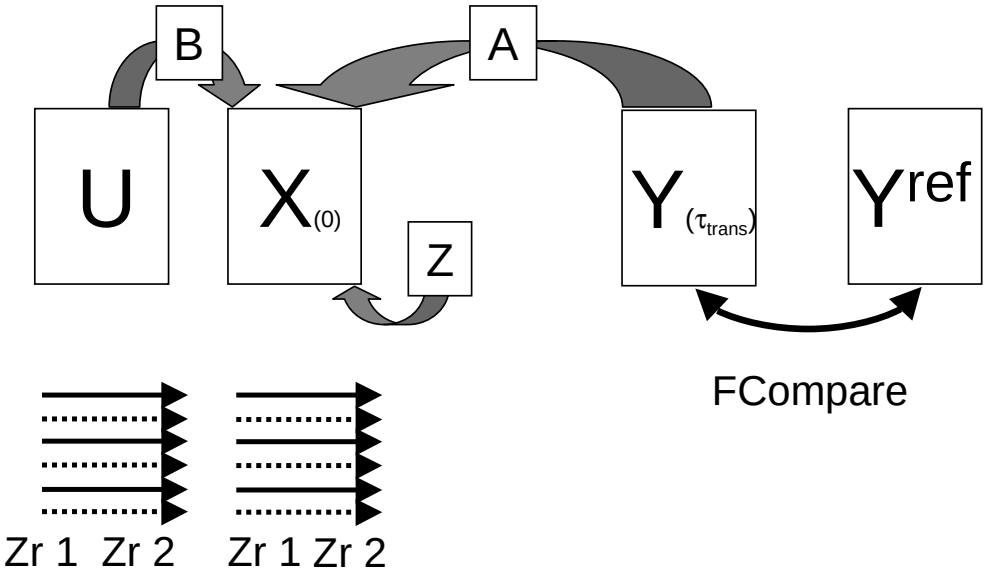


Abbildung 4.6: Topologie 3 mit einem 64×128 CNN

± 0.05 erlaubt. Dementsprechend wurde für jede Datenquelle die Trainingsmenge aus gleich vielen Zeitreihensegmentpaaren aufgebaut, die Synchronisationswerte $R \in [0.25, 0.35]$ und $R \in [0.85, 0.95]$ aufwiesen. Nur für die Trainingsmenge von P2 wurden Zeitreihensegmentpaare mit Synchronisationswerten $R \in [0.45, 0.55]$ und $R \in [0.75, 0.85]$ genutzt, da die vorherigen Wertebereiche nicht ausreichend besetzt waren (vgl. Abbildungen 4.7 a.) bis d.)). Die niedrigen Synchronisationswerte wurden durch eine Referenz mit $y_{ij}^{ref} = -1$ (*weiß*) für

alle Zellen ij und die hohen durch eine Referenz mit $y_{ij}^{ref} = 1$ (*schwarz*) für alle Zellen ij dargestellt.

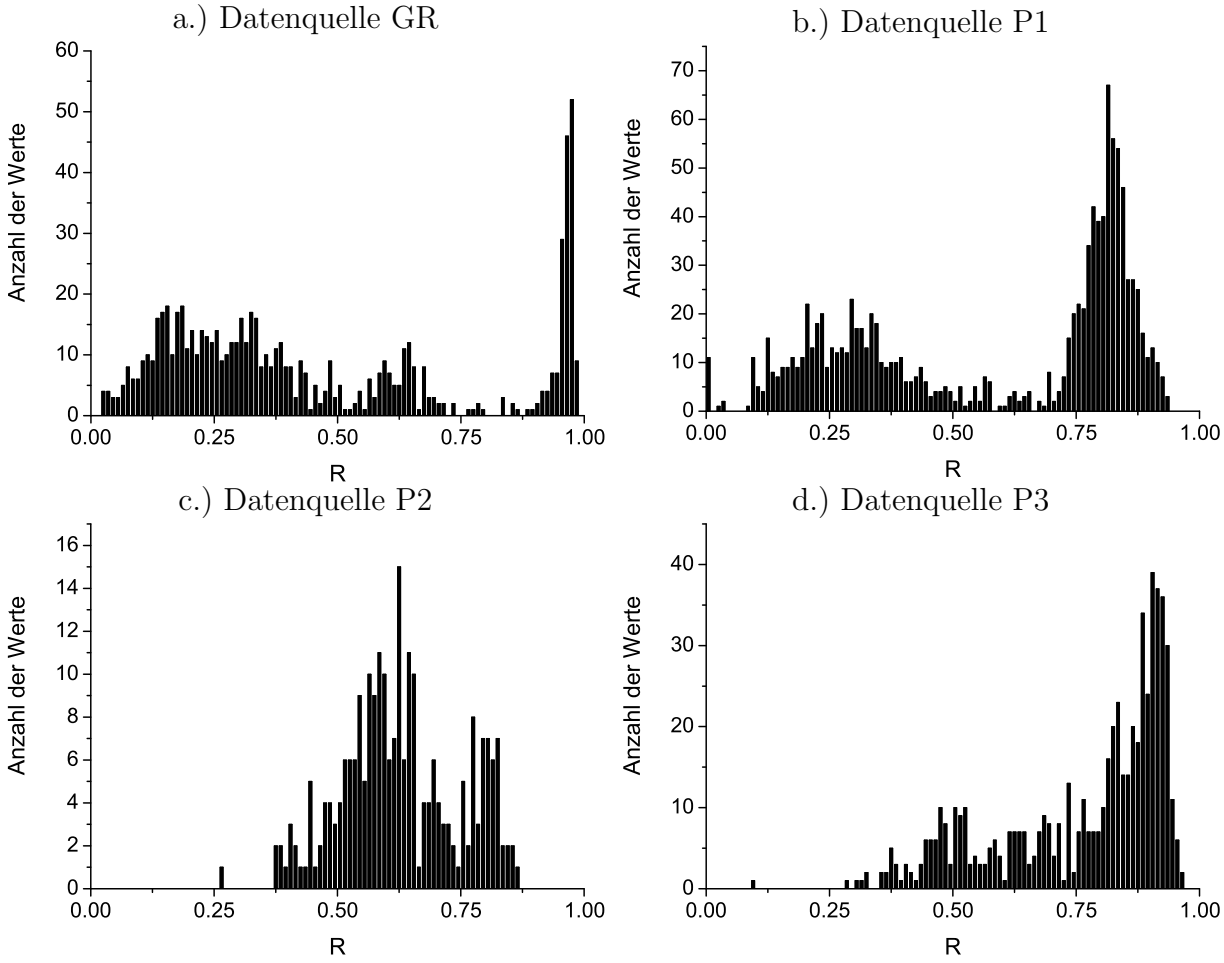


Abbildung 4.7: Verteilung der Synchronisationswerte R der Datenquellen GR, P1, P2 und P3

Die Trainingsmengen der vier Datenquellen wurden aus Elementen aufgebaut, die jeweils aus Datenfensterpaaren und den dazugehörigen Synchronisationswerten bestanden. Ihre Größe wurde von zwei über vier und acht zu 16 Elementen erhöht. Dabei entsprach jeweils eine Hälfte der Elemente Datenfensterpaaren, die niedrigen bzw. hohen Synchronisationswerten zugeordnet werden konnten. Die Trainings- und Testmengen der Datenquellen P1 bis P3 wurden aus jeweils nur einer Elektrodenkanalkombination mit dem möglichst größten Unterschied in den Synchronisationswerten R von interiktaler und hypothetischer prä-iktaler Phase gewonnen. Die jeweils anderen 17 Kanalkombinationen (siehe dazu Kapitel 1.4.2) wurden außer Acht gelassen. Diese Vorgehensweise wurde gewählt, um einen möglichst breiten Wertebereich von R trainieren zu können. Es wurden allerdings nur Elemente aus den interiktalen und prä-iktalen Zustände für das Training verwendet. Die iktalen und post-iktalen Zustände wurden nicht für das Training genutzt, da sie vollkommen ande-

re Dynamiken enthalten, die nicht primärer Gegenstand der Untersuchungen dieser Arbeit sind.

Die Abbildung 4.8 zeigt beispielhaft zwei Elemente einer Trainingsmenge für ein CNN der Topologie 1. Die obere Bilderreihe enthält die vorverarbeiteten Zeitreihensegmente in der Eingabe U und $X(0)$ und die Refernez mit $y_{ij}^{ref} = 1$ für alle Zellen ij (entspricht einem Synchronisationswert $R = 0.9$). Die untere Bilderreihe zeigt beispielhaft den Fall mit einem niedrigen Synchronisationswert $R = 0.3$. Die rechten beiden Bilder enthalten die tatsächlich vom SCNN System berechneten Ausgaben $Y(\tau_{trans})$. Zu erkennen sind Flecken in den Flächen und Bewertungen von $MSEn = 0.091$ bzw. $MSEn = 0.061$. Damit wurden durch das Training keine idealen Parameter gewonnenen und damit keine absolut fehlerfreien Ausgaben bei der Simulation erreichten.

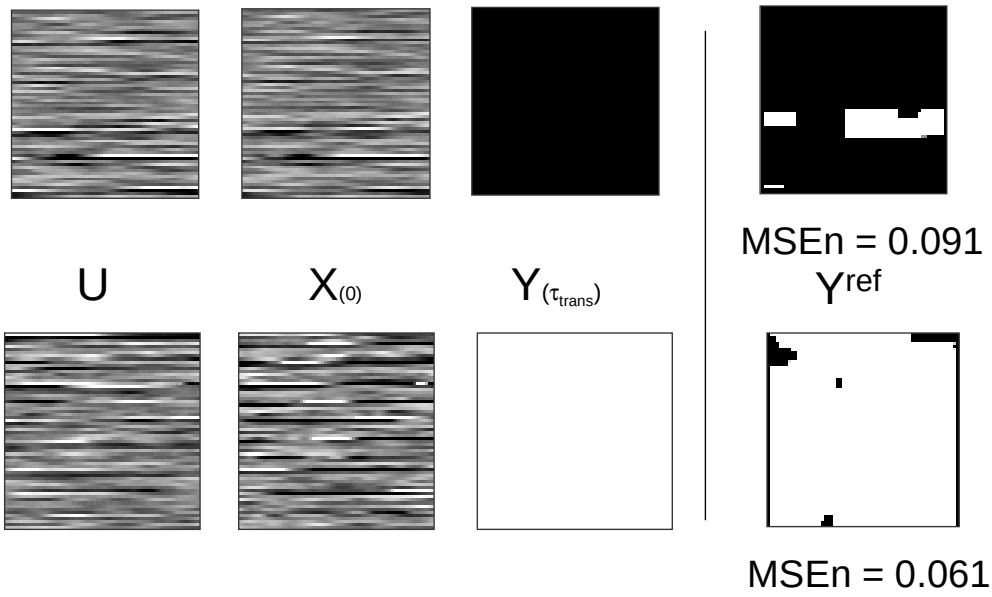


Abbildung 4.8: Zwei beispielhafte Elemente der Trainingsmenge einer Datenquelle (CNN mit der Topologie 1)

Um aus den gewonnene Ausgaben $Y(\tau_{trans})$ Synchronisationswerte berechnen zu können, wurde die folgende zweistufige *Rückrechnung* durchgeführt. In den folgenden Gleichungen 4.1, 4.3, 4.4 und 4.5 stellen N_x und N_y die Anzahl Zellen der CNN-Topologie in x bzw. y Richtung und H die Heaviside Funktion dar. Zunächst wurde mit R_{lin} (siehe Gleichung 4.1) eine lineare Wertebereichtransfomation von $[-1, 1]$ nach $[0, 1]$ durchgeführt.

$$R_{lin} = \frac{1}{N_x N_y} \sum_{i=0}^{N_x} \sum_{j=0}^{N_y} \frac{y_{ij}(\tau_{trans}) + 1}{2} \tag{4.1}$$

Im zweiten Schritt erfolgte die lineare Rücktransformation in den Wertebereich $[0.3, 0.9]$ (bzw. für Datenquelle P2 in den Wertebereich $[0.5, 0.8]$) und damit die Berechnung von R^{approx}:

$$R^{approx} = (R_{lin}(0.9 - 0.3)) + 0.3 \tag{4.2}$$

Mit dieser Vorgehensweise ergab sich beispielsweise für das Bild oben rechts in der Abbildung 4.8 ein approximierte Synchronisationswerte von $R^{approx} = 0.85$. Referenz-Synchronisationswerte die außerhalb des Wertebreichs $[0.3, 0.9]$ (bzw. für Datenquelle P2 von $[0.5, 0.8]$) lagen, wurden nach erfolgreichem Training entsprechend auf die Grenzen $R^{approx} = 0.3$ bzw. $R^{approx} = 0.9$ (bzw. $R^{approx} = 0.5$ bzw. $R^{approx} = 0.8$) gesetzt.

R_{lin} wurde in der vorliegenden Arbeit durchgehend genutzt, kann aber auch durch die nachfolgenden Funktionen ersetzt werden:

$$R_{pos} = \frac{1}{N_x N_y} \sum_{i=0}^{N_x} \sum_{j=0}^{N_y} H(y_{ij}(\tau_{trans})) \tag{4.3}$$

$$R_{mean} = \frac{1}{2N_x N_y} \left(\sum_{i=0}^{N_x} \sum_{j=0}^{N_y} y_{ij}(\tau_{trans}) \right) + 0.5 \tag{4.4}$$

$$R_{median} = \frac{1}{2} Y(\tau_{trans})_{med} + 0.5 \tag{4.5}$$

mit aufsteigend sortierten $y_{ij}(\tau_{trans})$ ($i \in [0, N_x - 1]$,$j \in [0, N_y - 1]$) der Ausgabe $Y(\tau_{trans})$ in $o_0 < o_1 < o_2 ... < o_{N_x N_y - 1}$ und

$$Y(\tau_{trans})_{med} = \begin{cases} o_{(N_x N_y + 1)/2} & : \quad N_x * N_y \quad \text{ungerade} \\ \frac{1}{2}(o_{N_x N_y/2} + o_{(N_x N_y/2)+1}) & : \quad N_x * N_y \quad \text{gerade} \end{cases} \tag{4.6}$$

4.2.5 Evaluation der Optimierung

Der folgende Abschnitt beschreibt die in der vorliegenden Arbeit genutze Evaluatuins-Technik. Ein Training wurde anhand seiner besten erreichten Bewertung Err^{global}, seiner lokalen Bewertungen der Trainingsmenge Err_i^{lokal} (mit $i \in N_m$ Mächtigkeit der Trainingsmenge) und anhand der Position der besten Bewertung im Trainingsverlauf evaluiert. Die tatsächliche Qualität, in Abhängigkeit der in Abschnitt 4.2 genannten Einstellungen, wurde durch seine beste Bewertung ausgedrückt. Abbildung 4.9 stellt beispielhaft Err^{global} in einen Trainingsverlauf dar, wobei nur die besseren Bewertungen eingetragen wurden. Abbildung 4.10 stellt den gleichen Trainingsverlauf mit allen Bewertungen Err^{global} dar. Hier leidet aber die Übersichtlichkeit an der komplexen Darstellung. Die wesentliche Information, der Abstieg im Fehlergebirge, ist schwer zu erkennen. Daher wurde in der vorliegenden Arbeit die Darstellungsform wie in Abbildung 4.9 gewählt. Im vorliegenden Fall wurde eine beste Bewertung von $Err^{global} = 0.0932$ nach 14301 Trainingsschritten erreicht.

Das gewonnene CNN wurde im zweiten Schritt mit der Testmenge evaluiert. Dabei wurde untersucht, ob das CNN mit den gewonnenen Parametern tatsächlich alle vorliegenden Daten erfolgreich approximiert oder nur mit der Trainingsmenge zufriedenstellend funktioniert und damit *übertrainiert* wurde. Abbildung 4.11 zeigt beispielhaft die berechneten Synchronisationswerte R, die approximierten Synchronisationswerte R^{approx} und die Differenzwerte $\delta R_i = |R_i - R_i^{approx}|$ (mit i als Positionsindex) für eine Zeitreihe.

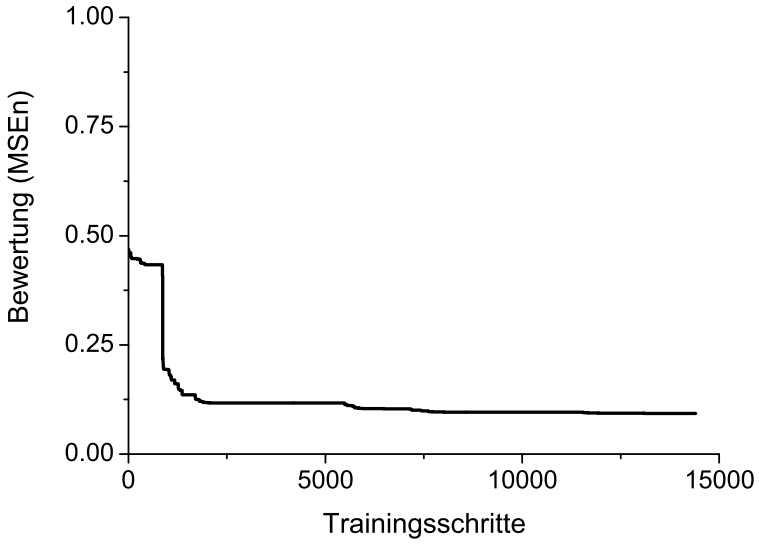


Abbildung 4.9: Beispielhafter Trainingsverlauf, nur bessere Bewertungen wurden eingetragen

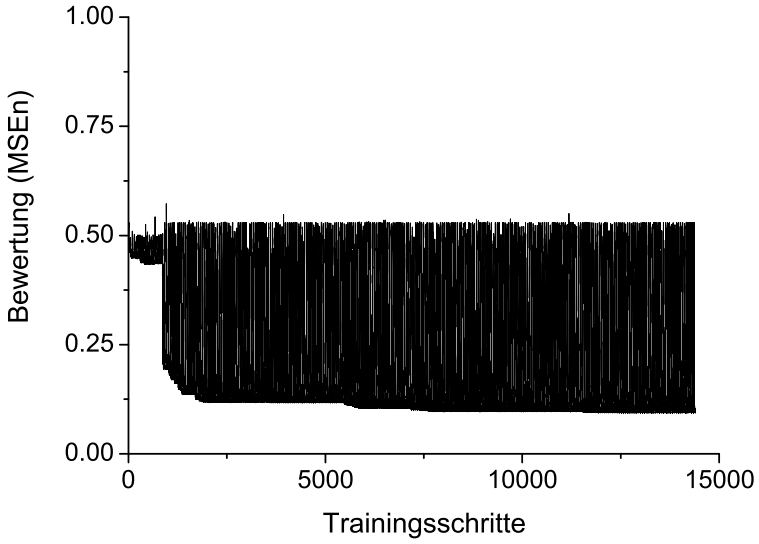


Abbildung 4.10: Beispielhafter Trainingsverlauf, alle Bewertungen wurden eingetragen

Zur weiteren Datenreduktion wurden weitere Größen eingeführt. Für die der Abbildung 4.11 zugrunde liegenden Daten wurde ein mittlerer Synchronisationswert von $\overline{R} = 0.867$ berechnet. Der mittlere approximierte Synchronisationswert wurde mit $\overline{R^{approx}} = 0.683$ bestimmt. Die mittlere Differenz wurden mit $\overline{\delta R} = 0.187$ berechnet.
Mit diesen Informationen wurde ein Maß entwickelt, welches eine Aussage über die *Approximationsqualität* (*Approximationsgüte*) in der gesamten Testmenge ermöglicht. Hier werden die mittleren Differenzen $\overline{\delta R}$ aller verwendeten Zeitreihen berechnet und entsprechend der

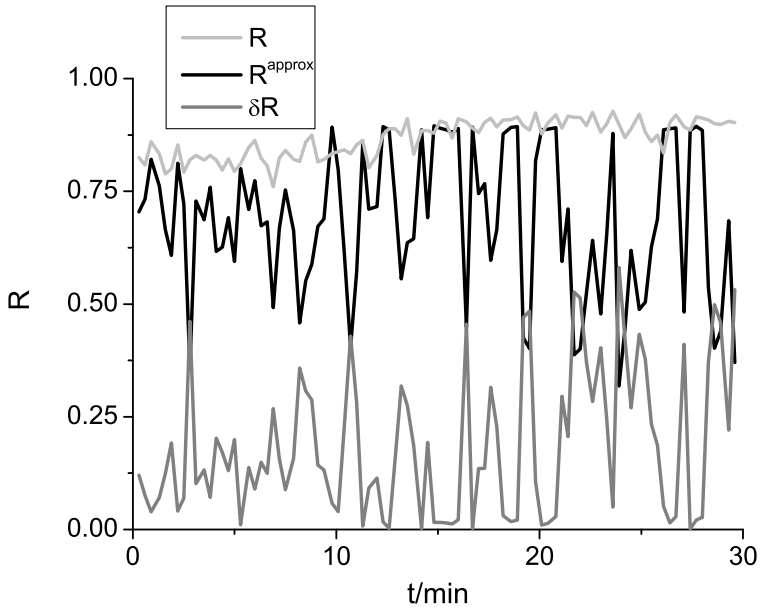


Abbildung 4.11: Beispielhafte Evaluation eines CNN an einem Teil seiner Testmenge: Zeitlicher Verlauf der berechneten Synchronisationswerte (R, hellgrau), der approximierten Synchronisationswerte (R^{approx}, schwarz) und der Differenzwerte (δR, dunkelgrau) für eine Zeitreihe

Dauer gewichtet und gemittelt:

$$Cost = \frac{1}{\sum_{j=0}^{N-1} L_j} \sum_{i=0}^{N-1} \overline{\delta R_i} L_i \tag{4.7}$$

In Gleichung 4.7 entspricht N der Anzahl der Zeitreihen in der Testmenge und L deren jeweilige Dauer.

In den folgenden Untersuchungen wurden bei den Datenquellen P1 bis P3 nur die interiktalen und die hypothetischen prä-iktalen Zeiträume der Zeitreihen als Testmenge genutzt. D.h. $\overline{R}$, $\overline{\delta R}$ und $Cost$ wurden jeweils nur mit Daten aus den interiktalen und den hypothetischen prä-iktalen Zeiträumen berechnet.

Die Tabelle 4.2 enthält eine Auflistung der mittleren Synchronisationswerte $\overline{R}$ aller Zeitreihenpaare der Datenquellen GR und P1 bis P3. Es ist zu berücksichtigen, dass in GRc der Kopplungsfaktor linear erhöht wurde und damit auch $\overline{R}$ entsprechend gestiegen ist. Die zeitliche Entwicklung der anderen Zeitreihen der Datenquellen GRr und P1 bis P3 entsprach beinahe konstanten Verläufen mit entsprechendem $\overline{R}$. Die mittleren Synchronisationswerte der Daten aus den hypothetischen prä-iktalen Phasen P1ef, P1ghij, P2bcd und P3f unterschieden sich deutlich von denen der interiktalen Phasen. Im Fall von P1ef, P1ghij und P2bcd wurden sie auf die Dauer gewichtet zusammengefaßt.

Dieser Unterschied fiel für die Datenquelle P2 geringer aus als für die Datenquellen P1 und P3. Die mittleren Synchronisationswerte der Datensätze P3g und P3f wurde mit sehr

ähnlichen Werten berechnet ($\overline{R} = 0.55$ bzw. $\overline{R} = 0.54$), obwohl die letzteren Daten aus einer interiktalen Phase stammten. Diese Ähnlichkeit läßt sich durch die Tatsache erklären, dass bei der Ableitung des Datensatzes P3f andere Einstellungen im Aufnahmesystem vorherrschten. Für die Aufzeichnung des Datensatzes P3f wurde für die Messapparatur eine andere Spannungs-Referenz gewählt als für die Datensätze P3a bis P3e und P3g. Falls der zeitliche Verlauf der Synchronisation in diesem Datensatz korrekt approximiert wird, so wäre das ein erster Hinweis auf *generelle* Eigenschaften des trainierten CNN.

Mit $Cost^{best}$ wird die maximal erreichbare Approximationsqualität angegeben. Sie errechnet sich nach Gleichung 4.7. Allerdings wurden statt der approximierten die berechneten Synchronisationswerte eingesetzt. Die berechneten Synchronisationswerte die außerhalb der Wertebereiche $[0.3, 0.9]$ (Datenquellen GR, P1 und P3) bzw. $[0.5, 0.8]$ (Datenquelle P2) lagen wurden auf die entsprechenden Grenzen gesetzt. Es ist auffällig, dass die maximale Approximationsqualität $Cost^{best}$ von P3 etwa um eine Größenordnung besser war als die der anderen Datenquellen (vgl. Tabelle 4.2). Dies folgt aus der Tatsache, dass fast alle berechneten Synchronisationswerte innerhalb des Wertebreich $[0.3, 0.9]$ liegen (vgl. Amplitudenverteilung in Abbildung 4.7 d.)).

gek. Rössler (*GR*) Zr - $\overline{R}$	Patient 1 (*P1*) Zr - $\overline{R}$	Patient 2 (*P2*) Zr - $\overline{R}$	Patient 3 (*P3*) Zr - $\overline{R}$
a - 0.19	a - 0.80	a - 0.79	a - 0.82
b - 0.21	b - 0.87	bcd - 0.59	b - 0.83
c - 0.42	c - 0.83		c - 0.86
d - 0.33	d - 0.83		d - 0.89
e - 0.66	ef - 0.37		e - 0.91
f - 0.96	ghij - 0.29		f - 0.55
	k - 0.79		g - 0.54
$Cost^{best} = 0.061$	$Cost^{best} = 0.03$	$Cost^{best} = 0.012$	$Cost^{best} = 0.005$

Tabelle 4.2: Durchschnittliche R-Werte und Approximationsqualitäten $Cost^{best}$ der Datenquellen GR und P1 bis P3

Die zeitlichen Verläufe der Synchronisationswerte der Zeitreihenpaare der Datenquellen werden im Kapitel 4.4 ausführlich behandelt.

4.3 Optimierung der Netzwerkeinstellungen

Der folgende Abschnitt gibt einen Überblick über die Erfahrungen, die zu den jeweils besten CNN-Einstellungen für die Datenquellen GR und P1 bis P3 geführt haben.

Vergrößerung der Trainingsmenge

Die Trainingsmenge wurde, bei gleich bleibenden CNN-Einstellungen, von 1-1 über 2-2 und 4-4 auf 8-8 Elemente erhöht (siehe auch Kapitel 4.2.4). 4-4 z.B. bedeutet, dass für die

Trainingsmenge je vier Datenfensterpaare mit entsprechend niedrigen und hohen Synchronisationswerten zufällig gewählt wurden. Das Parameter-Training mit 1-1 und 2-2 Elementen in der Trainingsmenge verlief erfolglos, da das Training nach kürzester Trainingszeit mit $Err^{global} = 0$ bewertet wurde. Damit wurde das CNN übertrainiert und erreichte nur auf der Trainingsmenge das gewünschte Ergebnis. Erst die Wahl von 4-4 und besonders 8-8 Elementen in der Trainingsmenge, erbrachte eine Steigerung der Approximationsgenauigkeit. Für das folgende Parameter-Training wurde für jede Datenquelle je eine Trainingsmenge mit 8-8 Elementen gewählt.

Wahl der Zusammenführungsfunktion

Als Zusammenführungsfunktion für das Parameter-Training wurde der Mittelwert genutzt. Durch den Mittelwert wurden die Fehlergebirge aller Elemente der Trainingsmenge gleichzeitig betrachtet. Obwohl die Möglichkeit von *verwaschenen* (globalen) Minima der einzelnen Fehlergebirge im gesamten gemittelten Fehlergebirge bestand, erwies sich der Mittelwert als geeignete Zusammenführungsfunktion und wurde im folgenden Parameter-Training genutzt. Weitere Informationen dazu sind im Anhang F.4 zu finden.

Wahl der Bewertungsfunktion

Als Bewertungsfunktion wurde $MSEn$ genutzt. Dieses Maß hatte sich aufgrund des Aufbaus der Referenzen Y^{ref} bewährt. Da die Referenzen nur weiß oder schwarz gewählt wurden, wurde durch den $MSEn$ der mittlere quadratische Abstand zu den *Grauwerten* in Y^{ref} angegeben. Weitere Informationen hierüber sind im Anhang F.3 aufgeführt.

Wahl der CNN-Topologie

Bei gleich bleibenden Einstellungen hatte sich die Topologie Nr1. bewährt. Tabelle 4.3 zeigt beispielhaft für drei verschiedene Template-Einstellungen die Trainingsergebnisse für die Topologie 1 und 2. Die Überlegenheit der CNN-Topologie 1 ist deutlich zu erkennen. Die Bewertung Err^{global} wurde um den Faktor 2 verbessert. Mit dieser Wahl wurde die Übertragbarkeit der Einstellungen auf die schaltungstechische-Realisation ACE4K vereinfacht, da dieses CNN auch einer Topologie von 64×64 Zellen entspricht.

Wahl des Optimierungsverfahrens

Mit dem Iterative-Annealing (siehe auch Anhang B) als Optimierungsverfahren wurden die besten Ergebnisse erzielt. Tabelle 4.4 zeigt beispielhaft das überlegene Abschneiden des Iterative-Annealing Optimierungsverfahrens bei gleichbleibenden CNN-Einstellungen (CNN-Topologie 1, Trainingsmenge 4-4, Datenquelle P1, Template-Topologie 3×3, polynomielle Templates vierter Ordnung (für A und B gleich), Schwellenwert Z invariant). Durch die Nutzung des mehrfachen Abkühlzyklus von einer Starttemperatur T_0 an, mit dem die Schrittreichweite beim Suchen nach dem globalen Minimum variiert wurde, wurden im

	Template-Topologie	Template-Ordnung	Err^{global}	bei Trainingsschritt
CNN	3×3	4	0.1332	9107
Topologie	5×5	1	0.1945	14034
1	5×5	2	0.1324	8758
CNN	3×3	4	0.2293	12845
Topologie	5×5	1	0.4545	3612
2	5×5	2	0.2554	4709

Tabelle 4.3: Beispielhafter Vergleich zwischen Topologie 1 und 2 bei gleich bleibenden CNN-Einstellungen

Vergleich zu Powell's Linienminimierungsverfahren und dem Downhill-Simplex Optimierungsverfahren um den Faktor 2 verbesserte Ergebnisse erzielt.

Optimierungsverfahren	Err^{global}	bei Trainingsschritt
Powell's	0.2183	14231
Simplex	0.1423	4997
Iterative-Annealing ($T_0 = 50$, 300 Abkühlschritte)	0.0969	14364

Tabelle 4.4: Beispielhafter Vergleich zwischen verschiedenen Optimierungsverfahren bei gleichbleibenden CNN-Einstellungen

Beim Powell's Linienminimierungsverfahren und dem Downhill-Simplex Optimierungsverfahren besteht die Gefahr, dass beide Verfahren lokale Minima finden, aber keine weitere Optimierung durchführen. Dies wird besonders am Eintrag zum Downhill-Simplex Optimierungsverfahren in der Tabelle 4.4 deutlich, da hier schon nach 4997 Trainingsschritten ein lokales Minimum erreicht wurde und in den letzten 9403 Schritten keine Verbesserung erzielt werden konnte.

Wahl der Template-Topologie und Ordnung

Bei gleich bleibenden sonstigen CNN-Einstellungen wurde die Template-Topologie 3×3 mit polynomiellen Templates 2. und 3. Ordnung (je gleiche Einstellungen für Template A und B) beim Training am besten bewertet. Auch größere Templates wie z.B. mit 5×5 Topologie, mit polynomiellen 2. Ordnung, wurden mit einem guten Err^{global} bewertet. Allerdings wurde im Hinblick auf die *einfache* technische Realisierbarkeit mit den Templates mit 3×3 Topologie weitergearbeitet.

Weitere Steigerung des Ergebnisqualität der Trainings

Die Qualität der Trainings konnte durch die Einführung anderer Randbedingungen gesteigert. Neben der bis dahin genutzten neutralen Randbedingung wurden die periodische

Randbedingung und geschlossene Spirale in x-Richtung eingeführt. Auch durch die Beschneidung des Parameterraums durch Einsatz von Symmetrien (Punkt, Kreuz, Stern) auf den Templates wurden weitere Erfolge erzielt. Dabei wurden den A und B Templates dieselbe Symmetrie aufgeprägt.

Wiederholung der besten Trainings

Die Trainings mit der besten Bewertung Err^{global} wurden immer sechs fach wiederholt, um zu prüfen, ob tatsächlich das globale Minimum gefunden wurde. Das Training eines CNN mit z.B. 3×3 Template-Topologie und polynomielle Templates 3. Ordnung, invariantem Schwellenwert Z und einer Trainingsmenge 8-8 mit dem Mittelwert als Zusammenführungsfunktion, bedeutet die Minimierung einer Bewertungsfunktion, die aus 16 per Mittelwert zusammengefaßten Bewertungsfunktionen, mit je 55 Paramtern, besteht (siehe auch 3.2.3). Dadurch entsteht ein *gemitteltes* Fehlergebirge aus 55 Parametern, mit vielen beieinander liegenden lokalen Minima. Diese lokalen Minima stammen von einigen Fehlergebirgen der Elemente der Trainingsmenge und senken durch die Mittelung diesen Punkt im *gemittelten* Fehlergebirge ab, obwohl andere Fehlergebirge dort möglicherweise kein Minimum besitzen. Dieses Problem kann durch die Nutzung des *durchschnittlichen Abstandes zur Bewertung* $ADev$ (also *Aufweitung* der einzelnen Err_i^{local}) quantifiziert werden:

$$ADev = \frac{1}{N_m} \sum_{i=0}^{N_m-1} \left| Err^{global} - Err_i^{local} \right| \quad \text{mit} \quad N_m \quad \text{als Anzahl der Trainingselemente} \tag{4.8}$$

Nimmt $ADev$ Werte nahe Null an, so werden alle Elemente der Trainingsmenge gleich gut, nahe am Err^{global}, bewertet. Bei einem großem $ADev$ gibt es einige Elemente in der Trainingsmenge, die kein Minimum in den gewonnenen Parametern haben. Durch den Mittelwert als Zusammenführungsfunktion verbessern die anderen Elemente der Trainingsmenge die Bewertung dieser Parameter und ermöglichen trotzdem einen guten Err^{global} Wert. Tabelle 4.5 zeigt beispielhaft die Ergebnisse für sechs Wiederholungen eines Trainings (Datenquelle P3, Trainingsmenge 8-8; Iteratives Annealing als Optimierungsverfahren; Schwellenwert Z invariant; Template-Topologie 3×3, polynomielle Templates 3. Ordnung und Stern-Symmetrie für A und B; periodische Randbedingungen). Die Aufweitung der Wiederholungen mit den Nummern 3 und 5 wurden ähnlich gut mit $ADev = 0.0400$ und $ADev = 0.0401$ bestimmt, d.h. die 16 Elemente der Trainingsmenge wurden sehr nah an den globalen Systemfehler Err^{global} bewertet. Allerdings wurde das lokale Minimum der Wiederholung 5 doppelt so gut bewertet wie das der Wiederholung 3. In den anderen Wiederholungen wurde in schlecht bewerteten Regionen des Fehlergebirges nach einem Minimum gesucht. Dabei konnte innerhalb der vorgegebenen Trainingszeit ein Optimum gefunden werden.

4.4 Optimierungsergebnisse

Im folgenden Abschnitt werden die Trainingsergebnisse dargestellt. Dabei wird zunächst auf die gekoppelten Rössler-Systeme (Datenquelle GR) eingegangen. Nachdem die CNN-

Wiederholung Nr.	Err^{global}	$ADev$	Trainingsschritt
1	0.4563	0.2595	13969
2	0.1665	0.1137	12139
3	0.0651	0.0400	13669
4	0.3736	0.3399	14065
5	0.0370	0.0401	7391
6	0.2372	0.2511	13235

Tabelle 4.5: Darstellung der Ergebnisse für sechs Wiederholungen eines Trainings

Einstellungen, die zur gefundenen optimalen Approximationsqualität geführt hatten, dargelegt wurden, wird die Testmenge dargestellt und eine Erläuterung der Ergebnis vorgenommen.

Im Anschluß daran wird ebenso auf die Ergebnisse eingegangen, die aus den Patientendaten (P1 bis P3) gewonnen wurden. Hier wurde der Weg, der zu den besten CNN-Einstellungen des Modellsystems führte, auf EEG-Daten übertragen.

Für alle vier Datenquellen wurden die folgenden Einstellungen gewählt: Es wurde eine 8-8 Trainingsmenge, das Iterative-Annealing als Optimierungsverfahren (mit Starttemperatur $T_0 = 50$ und 300 Schritten pro Abkühlperiode), 14400 Trainingsschritte, der Mittelwert als Zusammenführungsfunktion, $MSEn$ als Bewertungsfunktion und die CNN-Topologie Nr.1 genutzt.

4.4.1 Gekoppelte Rössler-Systeme

CNN-Einstellungen

Mit den folgenden CNN-Einstellungen wurde eine optimale Bewertung von $Err^{global} = 0.0707$ nach 14398 Trainingsschritten, mit einer Aufweitung von $ADev = 0.0427$, erreicht: Die A und B Templates hatten eine 3×3 Topologie, Punkt-Symmetrie und waren Polynome 2. Ordnung. Es wurde die Randbedingung geschlossene Spirale in x-Richtung gewählt. Die Approximationsqualität lag bei $Cost = 0.1408$. In der Abbildung 4.12 ist die Bewertung in Abhängigkeit der Trainingsschritte dargestellt.

Darstellung der gewonnenen Parameter

Die Gleichungen 4.9, 4.10 und 4.11 stellen die Templates A und B und den Schwellenwert Z dar, die durch das Parameter-Training als optimale Werte für die Datenquelle GR gewonnen wurden. In jeder Zelle der 3×3 Templates A und B sind jeweils zwei Werte geklammert eingetragen. Der obere Wert ist jeweils der erste und der untere der zweite Eintrag des Template-Polynoms. In diesen drei Gleichungen und in denen der Datenquellen P1 bis P3 werden die Einträge in voller Genauigkeit dargestellt. Diese Darstellungsweise wurde gewählt, da im Abschnitt 4.6 beschrieben wird, dass selbst kleinste Änderungen bei den Templates zu einer Verringerung der Approximationsqualität führen können. Diese

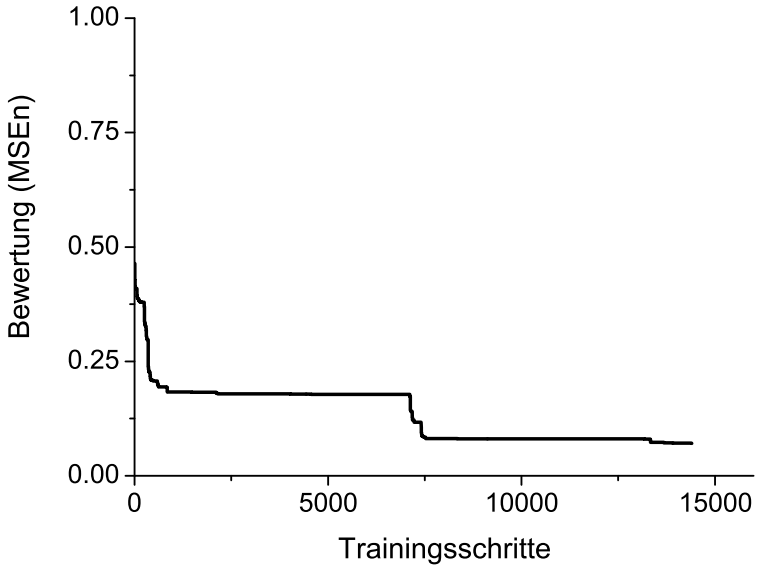


Abbildung 4.12: Trainingsverlauf des gekoppelten Rössler-Systems (Datenquelle GR)

Darstellungsform wird für alle folgenden Templates A und B und den Schwellenwert Z beibehalten.

$$A = \left\{ \begin{array}{ccc} (10.4698959015 & (6.654295705 & (8.14954372076 \\ 12.0021381311) & -8.29297309706) & 7.32597225806) \\ \\ (-5.24234520661 & (-7.46548382715 & (-5.24234520661 \\ 5.97859414135) & -2.71926781621) & 5.97859414135) \\ \\ (8.14954372076 & (6.654295705 & (10.4698959015 \\ 7.32597225806) & -8.29297309706) & 12.0021381311) \end{array} \right\} \tag{4.9}$$

$$B = \left\{ \begin{array}{ccc} (13.6562385835 & (-18.6812534226 & (-11.3704705392 \\ -12.3637035121) & 25.5786820125) & -1.40126867031) \\ \\ (15.7167272639 & (-2.00651076626 & (15.7167272639 \\ -10.492100532) & 9.23035115413) & -10.492100532) \\ \\ (-11.3704705392 & (-18.6812534226 & (13.6562385835 \\ -1.40126867031) & 25.5786820125) & -12.3637035121) \end{array} \right\} \tag{4.10}$$

$$Z = \left\{ \begin{array}{c} -27.5994423522 \end{array} \right\} \tag{4.11}$$

Darstellung der Testmenge

In den Datensätze GRa, GRb, GRd, GRe und GRf wurde für Datenpunkte eine konstante Kopplungsstärke ($\epsilon = 0$, $\epsilon = 0.01$, $\epsilon = 0.02$, $\epsilon = 0.03$ bzw. $\epsilon = 0.04$) verwendet (vgl. Kapitel

4.1), so dass die Ergebnisse der Approximation in der Abbildung 4.13 zusammengefaßt wurden. Der Graph in der Abbildung 4.14 stellt jeweils die berechneten Synchronisationswerte R (Referenzwerte), die durch das CNN approximierten Synchronisationswerte R^{approx} und die Differenzwerte δR für den Datensatz GRc dar.

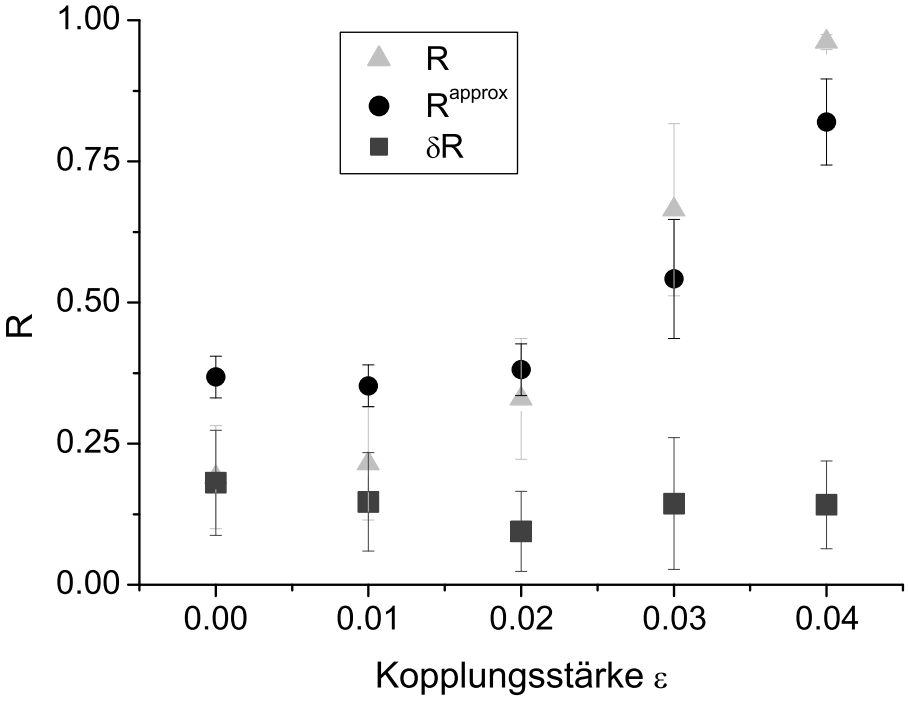


Abbildung 4.13: Berechnete mittlere Synchronisationswerte ($\overline{R}$, Dreieck), approximierte mittlere Synchronisationswerte ($\overline{R^{approx}}$, Kreis) und mittlere Differenzwerte ($\overline{\delta R}$, Rechteck) der gekoppelten Rössler-Systeme (Datenquelle GR) mit den jeweiligen Standardabweichungen unter der Wahl von verschiedenen Kopplungsstärken

Diskussion

Für die Datensätze GRa und GRb wurden aufgrund der schwachen Kopplung Synchronisationswerte nahe $R^{approx} = 0.3$ (vgl. Graph in Abbildung 4.13) approximiert. Dies entsprach der unteren Grenze der gewählten Auflösung von $[0.3, 0.9]$, so dass Kopplungen mit kleineren Synchronisationswerten als 0.3 mit $R^{approx} = 0.3$ approximiert wurden.

Der Datensatz GRd wurde, im Vergleich zu den anderen Datensätzen der Datenquelle GR, mit einer mittleren Differenz der Synchronisationwerte von $\overline{\delta R} = 0.0947$ abgeschätzt und damit am besten bewertet (vgl. Tabelle 4.6). In den Datensätzen GRe und GRf nahm die Approximationsqualität wieder ab. In beiden Datensätzen wurde die Kopplungsstärke von $\epsilon = 0.02$ (Datensatz GRd) auf $\epsilon = 0.03$ bzw. $\epsilon = 0.04$ (Datensatz GRe bzw. GRf) erhöht. Diese Erhöhung wird in der Abbildung 4.14 noch einmal verdeutlicht. In diesem Datensatz GRc wurde die Kopplung linear von $\epsilon = 0$ (für den ersten Datenpunkt) bis $\epsilon = 0.04$ (für den letzten Datenpunkt) gesteigert. Für eine Kopplungsstärke im Bereich

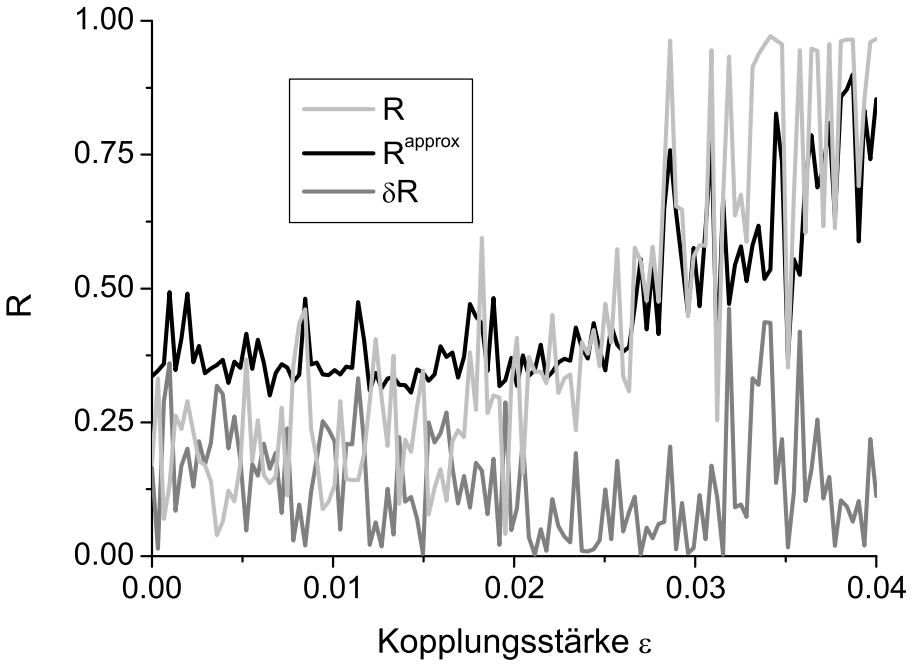


Abbildung 4.14: Zeitlicher Verlauf der Synchronisationswerte (R, hellgrau), der approximierten Synchronisationswerte (R^{approx}, schwarz) und der Differenzwerte (δR, dunkelgrau) des Datensatzes GRc der gekoppelten Rössler-Systeme (Datenquelle GR)

$\epsilon \in [0.02, 0.03]$ (entspricht Synchronisationswerten aus dem Bereich $R \in [0.3, 0.7]$) wurde eine deutliche Verringerung der Differenz der Synchronisationswerte erkennbar. Daten, die aus Bereichen mit kleinerer oder größerer Kopplungsstärke stammten, wurden, wie oben erläutert, mit größeren Differenzen zu den tatsächlichen Werten approximiert.

Für die Trainingsmenge wurden Datenfensterpaare mit Synchronisationswerten aus den Bereichen $R \in [0.25, 0.35]$ und $R \in [0.85, 0.95]$ verwendet. Mit dem gewonnenen CNN wurde allerdings ein gutes Approximationsverhalten im Bereich $R \in [0.3, 0.7]$ erreicht. Zu höheren Synchronisationswerten nahm die Qualität des Approximationsverhaltens ab. Trotzdem wurde durch die Messung von Synchronisation per CNN ein deutlicher Unterschied von starker und schwacher Synchronisation festgestellt, so dass das hier genutzte Verfahren zur Messung von Synchronisation in EEG-Zeitreihen als vielversprechender Ansatz angesehen werden kann.

4.4.2 EEG-Daten

Nach den ermutigenden Ergebnis der Synchronisationsmessung an gekoppelten Rössler-Systemen mit CNN, wurde die entwickelte Methodik zur Untersuchung von EEG-Daten angewendet. Im Gegensatz zu den synthetisch generierten Zeitreihen ist die den EEG-Zeitreihen zugrunde liegende Dynamik nicht direkt beschreibbar. Durch die EEG-Aufzeichnungen wird nicht nur der pathologische epileptogene Prozess, sondern auch die physiologischen Dynamiken die am Aufzeichungsort im Gehirn vorherrschen aufgezeichnet. Das

Synchronisationsmaß R ist geeignet um Teilaspekte dieser *Misch-Dynamik* zu charakterisieren (vgl. Kapitel 1.4.3).

EEG Daten Patient 1 (Datenquelle P1)

CNN-Einstellungen Mit den folgenden CNN-Einstellungen wurde eine optimale Bewertung von $Err^{global} = 0.0403$ nach 14160 Trainingsschritten, mit einer Aufweitung von $ADev = 0.0549$, erreicht: Die A und B Templates hatten eine 3×3 Topologie und waren Polynome 3. Ordnung. Es wurde die neutrale Randbedingung gewählt. Die Approximationsqualität lag bei $Cost = 0.1055$. In der Abbildung 4.15 ist die Bewertung in Abhängigkeit der Trainingsschritte dargestellt.

Darstellung der gewonnenen Parameter Die Gleichungen 4.12, 4.13 und 4.14 stellen die Templates A und B und den Schwellenwert Z dar, die durch das Parameter-Training als optimale Werte für die Datenquelle P1 gewonnen wurden. In jeder Zelle der 3×3 Templates A und B sind jeweils drei Werte geklammert eingetragen. Der obere Wert ist jeweils der erste, der mittlere der zweite und der untere der dritte Eintrag des Template-Polynoms.

Darstellung der Testmenge Die Abbildungen 4.19 a.) bis f.) und 4.20 g.) bis k.) am Ende dieses Kapitels stellen jeweils die zeitliche Entwicklung der berechneten Synchronisationswerte R (Referenzwerte), der durch das CNN approximiertem Synchronisationswerte R^{approx} und der Differenzwerte δR für die Datensätze P1a bis P1f dar.

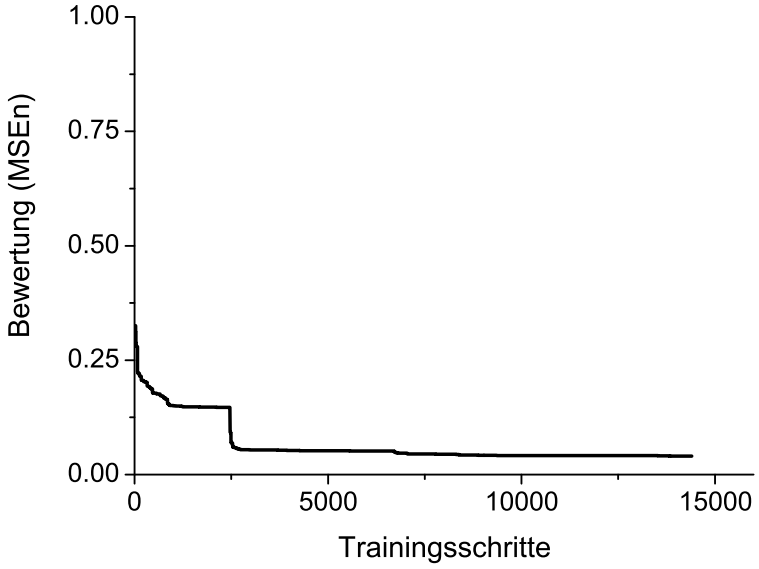


Abbildung 4.15: Trainingsverlauf für die EEG-Datenquelle P1

$$A = \left\{ \begin{array}{ccc} (0.420523508611 & (0.178229030036 & (0.420523508611 \\ -0.364863449636 & -0.465144407505 & -0.364863449636 \\ 0.496380603725) & -0.00141091633297) & 0.496380603725) \\ \\ (0.178229030036 & (-0.859654091692 & (0.178229030036 \\ -0.465144407505 & 0.975130435745 & -0.465144407505 \\ -0.00141091633297) & -0.266789593765) & -0.00141091633297) \\ \\ (0.420523508611 & (0.178229030036 & (0.420523508611 \\ -0.364863449636 & -0.465144407505 & -0.364863449636 \\ 0.496380603725) & -0.00141091633297) & 0.496380603725) \end{array} \right\} \quad (4.12)$$

$$B = \left\{ \begin{array}{ccc} (-0.306931976806 & (0.0270188871222 & (-0.306931976806 \\ 0.453807311111 & -0.0903871539463 & 0.453807311111 \\ 0.200700644172) & -0.0339893457456) & 0.200700644172) \\ \\ (0.0270188871222 & (-1.0806653145 & (0.0270188871222 \\ -0.0903871539463 & -0.583972759404 & -0.0903871539463 \\ -0.0339893457456) & 0.726987405908) & -0.0339893457456) \\ \\ (-0.306931976806 & (0.0270188871222 & (-0.306931976806 \\ 0.453807311111 & -0.0903871539463 & 0.453807311111 \\ 0.200700644172) & -0.0339893457456) & 0.200700644172) \end{array} \right\} \quad (4.13)$$

$$Z = \left\{ \begin{array}{c} 0.966623230735 \end{array} \right\} \quad (4.14)$$

Diskussion Die interiktalen Datensätze P1a, P1b, P1c und P1k (siehe Abbildungen 4.19 a.) bis c.) und 4.20 k.)) zeichnen sich durch eine starke Streuung der approximierten Synchronisationswerte aus, wohingegen der interiktale Datensatz P1d (siehe Abbildung 4.19 d.)) eine gute Approximationsqualität aufweist (vgl. Tabelle 4.6).

Die hypothetischen prä-iktalen Datensätze P1e, P1g, P1h und P1i (siehe Abbildungen 4.19 e.) und 4.20 g.) bis i.)) wurden mit einer gute Approximationsqualität bewertet, wobei bei den Datensätzen P1e und P1i die Synchronisationswerte, die kleiner als 0.3 sind, korrekt mit 0.3 approximiert wurden. Die Datensätze P1f und P1j (siehe Abbildungen 4.19 f.) und 4.20 j.)) enthielten je einen Anfall. Obwohl die Anfallszustände und die post-iktalen Zustände nicht in der Trainingsmenge enthalten waren und nicht in der Bewertung der Approximationsqualität berücksichtigt wurden, wurden die dort vorherrschenden Dynamiken mit approximierten Synchronisationswerten abgeschätzt, die die berechneten nur leicht unterschätzen. Das CNN approximierte also nicht nur die Dynamiken mit einer hohen Qualität, die mit Synchronisationswerte um 0.3 und 0.9 bewertet wurden (durch die Trainingsmenge präsentiert), sondern auch Dynamiken, die mit Synchronisationswerten zwischen 0.3 und 0.9 bewertet wurden und Dynamiken aus Zuständen, die nicht in der Trainingsmege vorhanden waren. Damit zeigte das CNN wiederum *generelle* Eigenschaften.

Die Trennung zwischen niedrigen (0.3) und hohen (0.9) Synchronisationswerten, sowie

die Approximation der Werte dazwischen, die nicht in der Trainingsmenge vorhanden waren, war gelungen. Das gewonnene CNN eignet sich somit zur Definition eines hypothetischen prä-iktalen Zustandes in diesen Datensätzen, auch wenn bei den interiktalen Datensätzen einige Schwächen in der Approximation zu erkennen waren. Weiterhin ist anzumerken, dass die Synchronisationswerte vor einem zweiten Anfall korrekt approximiert wurden, obwohl nur acht niedrige Synchronisationswerte aus dem Bereich vor dem ersten Anfall in der Trainingsmenge vorhanden waren. Das Verhalten vor Anfällen ist reproduzierbar. Damit zeigte das gewonnene CNN weitere *generelle* Eigenschaften. Schließlich soll durch die Synchronisationsmessung nicht nur der der in der Trainingsmenge berücksichtigte prä-iktale Zustand, sondern möglichst auch alle Folgenden definiert werden können.

EEG Daten Patient 2 (Datenquelle P2)

CNN-Einstellungen Mit den folgenden CNN-Einstellungen wurde eine Bewertung von $Err^{global} = 0.2334$ nach 14303 Trainingsschritten, mit einer Aufweitung von $ADev = 0.0585$, erreicht: Die A und B Templates hatten eine 3×3 Topologie, Punkt-Symmetrie und waren Polynome 2. Ordnung. Es wurde die periodische Randbedingung gewählt. Die Approximationsqualität lag bei $Cost = 0.0979$. In der Abbildung 4.16 ist die Bewertung in Abhängigkeit der Trainingsschritte dargestellt.

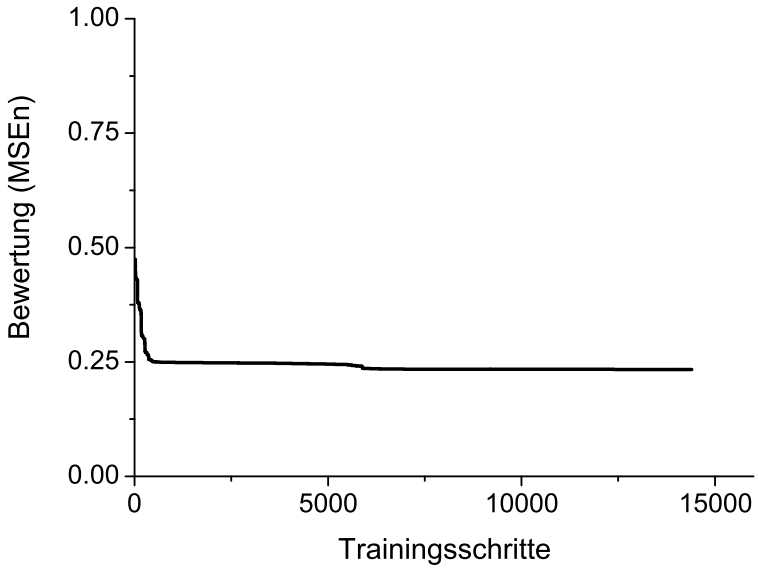


Abbildung 4.16: Trainingsverlauf für die EEG-Datenquelle P2

Darstellung der gewonnenen Parameter Die Gleichungen 4.15, 4.16 und 4.17 stellen die Templates A und B und den Schwellenwert Z dar, die durch das Parameter-Training als optimale Werte für die Datenquelle P2 gewonnen wurden. In jeder Zelle der 3×3 Templates A und B sind jeweils zwei Werte geklammert eingetragen. Der obere Wert ist jeweils der

erste und der untere der zweite Eintrag des Template-Polynoms.

$$A = \left\{ \begin{array}{ccc} (0.330332480782 & (0.309522685885 & (0.474812000401 \\ 0.180651329148) & -0.745124864095) & 0.187619234618) \\ \\ (-0.0992623470826 & (-1.51050133797 & (-0.0992623470826 \\ 0.380649198548) & -0.0299955175021) & 0.380649198548) \\ \\ (0.474812000401 & (0.309522685885 & (0.330332480782 \\ 0.187619234618) & -0.745124864095) & 0.180651329148) \end{array} \right\} \quad (4.15)$$

$$B = \left\{ \begin{array}{ccc} (0.0792535448373 & (-0.131152134097 & (0.12338040961 \\ 0.0850349512259) & -0.244021551987) & 0.249091518377) \\ \\ (0.114131575858 & (-0.355665307783 & (0.114131575858 \\ 0.180181997366) & -0.507451198325) & 0.180181997366) \\ \\ (0.12338040961 & (-0.131152134097 & (0.0792535448373 \\ 0.249091518377) & -0.244021551987) & 0.0850349512259) \end{array} \right\} \quad (4.16)$$

$$Z = \left\{ \begin{array}{c} -0.0109903171914 \end{array} \right\} \quad (4.17)$$

Darstellung der Testmenge In den Abbildungen 4.21 a.) bis d.) am Ende dieses Kapitels stellen jeweils die zeitliche Entwicklung der berechneten Synchronisationswerte R (Referenzwerte), der durch das CNN approximierten Synchronisationswerte R^{approx} und der Differenzwerte δR der Datensätze P2a bis P2d dar.

Diskussion Die vier Datensätzen P2a bis P2d (siehe Abbildungen 4.21 a.) bis d.)) wurden mit glatten Verläufen der Synchronisationswerte approximiert. Allerdings täuschte der niedrige Wert $Cost = 0.0979$ eine hohe Approximationsqualität der Synchronisationswerte vor. Tatsächlich wurde in diesem Fall mit dem gewonnenen CNN keine optimale Trennung zwischen niedrigen (0.5) und hohen (0.8) Synchronisationswerten erreicht. Damit entfiel auch die Möglichkeit der Definition eines hypothetischen prä-iktalen Zustandes in den Datensätzen der Datenquelle P2.

EEG Daten Patient 3 (Datenquelle P3)

CNN-Einstellungen Mit den folgenden CNN-Einstellungen wurde eine Bewertung von $Err^{global} = 0.03697$ nach 7391 Trainingsschritten, mit einer Aufweitung von $ADev = 0.0401$, erreicht: Die A und B Templates hatten eine 3×3 Topologie, Stern-Symmetrie und waren Polynome 3. Ordnung. Es wurde die periodische Randbedingung gewählt. Die Approximationsqualität lag bei $Cost = 0.0934$. In der Abbildung 4.17 ist die Bewertung in Abhängigkeit der Trainingsschritte dargestellt.

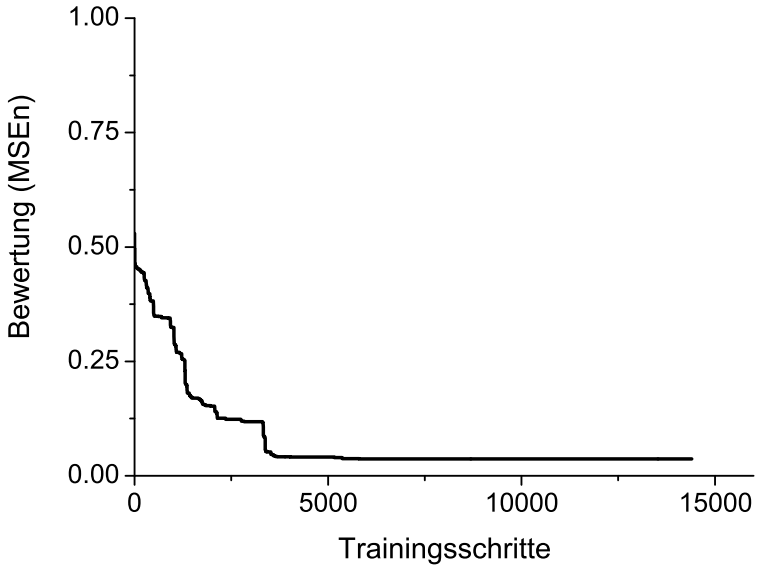


Abbildung 4.17: Trainingsverlauf für die EEG-Datenquelle P3

$$A = \left\{ \begin{array}{ccc} (38.2827584399 & (-6.58872709912 & (38.2827584399 \\ -8.71382200309 & -16.7821166964 & -8.71382200309 \\ -7.98236385982) & -10.6318515339) & -7.98236385982) \\ \\ (-12.1387744763 & (-22.4194134045 & (-12.1387744763 \\ -38.0006775272 & 49.4214666776 & -38.0006775272 \\ 32.2443339806) & 2.33785674317) & 32.2443339806) \\ \\ (38.2827584399 & (-6.58872709912 & (38.2827584399 \\ -8.71382200309 & -16.7821166964 & -8.71382200309 \\ -7.98236385982) & -10.6318515339) & -7.98236385982) \end{array} \right\} \tag{4.18}$$

$$B = \left\{ \begin{array}{ccc} (-34.4788733538 & (28.3001218932 & (-34.4788733538 \\ 3.41574396852 & 38.643774619 & 3.41574396852 \\ 23.5955708089) & -12.4034872169) & 23.5955708089) \\ \\ (3.76606097294 & (-30.3317517296 & (3.76606097294 \\ -2.61880056398 & -17.422292787 & -2.61880056398 \\ 24.960389491) & -25.4875734418) & 24.960389491) \\ \\ (-34.4788733538 & (28.3001218932 & (-34.4788733538 \\ 3.41574396852 & 38.643774619 & 3.41574396852 \\ 23.5955708089) & -12.4034872169) & 23.5955708089) \end{array} \right\} \tag{4.19}$$

$$Z = \left\{ \begin{array}{c} 57.1119508436 \end{array} \right\} \tag{4.20}$$

Darstellung der gewonnenen Parameter Die Gleichungen 4.18, 4.19 und 4.20 stellen die Templates A und B und den Schwellenwert Z dar, die durch das Parameter-Training als optimale Werte für die Datenquelle P3 gewonnen wurden. In jeder Zelle der 3×3 Templates A und B sind jeweils drei Werte geklammert eingetragen. Der obere Wert ist jeweils der erste, der mittlere der zweite und der untere der dritte Eintrag des Template-Polynoms.

Darstellung der Testmenge Die Graphen in den Abbildungen 4.22 a.) bis f.) und 4.23 g.) am Ende dieses Kapitels stellen jeweils die zeitliche Entwicklung der berechneten Synchronisationswerte R (Referenzwerte), der durch das CNN approximierten Synchronisationswerte R^{approx} und der Differenzwerte δR für die Datensätze P3a bis P3g dar.

Diskussion In den interiktalen Datensätzen P3a bis P3e (siehe Abbildungen 4.22 a.) bis e.)) wurden, bis auf einige wenige Ausnahmen in P3d und P3e, die Synchronisationswerte erfolgreich approximiert (vgl. Tabelle 4.6).

Im peri-iktalen Datensatz P3f (siehe Abbildung 4.22 f.)) wurde die hypothetische präiktale Phase erfolgreich approximiert, wobei einige Werte deutlich unterbewertet wurden. Genau wie bei der Datenquelle P1 wurden auch hier die Synchronisationswerte, mit denen die Dynamiken aus den Anfallszustand und dem postiktalen Zustand beschrieben wurden, mit einer hohen Qualiät approximiert. Dabei wurde sogar eine höhere Qualität als bei der Datenquelle P1 erreicht.

Der interiktale Datensatz P3g wurde mit einer anderen Spannungs-Referenz als die anderen Datensätze P3a bis P3f aufgezeichnet (siehe Kapitel 4.2.5). Dadurch ergaben sich bei der Berechnung der Synchronisationswerte niedrigere Werte als bei den anderen interiktalen Datensätzen. Trotz einer häufigen Unterbewertung der berechneten durch die approximierten Synchronisationswerte, kann auch hier von Erfolg gesprochen werden. Obwohl keines der Datenfensterpaare innerhalb der Trainingsmenge genutzt wurde, wurde eine erfolgreiche Approximation durchgeführt. Dies spricht für eine hohe Robustheit der Synchronisationsmessung mit CNN und ist ein weiteres Indiz für *generelle* Eigenschaften des CNN.

Auch hier ist die Trennung zwischen niedrigen (0.3) und hohen (0.9) Synchronisationswerten und die Approximation der Werte dazwischen, die nicht in der Trainingsmenge vorhanden waren, gelungen. Somit eignet sich das gewonnene CNN in diesen Datensätzen zur Definition eines hypothetischen prä-iktalen Zustands, auch wenn bei den prä-iktalen Datensätzen, bzw. Datensätzen mit niedrigen Synchronisationswerten, einige Schwächen in der Approximation zu erkennen waren.

In der Tabelle 4.6 sind die mittleren Differenzwerte der Synchronisationwerte aller verwendeten Datensätze eingetragen. Die daraus gewonnenen gewichteten Mittelwerte ergaben die Approximationsqualität (siehe auch Gleichung 4.7), die zur Bewertung der oben genannten Testmengen der vier Datenquellen genutzt wurde.

Datensatz	$\overline{\delta R}$ GR	$\overline{\delta R}$ P1	$\overline{\delta R}$ P2	$\overline{\delta R}$ P3
a	0.1806	0.1434	0.0950	0.0456
b	0.1468	0.1029	0.1088	0.0856
c	0.1371	0.1070	0.0988	0.0734
d	0.0947	0.0696	0.0760	0.0619
e	0.1438	0.1346		0.0474
f	0.1416	0.0948		0.1662
g		0.0744		0.1735
h		0.1482		
i		0.1078		
j		0.0791		
k		0.1424		

Tabelle 4.6: Mittleren Differen der berechneten und approximierten Synchronisationwerte für alle Datensätze der Datenquellen GR und P1 bis P3

Generalisierungsverhalten mit anderen Datenquellen

Um das *Generalisierungsverhalten* der vier gewonnenen CNN (CNN GR und CNN P1 bis CNN P3) zu untersuchen, wurde jeder CNN auf die anderen drei Testmengen angewendet, die zeitlichen Verläufe der Synchronisationswerte approximiert und bewertet. In der Tabelle 4.7 ist die Approximationsqualität der Testmengen der einzelnen Datenquellen GR und P1 bis P3 unter Nutzung der vier gewonnenen CNN dargestellt. Die Diagonale enthält die oben aufgeführten Werte, die durch die CNN und ihre eigenen Testmengen errechnet wurden.

	Cost GR	*Cost* P1	*Cost* P2	*Cost* P3
CNN GR	0.1468	0.4458	0.1691	0.2571
CNN P1	0.2608	0.1055	0.1596	0.2020
CNN P2	0.3114	0.2924	0.0979	0.1734
CNN P3	0.2601	0.1999	0.1849	0.0934

Tabelle 4.7: Bewertung der Approximiation der Testmengen der Datenquellen GR und P1 bis P3 mit den vier gewonnenen CNN

Diese Untersuchung zeigt, dass durch die gewonnenen CNN nur die Testmengen der Datenquellen, aus denen ihre Trainingsmenge gewonnen wurde, erfolgreich approximieren wurden (siehe Diagonale in der Tabelle 4.7). Die Approximationen der Testmengen der anderen Datenquellen wurden äußerst schlecht bewertet. In keinem der Fälle wurde erfolgreich zwischen hohen und niedrigen Synchronisationswerten unterschieden. Dementsprechend konnte auch in den Datenquellen P1 und P3 kein hypothetischer prä-iktualer Zustand definiert werden, wenn das CNN nicht mit der entsprechenden Trainingsmenge trainiert wurde.

Aufgrund der Nutzung der CNN-Topologie 1 beim Training, wurde die oben genannte Kreuzvalidierung zur Untersuchung des *Generalisierungsverhaltens mit anderen Datenquellen* erneut, unter Vertauschung der Zeitreihensegmente, durchgeführt. Dabei wurde in diesem Fall das Zeitreihensegment 1 in den Status $X(0)$ und das Zeitreihensegement 2 in die Eingabe U geladen (siehe auch Abbildung 4.4). Tabelle 4.8 enthält die so gewonnene Approxminationsqualität der einzelnen Testmengen der Datenquellen GR und P1 bis P3 durch die Nutzung der vier gewonnenen CNN.

	Cost GR	*Cost* P1	*Cost* P2	*Cost* P3
CNN GR	0.2528	0.4368	0.1559	0.2604
CNN P1	0.2613	0.1941	0.3241	0.2098
CNN P2	0.3112	0.3054	0.0911	0.1706
CNN P3	0.2556	0.2838	0.1502	0.1636

Tabelle 4.8: Bewertung der Approximiation der Testmengen der Datenquellen GR und P1 bis P3 mit den vier gewonnenen CNN mit vertauschter Eingabe U und Status $X(0)$

Diese Untersuchung zeigt, dass die zeitlichen Verläufe der approximierten Synchronisationswerte der Testmengen für alle 16 Fälle schlecht approximiert wurden. Dieses Ergebnis war zu erwarten, da mit der Vertauschung der Inhalte der Eingabe U und des Status $X(0)$ ein erheblicher Eingriff in die Konfiguration vorgenommen wurde. Die Parametersuche im Training wurde mit einer völlig anderen Konfiguration durchgeführt und damit in einem anderen Fehlerraum. Es konnte nicht davon ausgegangen werden, dass das durch das Training gefundene Minimum auch das gewünschte globale Minimum in dem der aktuellen (veränderten) Konfiguration zugrunde liegenden Fehlerraum ist.

Gesamtbetrachtung

Trozt des oben genannten Problems, das EEG-Daten, im Gegensatz zu synthetisch erzeugten, eine Messung einer komplexen *Misch-Dynamik* darstellen, und weiterer Unterschiede, wie z.B. die Aufzeichnung mit einer endlichen Abtastfrequenz und endlicher Auflösung, konnte die an den Modellsystemen entwickelte Methodik zur Synchronisationsmessung mit CNN erfolgreich übertragen werden. Die Bewertungen der Trainings das CNN P1 und CNN P3 und deren Approximationsqualität (siehe Diagonale in der Tabelle 4.7) hatten die Werte der gekoppelten Rössler-Systeme übertroffen.

Es war interessant zu sehen, dass das gewonnene CNN für die Datenquelle P1 Schwächen in der Synchronisationsmessung in den interiktalen Datensätzen hatte, während das gewonnene CNN der Datenquelle P3 eher Schwächen in hypothetischen prä-iktalen Datensätzen und in Daten mit niedriger Synchronisation aufwies. Dieses Verhalten könnte durch die Wahl anderer Simulations- und Trainingseinstelungen, wie z.B. einer anderen Bewertungsfunktion, einer anderen Template-Topologie, etc., optimiert werden.

Die Trainingsverläufe der vier Datenquellen gaben unterschiedliche Verhaltensweisen wieder. Im Trainingsverlauf der Datenquelle GR fand am Anfang ein steiler Abstieg und

bei etwa 7400 Trainingsschritten ein weiterer statt (siehe Abbildung 4.12). In den Trainingsverläufen der Datenquellen P1 und P3 fanden sich die steilsten Abstiege bei etwa 2500 Trainingsschritten (siehe Abbildungen 4.15 und 4.17), während im Trainingsverlauf der Datenquelle P2 nach dem Anfang kein weiterer deutlicher Abstieg zu erkennen war (siehe Abbildung 4.12). Dieses Ergebnis ermutigte einerseits, die Trainingszeit zu verringern. Schließlich geschah lange Zeit keine Veränderung mehr. Allerdings deutete sich z.B. im Trainingsverlauf des CNN für die Datenquelle GR ein weiterer Abstieg bei etwa 13500 Schritten an. Solch ein Verhalten war zu erwarten, da nicht davon ausgegangen werden konnte in den vier Fällen Templates und Schwellenwerte (CNN-Parameter) zu finden, die dem globalen Minimum in den jeweiligen Fehlergebirgen (durch die Trainingsmengen der Datenquellen LP, P1 bis P3 und deren CNN-Konfiguration bestimmt) entsprachen. Somit muss eine Mittelweg im Training gefunden werden, denn eine lange Trainingszeit bedeutet auch eine lange Rechenzeit (14400 Trainingsschritte mit den oben genannten CNN-Konfigurationen benötigen auf einem 2 GHz Athlon Standard-PC etwa vier Tage Rechenzeit).

Aus der Bestückung der einzelnen A und B Templates und des Schwellenwerts Z der gewonnenen CNN lassen sich kaum Rückschlüsse auf eine *gemeinsame* Struktur der Parameter ziehen, mit denen ein CNN konfiguriert werden muss, um die Aufgabe der Synchronisationsmessung erfolgreich zu meistern. Es fiel aber auf, dass die Werte der Zellen der A und B Templates und des Schwellenwerts Z das CNN für die Datenquellen GR und P3 etwa eine Größenordnung größer waren als die der CNN für die Datenquellen P1 und P2. Da davon ausgegangen werden muss, dass die Parameter der gewonnnenen CNN kein globales Minimum in den Fehlergebirgen des jeweiligen Optimierungsproblems darstellen, existieren sicherlich noch weitere und in der Struktur andere Templates und Schwellenwerte, die kleinere Minima in den jeweiligen Fehlergebirgen darstellen könnten.

Abschließend muss ergänzt werden, dass die gewonnen CNN und ihre Approximationsqualität auf ihren Testmengen eine untere Abschätzung für die Leistungsfähigkeit von Synchronisationsmessungen mit CNN sind. Die in der Datenvorverarbeitung genutzte Aufteilung in Fenster mit einer Überlagerung von 20% (siehe Kapitel 1.2) wurde nicht durchgeführt, um künstlich mehr Daten zur Berechnung zur Verfügung zu stellen. Dieses Verfahren wurde aufgrund einer Besonderheit in der Berechnung der mittleren Phasenkohärenz R durchgeführt (siehe Kapitel 1.2). Um Randeffekte in den Datenfenstern (4096 Datenpunkte pro Datenfenster) zu vermeiden, wurden bei jedem Datenfenster 10% der Datenpunkte an beiden Rändern der berechneten instantanen Phase verworfen. Damit enthielt jedes Fensterpaar der Trainingsmenge 820 Datenpunkte (410 auf jeder Seite), die in der Berechnung der Referenz-Synchronisationswerte verworfen wurden, die somit als *unbestimmt* aufgefasst werden konnten. Weiterhin existierten nicht genug Datenfensterpaare für die entsprechenden Trainingsmengen, die den exakten Synchronisationsgraden von $R = 0.3$ und $R = 0.9$ (bzw. $R = 0.5$ und $R = 0.8$) entsprachen. Deshalb wurden wie in Kapitel 4.2.4 dargestellt auch Datenfensterpaare zugelassen, deren Synchronisationsgrade nahe der oben genannten Grenzen waren. Trotz dieser beiden Faktoren wurde die Aufgabe, Synchronisationsmessungen mit CNN durchzuführen, erfolgreich durchgeführt.

4.5 Training mit dem ACE4K

Aufgrund der Anbindung des ACE4K an das SCNN System (siehe auch Anhang G.2) konnte die schaltungstechnische Realisation eines CNN im Training genutzt werden. Die Rahmenbedingungen waren gegeben, da der ACE4K CNN in einer 64×64 Topologie aufgebaut ist. Allerdings erlaubt der ACE4K CNN nur eine Nutzung von 3×3 Templates, Template Polynome 1. Ordnung. Deshalb schlug das Training fehl. Für diese Aufgabenstellung der Synchronisationsmessung werden nach den Ergebnissen des vorherigen Abschnitts polynomielle Templates benötigt. Eine Lösung dieses Problems ist die Entwicklung eines neuen ACE4K, der polynomielle Templates zur Problemlösung zur Verfügung stellt. Aber auch andere schaltungstechnische Realisationen, wie das CNN mit 72×72 Topologie und Template-Polynomen bis zur 3. Ordnung, dargestellt in [LPKH04], können in Betracht gezogen werden. Eine weitere Möglichkeit ist die Aufspaltung der CNN Synchronisationsmessung von einer CNN-Operation mit polynomiellen Templates in mehrere hintereinandergeschaltete CNN Operationen mit linearen Templates. Schönmeyer und Kollegen konnten in [Sch02] zeigen, dass die Operation eines CNN mit Templates A und B, Polynome 2. Ordnung, durch zwei hintereinander geschaltete CNN Operationen mit einfachen Templates A und B realisiert werden kann. Die Nutzung einer solchen Aufspaltung für die Synchronisationsmessung mit CNN war jedoch nicht Gegenstand dieser Arbeit und bleibt zukünftigen Untersuchungen vorbehalten.

4.6 Einflüsse von Fertigungstoleranzen auf die Approximationsqualität

Jede Zelle eines CNN besteht aus einer Kombination aus linearen (z.B. Widerstände, Kondensatoren) und nichtlinearen Bauteilen (z.B. Transistoren). Fertigungstoleranzen in schaltungstechnischen Realisationen äußern sich im Unvermögen, gleich bleibende Eigenschaften für jede Zelle eines CNN garantieren zu können. Hinzu kommt eine Temperaturabhängigkeit: Z.B. ändert sich im Volllastbetrieb die Temperatur des CNN. Dies kann z.B. zu Veränderungen in den Kennlinien der Transistoren führen. Aber auch eine erhöhte Umgebungstemperatur kann zu Veränderungen in den Arbeitsbedingungen eines CNN führen. Um diesen Problemen entgegen zu wirken, wird in der vorliegenden schaltungstechischen CNN Realisation ACE4K eine Zellgenauigkeit von 7.6 Bit angegeben. D.h. in dem definierten Betriebstemperaturbereich wird eine Genauigkeit von 128 bis 256 Stufen in der Zellauflösung von $[-1, 1]$ garantiert. Eine feinere Auflösung wird, zu Gunsten des *gleichbleibenden* Verhaltens bei verschiedenen Umgebungsbedingungen, nicht zur Verfügung gestellt. Nichts desto trotz bleibt eine kleine Unsicherheit, die Auflösung wird nicht mit 7 Bit und auch nicht mit 8 Bit beziffert, sondern mit einem Wert dazwischen.

Der SCNN-Simulator bietet aufgrund seiner hohen Genauigkeit und seiner universellen Konfigurier- und Steuerbarkeit die Möglichkeit, Einflüsse von Fertigungstoleranzen auf die Qualität der Ergebnisse der gewonnenen CNN zu simulieren. Zunächst müssen die Fertigungstoleranzen im SCNN modelliert werden. In [Ge98] und [TKGW98] wurden solche Toleranzen durch die Nutzung von *varianten* Templates und einem *varianten* Schwellen-

wert modelliert. Dieses Verfahren wurde auch in der folgenden Untersuchung durchgeführt. Dabei besteht die Grundkonfiguration aus den gegebenen Templates A und B und dem Schwellenwert Z. Variante Templates und Schwellenwerte zu nutzen bedeutet, die Grundkonfiguration zu nutzen, und jeder Zelle kl eine eigene *veränderte* Konfiguration, also eigene Templates A_{kl} und B_{kl} und einen eigenen Schwellenwert Z_{kl}, zu geben. Diese Veränderung für die Konfigurationen der einzelnen Zellen wird durch die folgenden Gleichungen beschreiben:

$$a_{ij,kl} \rightarrow a_{ij} + \Delta a_{ij,kl} \tag{4.21}$$

$$b_{ij,kl} \rightarrow b_{ij} + \Delta b_{ij,kl} \tag{4.22}$$

$$z_{kl} \rightarrow z + \Delta z_{kl} \tag{4.23}$$

mit ij =Index des Template Wertes, kl =Index der CNN Zelle

Jedem Element wird zusätzlich zu seinem Grundwert ein zusätzlicher Term hinzu addiert. $\Delta a_{ij,kl}$, $\Delta b_{ij,kl}$ und Δz_{kl} sind dabei normalverteilte mittelwertfreie Zufallszahlen. Die Standardabweichung der Normalverteilung wurde in den Untersuchungen variiert, um verschieden große Fehler zu simulieren. Dabei wurde ein Bereich von $[10^{-11}, 10]$ abgeschritten, um einerseits die Auflösungsgrenze des Simulators zu erreichen und andererseits eine Fehlergrößenordnung zu erreichen, die in die Größenordnung der durch das Training gewonnenen Template- und Schwellenwerte (Werte im Bereich $[-40, 70]$) reicht.

Die folgenden Betrachtungen wurden beispielhaft an einem CNN durchgeführt, das an der Trainingsmenge der Datenquelle P3 trainiert wurde. Die A und B Templates hatten 3×3 Topologie, Punkt-Symmetrie und waren Polynome 2. Ordnung. Es wurde die geschlossene Spirale in x-Richtung als Randbedingung gewählt. Im Training wurde eine Bewertung von $Err^{global} = 0.0479$ nach 13969 Trainingsschritten, mit einer Aufweitung von $ADev = 0.0347$, erreicht. Allerdings wurde für diese Aufgabe das einfachere Approximationsqualitätsmaß $3Cost$ genutzt:

$$3Cost = \frac{1}{\sum_{j=0}^{2} L_j} \sum_{i=0}^{2} \overline{\delta R_i} L_i \tag{4.24}$$

Dieses Maß nutzte drei Datensätze (P3d, P3f und P3g) der Testmenge der Datenquelle P3. Die Approximationsqualität lag im invarianten Fall bei $3Cost = 0.1159$.

Durchführung

Es wurden drei Szenarien untersucht. Im ersten Szenario wurden die Templates A und B und der Schwellenwert Z variant gesetzt. Im zweiten Szenario wurden nur die Templates A und B variant gesetzt, im dritten nur der Schwellenwert Z. Die Abbildungen 4.18 a.) (Szenario 1), b.) (Szenario 2) und c.) (Szenario 3) stellen die Entwicklung der Approximationsqualität gegen die Steigerung der Standardabweichung der Zufallszahlen dar. Die Approximationsqualität ohne zusätzliche Einflüsse ($3Cost = 0.1159$) wurde in jedem Graphen durch eine durchgezogene waagerechte Linie markiert.

Ergebnisse

Die Abbildung 4.18 a.) (und b.)) zeigt eine starke Empfindlichkeit des CNN auf Veränderungen seiner Templates A und B und seines Schwellenwerts Z (bzw. nur auf Veränderungen seiner Templates A und B). Selbst bei kleinsten Veränderungen trat sofort eine drastische Verschlechterung der Approximationsqualität ($3Cost > 0.22$) ein. Wurden allerdings nur am Schwellenwert Z Veränderungen vorgenommen (siehe Graph in Abbildung 4.18 c.)), dann stieg die Approximationsqualität nicht sofort in einem großen Schritt an, sondern sank zunächst ab und stieg dann beständig mit steigenden additiven Zufallszahlen.

Werden diese Ergebnisse auf schaltungstechnische Realisationen übertragen, so kann davon ausgegangen werden, dass eine Inhomogenität in der CNN Matrix (hier dargestellt durch einen varianten Schwellenwert Z mit additiven Zufallszahlen in jeder Zelle) weit weniger Probleme bereiten wird, als Inhomogenitäten in den Templates A und B. Diese Inhomogenität kann sogar durch Einbeziehung eines varianten Schwellenwerts Z durch das Training neutralisiert werden. Allerdings erhöht sich die Mächtigkeit des Parameterraums nicht um 1 (von einem invarianten Schwellenwert Z) sondern um 4096 (von einem varianten Schwellenwert Z bei einer 64×64 CNN-Topologie). Ein solch hochdimensionales Fehlergebirge kann im Training Probleme bereiten.

Die starke Verschlechterung der Approximationsqualität bei der Nutzung von varianten Templates A und B, mit additiven Zufallszahlen, läßt sich durch die komplizierte Struktur der Templates erklären. Die Templates sind in diesem Fall Polynome 2. Ordnung. Somit pflanzt sich ein kleiner Fehler nach wenigen Simulationsschritten, noch vor Erreichen des Abbruchs τ_{trans}, schnell fort. Könnte die Aufgabe der Synchronisationsmessung durch z.B. zwei hintereinander geschaltete CNN-Operationen mit jeweils einfachen Templates A und B gelöst werden (siehe Kapitel 4.5), dann würde dieser Effekt weniger zum Tragen kommen. Trotzdem würde das dann trainierte CNN sehr empfindlich auf Schwankungen in den Elementen seiner Templates A und B reagieren (siehe Untersuchungen in [Ge98] und [TKGW98]).

Zusammenfassend läßt sich sagen: Es wurde zunächst erfolgreich ein Parametertraining mit Hilfe des Synchronisationsmaßes mittlere Phasenkohärenz R durchgeführt, um Synchronisationsmessungen mit CNN durchführen zu können. Eine der vier Datenquellen wurde durch Modellsysteme erzeugt, die anderen drei enthielten EEG-Daten von Epilepsiepatienten. Die Approximationsqualität der Synchronisationsmessungen von zwei der drei mit EEG-Daten bestückten Datenquellen würde zur Definition von hypothetischen präiktalen Zuständen genügen. Weiterhin wurden die Testmengen der vier Datenquellen mit den vier gewonnenen CNN approximiert, um ein mögliches *Generalisierungsverhalten* zu untersuchen. Leider wurde solch ein Verhalten mit den gewonnenen CNN nicht erreicht. Zuletzt wurden Einflüsse von Fertigungstoleranzen in schaltungstechnischen Realisationen von CNN durch Simulationen untersucht. Es konnte gezeigt werden, dass eine hohe Empfindlichkeit in den Templates vorherrscht, während das CNN robuster gegen Schwankungen im Schwellenwert Z reagiert.

Insgesamt wurde gezeigt, dass mit der entwickelten Methode die Aufgabe der Synchronisationsmessung mit CNN möglich ist. Die Datenquellen in diesem Kapitel enthielten

Datensätze von insgesamt mehreren Stunden Dauer. Im folgenden Kapitel wird eine Datenquelle von insgesamt mehreren Tagen Dauer betrachtet. Anhand dieser Datenquelle werden weitere Untersuchungen auf Generalisierbarkeit durchgeführt. Dabei wird zunächst die Langzeitstabilität der Synchronisationsmessung mit CNN betrachtet. Weiterhin wird eine weiteren Elektrodenkombination in die Untersuchung einbezogen, deren Datenfensterpaare nicht in der Trainingsmenge vorhanden waren.

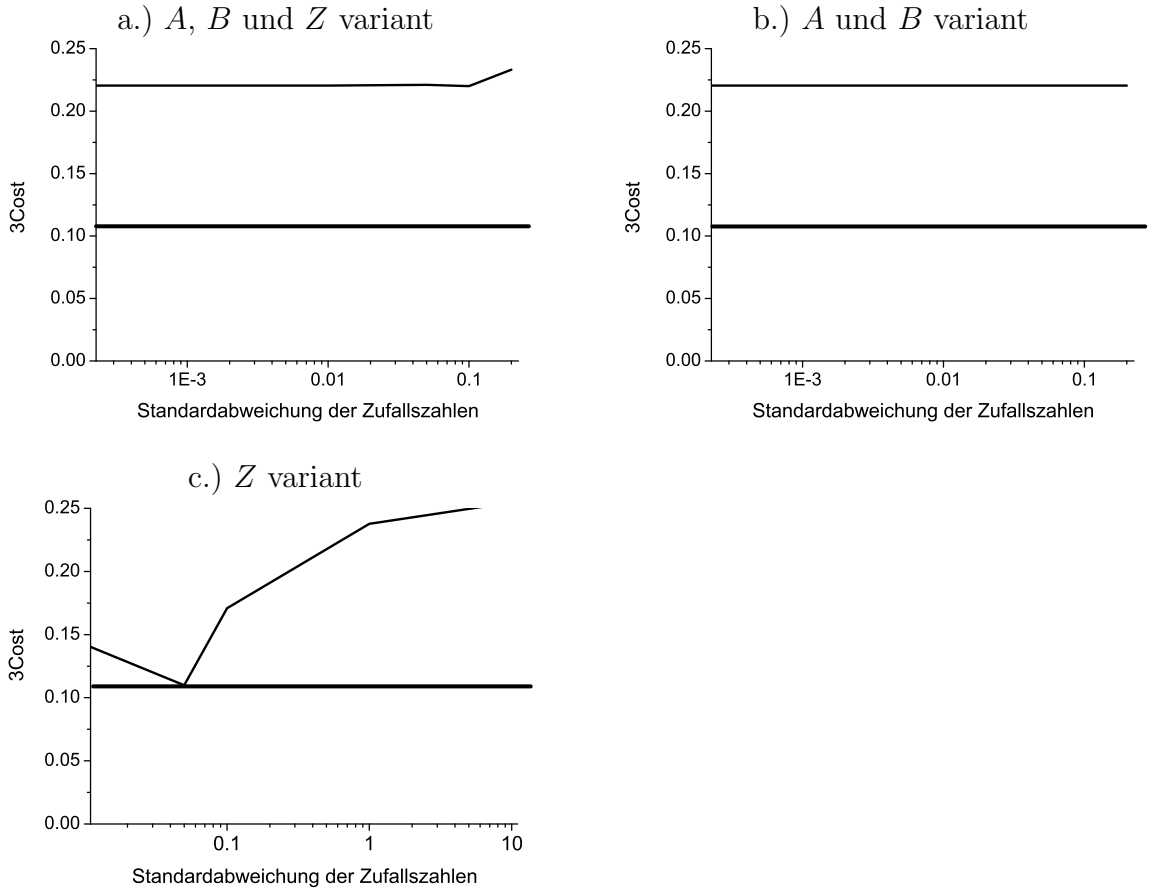


Abbildung 4.18: Approximationsqualität in Abhängigkeit der Standardabweichung der additiven Zufallszahlen (Simulation von Bauteiltoleranzen) für das Szenario 1 (variante Templates A und B und Schwellenwert Z, a.)), für das Szenario 2 (variante Templates A und B, b.)), und für das Szenario 3 (varianter Schwellenwert Z, c.)). Die Approximationsqualität ohne zusätzliche Einflüsse ($3Cost = 0.1159$) wurde in jedem Graphen durch eine waagerechte Linie markiert.

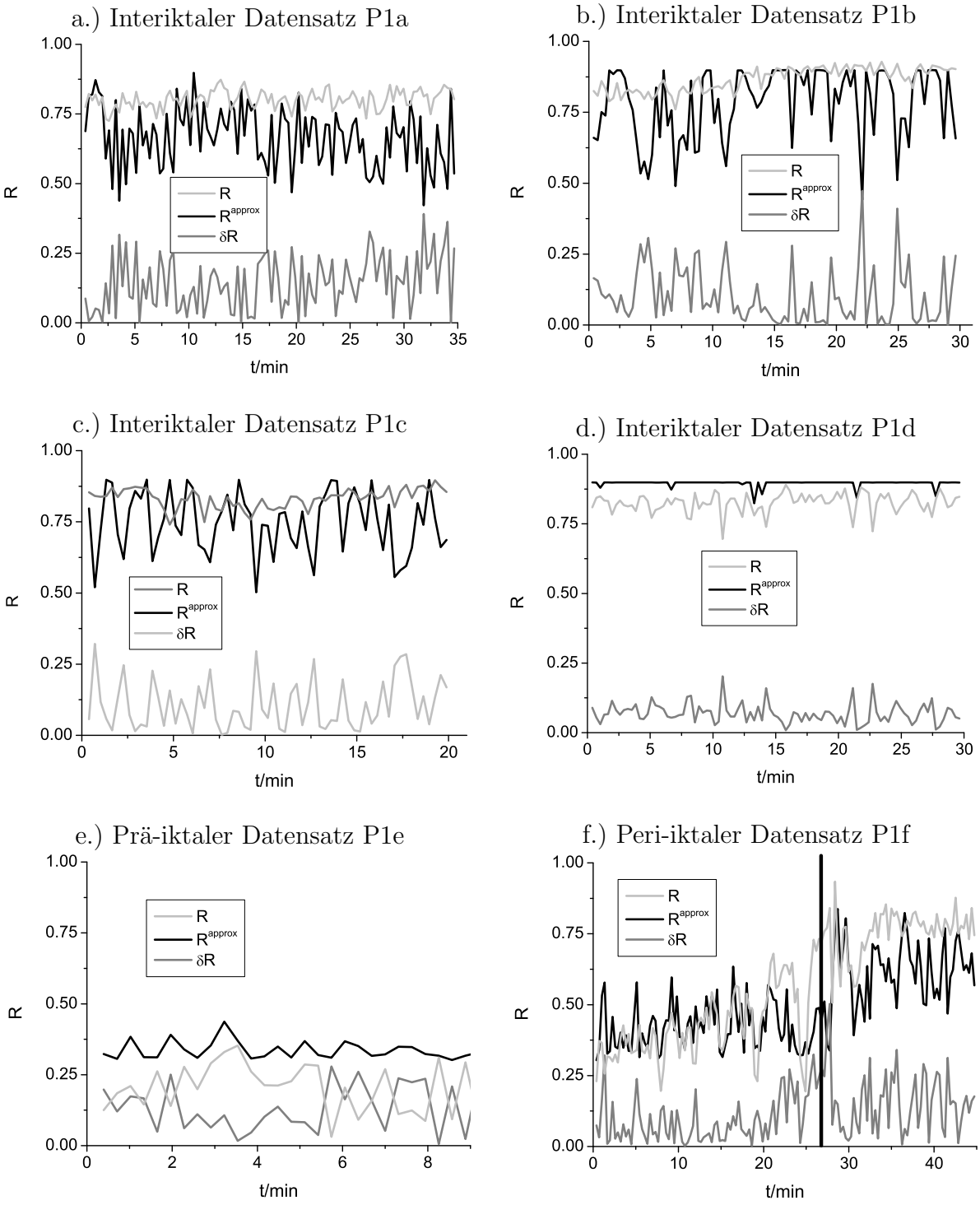
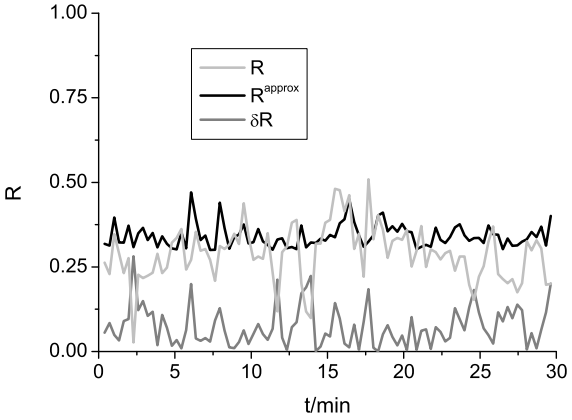
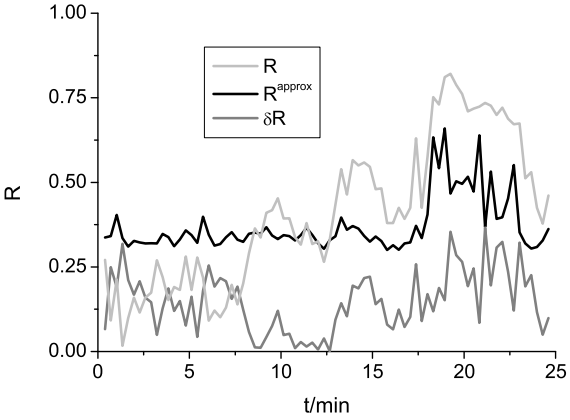


Abbildung 4.19: Zeitliche Verläufe der berechneten Synchronisationswerte (R, hellgrau), der approximierten Synchronisationswerte (R^{approx}, schwarz) und der Differenzwerte (ΔR, dunkelgrau) der Datensätze P1a (a.)) bis P1f (f.)) der Datenquelle P1. Die senkrechte Linie in Abbildung f.) markiert den elektrischen Beginn eines Anfalls.

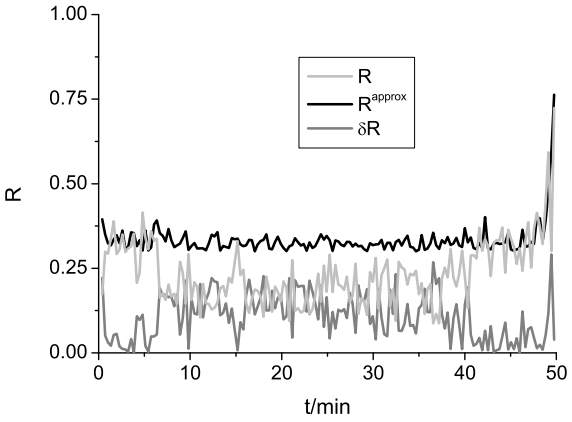
g.) Prä-iktaler Datensatz P1g



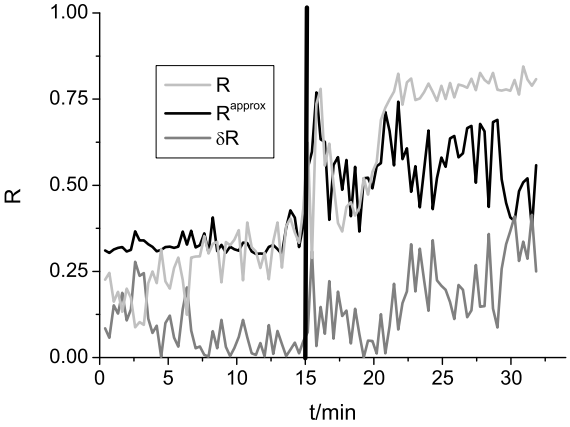
h.) Prä-iktaler Datensatz P1h



i.) Prä-iktaler Datensatz P1i



j.) Peri-iktaler Datensatz P1j



k.) Interiktaler Datensatz P1k

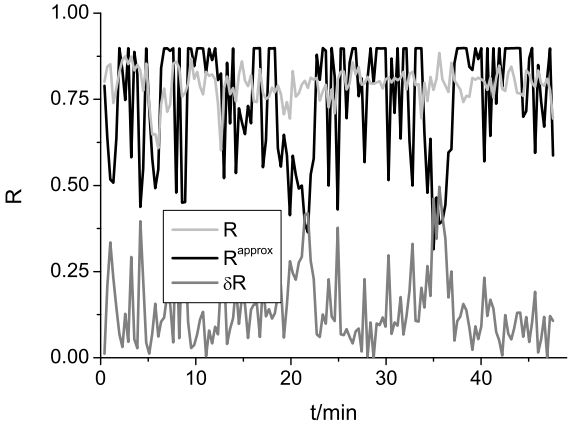


Abbildung 4.20: Wie in Abbildung 4.19, allerdings für die Datensätze P1g (g.)) bis P1k (k.)). Die senkrechte Linie in Abbildung j.) markiert den Beginn Anfalls dar.

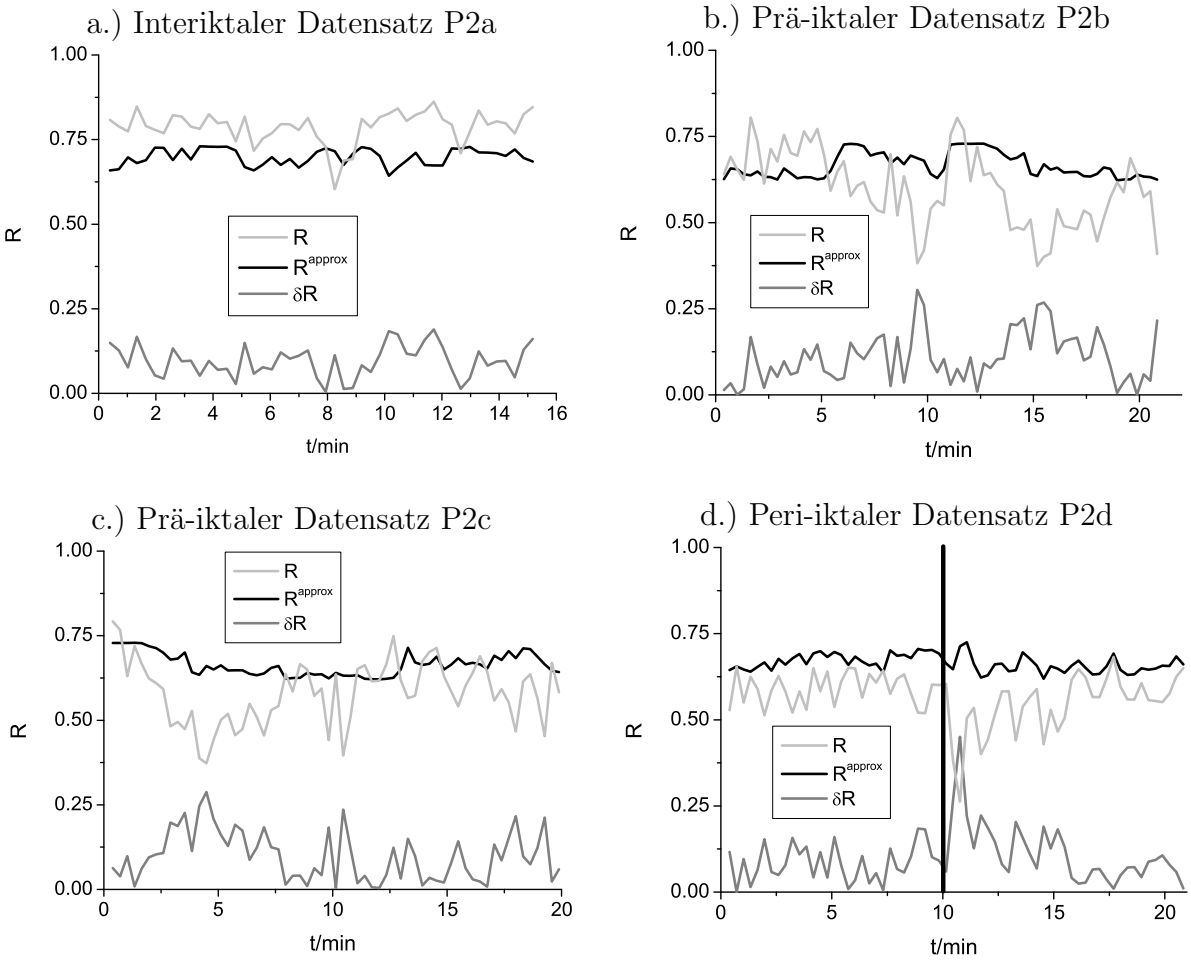


Abbildung 4.21: Zeitliche Verläufe der berechneten Synchronisationswerte (R, hellgrau), der approximierten Synchronisationswerte (R^{approx}, schwarz) und der Differenzwerte (ΔR, dunkelgrau) der Datensätze P2a (a.)) bis P2d (d.)) der Datenquelle P2. Die senkrechte Linie in Abbildung d.) markiert den elektrischen Beginn eines Anfalls.

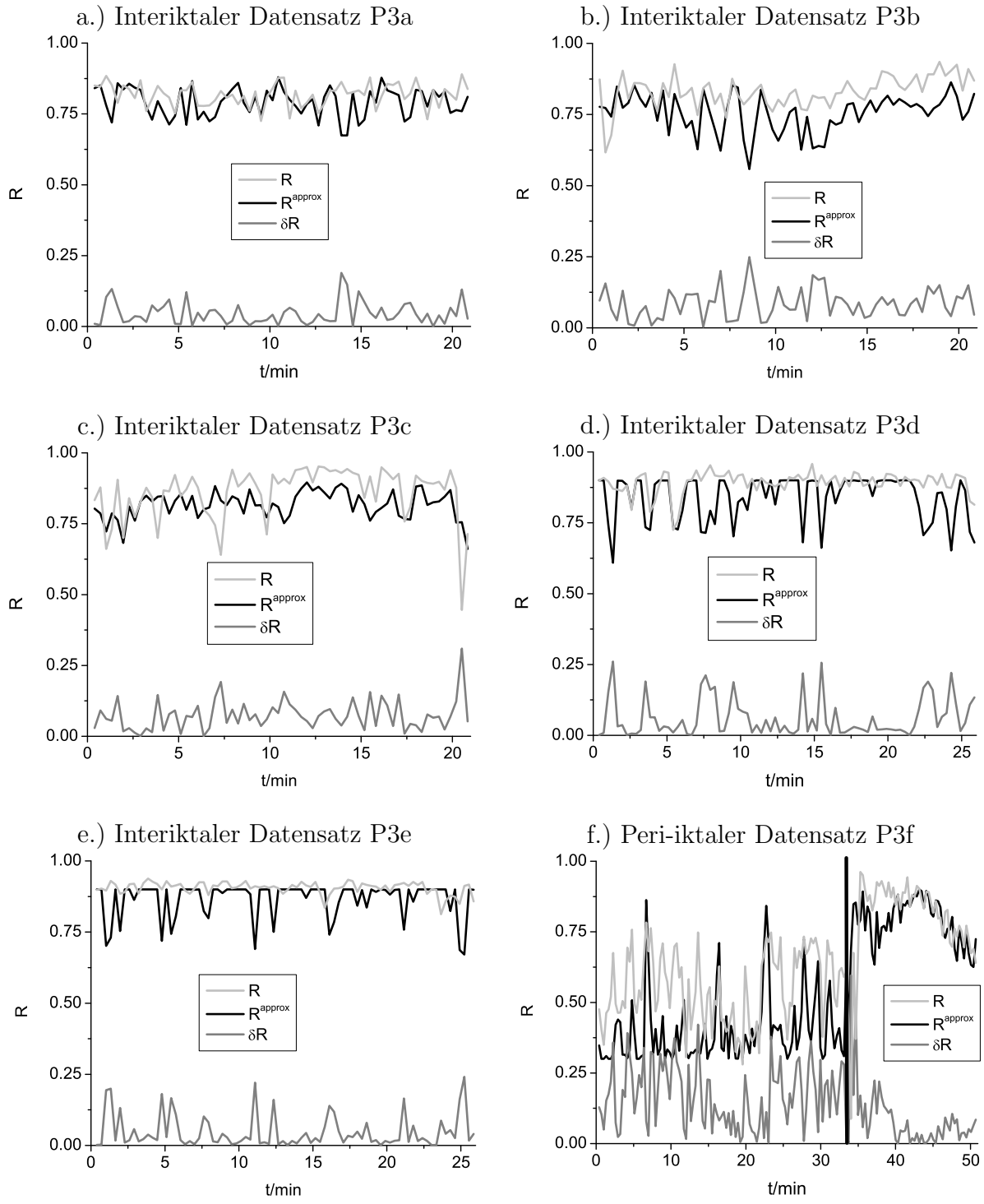


Abbildung 4.22: Zeitliche Verläufe der berechneten Synchronisationswerte (R, hellgrau), der approximierten Synchronisationswerte (R^{approx}, schwarz) und der Differenzwerte (ΔR, dunkelgrau) der Datensätze P3a (a.)) bis P3f (f.)) der Datenquelle P3. Die senkrechte Linie in Abbildung f.) markiert den elektrischen Beginn eines Anfalls.

g.) Interiktaler Datensatz P3g

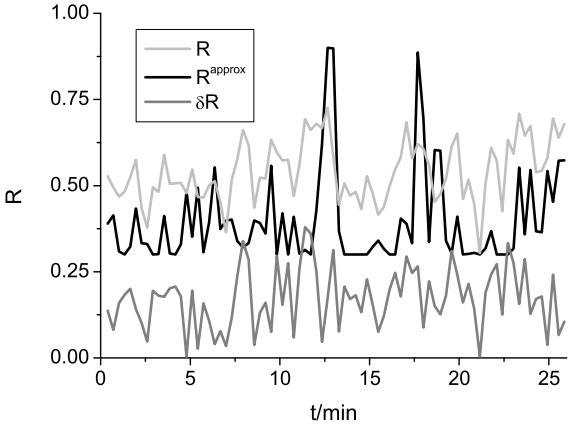


Abbildung 4.23: Wie in Abbildung 4.22, allerdings für den Datensatz P3g.

Kapitel 5

Untersuchungen zur Generalisierbarkeit

Ziel der folgenden Untersuchungen war es, die *Generalisierbarkeit* der CNN zu testen, die mit der in Kapitel 4 beschriebenen Methode zur Synchronisationsmessung in dynamischen Systemen gewonnen wurden. Dafür wurde eine weitere EEG-Datenquelle (Patient *LP*) verwendet, deren Testmenge Aufzeichnungen von insgesamt 4.4 Tagen Dauer enthält. Für das Training wurden wie in den vorigen Untersuchungen EEG-Daten von insgesamt 6 Minuten Dauer genutzt. Nachdem die Langzeitstabilität der Approximation der Synchronisationswerte überprüft wurde, wurden zwei weitere Untersuchungen durchgeführt. Zuerst wurde die Approximationsqualität des CNN, das mit der Trainingsmenge der Datenquelle LP trainiert wurde, an den Testmengen der Datenquellen GR, P1, P2 und P3, die im Kapitel 4 beschrieben wurden, ermittelt. Mit den an den Trainingsmengen der Datenquellen GR, P1, P2 und P3 gewonnenen CNN wurde genauso die Approximationsqualität für die Testmenge der Datenquelle LP ermittelt. Diese Untersuchungen entsprachen den Untersuchungen in Kapitel 4.4. In einem zweiten Schritt wurde eine weitere Testmenge der Datenquelle LP verwendet, deren Datenfenster an einer weiteren Elektrodenkombination aufgezeichnet wurden. Diese Datenfenster waren nicht in der ersten Trainings- und Testmenge enthalten.

5.1 CNN-Einstellungen

Die in dieser Untersuchung genutzten Daten sind kontinuierliche EEG-Aufzeichnungen von etwa 4.4 Tagen Dauer. Die Aufzeichnung wurde zwei mal, mit einer Dauer von 5.7 und 0.9 Stunden, aufgrund anderer Untersuchungen des Patienten unterbrochen. Diese Lücken sind in den Abbildungen der Testmenge entsprechend dargestellt. Die Abtastfrequenz dieser Datenquelle LP betrug, im Unterschied zu den in Kapitel 4 beschriebenen Datenquellen GR und P1 bis P3, 200 Hz, bei einer ADC-Auflösung von 16 Bit. Insgesamt wurden 10 Anfälle während der Datennahme aufgezeichnet.

Die Datenvorbereitung wurde wie in Kapitel 4.1.1 beschrieben durchgeführt. Der Wertebereich der EEG-Daten wurde linear von [-150 mV,150 mV] nach [-1,1] skaliert. Die Einstellungen des Simulators sowie des Trainings und die Wahl der CNN-Topologie wurden entsprechend Kapitel 4.2.1, 4.2.2 und 4.2.3 gewählt.

Der Aufbau der Trainings-/Testmenge entsprach dem Aufbau, der in Kapitel 4.2.4 beschrieben wurde. Die Datenfenster wurden aus den Daten der Kanalkombination TR8-TR9 (Trainingsmenge bzw. Testmenge TM1) gewonnen. Diese Wahl stüzt sich auf die in [MAKRDEL03] publizierten Ergebnisse. Mit dieser Kanalkombination konnte die beste Spezifität und Sensitivität für die Datenquelle LP im Hinblick auf die Definition eines hypothetischen prä-iktalen Zustands erreicht werden.

Die Trainingsmenge wurde aus jeweils gleich vielen Datenfensterpaaren mit Synchronisationswerten aus dem Bereich $[0.25, 0.35]$ und $[0.85, 0.95]$ gebildet. Auch in diesem Training wurden keine Datenfensterpaare aus den Anfalls- oder den post-iktalen Zuständen verwendet. Das Parameter-Training wurde entsprechend Kapitel 4.3 durchgeführt. Die Abbildung 5.1 stellt die Verteilung der errechneten Synchronisationswerte der Datenquelle LP für die Elektrodenkombination TR08-TR09 dar.

Ein weiterer Unterschied zu den Einstellungen in Kapitel 4 lag in der Definition der Testmenge. Alle Datenfensterpaare, auch die aus den iktalen und post-iktalen Zuständen, wurden in die Testmenge aufgenommen, um Aussagen zur zeitlichen Stabilität der approximierten Werte über den gesamten Zeitraum treffen zu können. Wegen der zuvor zweiten Überprüfung der Generalisierbarkeit wurde eine weitere Testmenge (TM2) generiert, deren Datenfensterpaare an der Elektrodenkombination TL08-TL09 aufgezeichnet wurden. Die Daten dieser Testmenge stammten also aus der anderen Hemisphäre des Gehirns. Abbildung 5.2 stellt die Verteilung der errechneten Synchronisationswerte der Datenquelle LP dieser Elektrodenkombination dar.

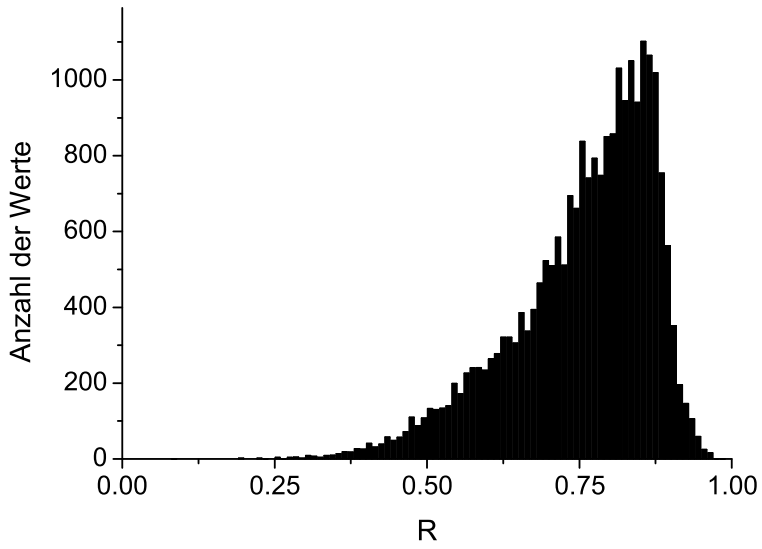


Abbildung 5.1: Verteilung der Synchronisationswerte (R) der Datenquelle LP für die Kanalkombination TR08-TR09

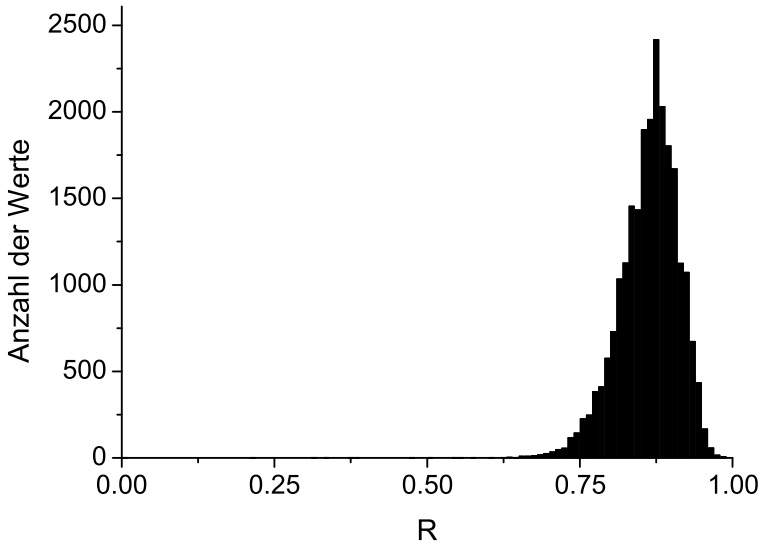


Abbildung 5.2: Verteilung der Synchronisationswerte (R) der Datenquelle LP für die Kanalkombination TL08-TL09

5.2 Optimierungsergebnis

Mit den folgenden CNN-Einstellungen wurde eine optimale Bewertung von $Err^{global} = 0.0889$ nach 14396 Trainingsschritten, mit einer Aufweitung von $ADev = 0.0549$, erreicht: Es wurden eine 8-8 Trainingsmenge, das Iterative-Annealing ($T_0 = 50$, 300 Abkühlschritte) als Optimierungsverfahren, 14400 Trainingsschritte, der Mittelwert als Zusammenführungsfunktion, $MSEn$ als Bewertungsfunktion, und die CNN-Topologie 1 als Einstellungen genutzt. Die A und B Templates des gewonnenen besten CNN hatten eine 3×3 Topologie, Punkt-Symmetrie und waren Polynome 2. Ordnung. Es wurde die Randbedingung geschlossene Spirale in x-Richtung gewählt. In der Abbildung 5.3 ist die Bewertung in Abhängigkeit der Trainingsschritte dargestellt.

Im Trainingsverlauf der Datenquelle LP fand in den ersten 3000 Trainingsschritten eine massive Änderung statt. Die Bewertung verbesserte sich von $Err^{global} = 0.4300$, Trainingsschritt 1 auf $Err^{global} = 0.1054$, Trainingsschritt 3000. Diesem starken Abstieg folgte zunächst keine Änderung. Erst in den letzten Trainingsschritten wurde eine Bewegung zum finalen Err^{global} erkennbar. Solch ein Verhalten wurde im Kapitel 4.4.2 für die Datenquelle GR beschrieben. Auch im Fall der Datenquelle LP war dieses Verhalten zu erwarten, da nicht davon ausgegangen werden konnte, die jeweiligen Templates und den Schwellenwert (CNN-Parameter) zu finden, die dem globalen Minimum in dem Fehlergebirge (durch die Trainingsmenge der Datenquelle LP und deren CNN-Konfiguration bestimmt) entsprachen.

Für die Datenquelle LP wurde die Approximationsqualität durch die mittlere Differenz der berechneten und approximierten Synchronisationswerte definiert. Im Gegensatz zur Definition in Kapitel 4.2.5 wurde nicht das vorzeichenlose $\delta R_i = |R_i - R_i^{approx}|$ (mit i als Positionsindex in der Zeitreihe), sondern das vorzeichenbehaftete $\Delta R_i = R_i - R_i^{approx}$

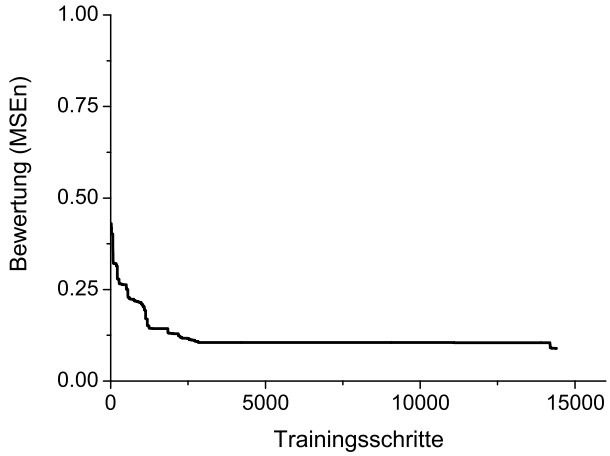


Abbildung 5.3: Trainingsverlauf für die EEG-Datenquelle LP

verwendet, um Tendenzen im Verlauf der Differenz der berechneten und approximierten Synchronisationswerte über den gesamten Aufzeichnungszeitraum erkennen zu können. In Kapitel 4 hingegen wurde das Hauptaugenmerk auf die Unterscheidbarkeit zwischen hohen und niedrigen Synchronisationswerten und damit bei den EEG-Datenquellen auf die Möglichkeit der Definition eines hypothetischen prä-iktalen Zustandes gelegt. Für diese Betrachungen genügte die vorzeichenlose Differenz δR_i.

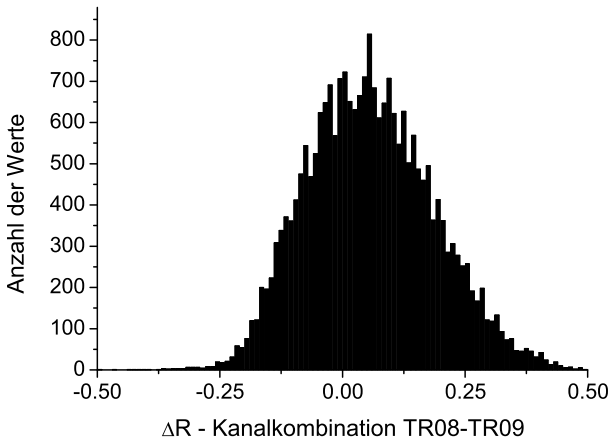


Abbildung 5.4: Verteilung der Differenzen (ΔR) der berechneten und der approximierten Synchronisationswerte der Testmenge TM1 (Elektrodenkombination TR08-TR09) der Datenquelle LP, approximiert mit dem CNN LP

Die Approximationsqualität der Testmenge TM1 lag bei $\overline{\Delta R} = 0.0516$ mit einer Standardabweichung von $\sigma = 0.1287$ (vgl. Abbildung 5.4). Die Approximationsqualität der Testmenge TM2 lag bei $\overline{\Delta R} = 0.0008$ mit einer Standardabweichung von $\sigma = 0.0703$ (vgl. Abbildung 5.5).

Darstellung der gewonnenen Parameter

Die gewonnenen Templates A und B und der Schwellenwert Z, die durch das Parameter-Training als optimale Werte für die Datenquelle LP gewonnen wurden, sind in den Gleichungen 5.1, 5.2 und 5.3 wiedergegeben. In jeder Zelle der 3×3 Templates A und B sind jeweils zwei Werte geklammert eingetragen. Der obere Wert ist jeweils der erste und der untere der zweite Eintrag des Template-Polynoms.

$$A = \left\{ \begin{array}{ccc} \begin{array}{c}(0.34551662529\\ -1.35540085291)\end{array} & \begin{array}{c}(0.340944177905\\ 0.0546494981235)\end{array} & \begin{array}{c}(0.990001227694\\ 0.024294622563)\end{array} \\ \\ \begin{array}{c}(-0.0782314055046\\ 0.12382029677)\end{array} & \begin{array}{c}(-0.592563556054\\ -0.30230839852)\end{array} & \begin{array}{c}(-0.0782314055046\\ 0.123820296776)\end{array} \\ \\ \begin{array}{c}(0.990001227694\\ 0.024294622563)\end{array} & \begin{array}{c}(0.340944177905\\ 0.0546494981235)\end{array} & \begin{array}{c}(0.34551662529\\ -1.35540085291)\end{array} \end{array} \right\} \quad (5.1)$$

$$B = \left\{ \begin{array}{ccc} \begin{array}{c}(-0.12159977125\\ 1.01315123117)\end{array} & \begin{array}{c}(0.0658724361623\\ -0.764929017977)\end{array} & \begin{array}{c}(-0.412007335522\\ 0.617496533908)\end{array} \\ \\ \begin{array}{c}(-0.478453681029\\ -0.365640226291)\end{array} & \begin{array}{c}(0.533628657649\\ 0.109559344882)\end{array} & \begin{array}{c}(-0.478453681029\\ -0.365640226291)\end{array} \\ \\ \begin{array}{c}(-0.412007335522\\ 0.617496533908)\end{array} & \begin{array}{c}(0.0658724361623\\ -0.764929017977)\end{array} & \begin{array}{c}(-0.12159977125\\ 1.01315123117)\end{array} \end{array} \right\} \quad (5.2)$$

$$Z = \left\{ \; 1.12704285438 \; \right\} \quad (5.3)$$

Darstellung der Testmengen TM1 und TM2

Die Abbildungen 5.9 und 5.12 stellen jeweils die zeitlichen Verläufe der berechneten Synchronisationswerte R (Referenzwerte) der Testmengen TM1 und TM2 dar. Die Abbildungen 5.10 und 5.13 stellen entsprechend die zeitlichen Verläufe der jeweils durch das CNN approximierten Synchronisationswerte R^{approx} und die Differenzen ΔR beider Testmengen dar. Für eine übersichtlichere Darstellung der Ergebnisse wurde auf die Daten ein *10 Punkte-Rückwärts-Filter* angewendet. Dabei wurde jeweils für die berechneten Synchronisationswerte, für die approximierten Synchronisationswerte und für die Differenzwerte zum Zeitpunkt t 10 Werte aus dem Bereich $[t-9, t]$ gemittelt. Die Aufzeichungslücken wurden entsprechend angepaßt.

Die Differenzen der berechneten und approximierten Synchronisationswerte (ΔR) beider Testmengen wiesen im zeitlichen Verlauf der ersten 1.5 Tage der Datenaufzeichnung stabile Schwankungen um die entsprechenden mittleren Differenzen ($\overline{\Delta R} = 0.0516$ und $\overline{\Delta R} = 0.0008$) auf. Bei den folgenden stärkeren Änderungen in den Dynamiken, dargestellt durch Änderungen in den Synchronisationswerten, entwickelte sich ein Verhalten von lokaler Divergenz. Die Differenzen der berechneten und approximierten Synchronisationswerten drifteten vom mittleren Differenzwert weg. Die Dauer dieser lokalen Divergenzen schwankte für die Daten der Testmenge TM1 im Bereich von $[0.3, 0.75]$ Tagen und für die Daten

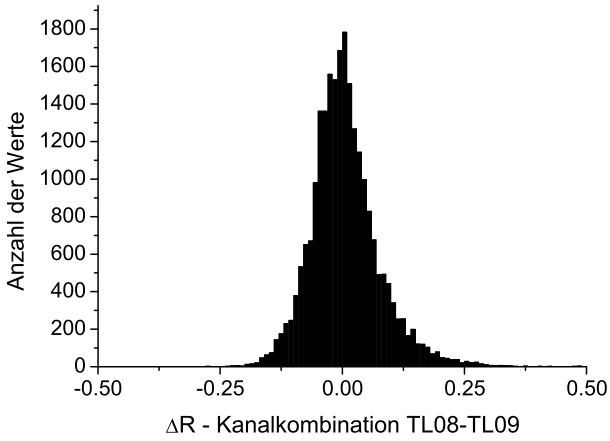


Abbildung 5.5: Verteilung der Differenzen (ΔR) der berechneten und der approximierten Synchronisationswerte der Testmenge TM2 (Elektrodenkombination TL08-TL09) der Datenquelle LP, approximiert mit dem CNN LP

der Testmenge TM2 im Bereich von $[0.5, 1.4]$ Tagen. Global betrachtet hoben sich die Divergenzen jedoch in beiden Fällen über den gesamten Aufzeichnungszeitraum auf, so dass insgesamt von einem stabilen Langzeitverhalten in beiden Testmengen gesprochen werden kann.

Es war auffällig, dass die Bewertung der Approximationsqualität der Testmenge TM2 ($\overline{\Delta R} = 0.0008$) um zwei Größenordnungen besser ausfiel als die der Testmenge TM1 ($\overline{\Delta R} = 0.0516$). Weiterhin wurde auch eine deutlich kleinere Aufweitung der Verteilung der Differenzwerte festgestellt (vgl. Abbildung 5.4 und 5.5; Testmenge TM1: $\sigma = 0.1287$; Testmenge TM2: $\sigma = 0.0703$). Die berechneten Synchronisationwerte der Testmenge TM2 stammen hauptsächlich aus dem Bereich $R \in [0.75, 0.95]$ (vgl. Abbildung 5.2). Die Berechnung solch hoher Synchronisationswerte kann auf die Tatsache zurückgeführt werden, dass die EEG-Daten weit entfernt vom Anfallsgeschehen in der anderen Hemisphäre des Gehirns aufgezeichnet wurden. Sie enthalten eher Informationen über physiologische Dynamiken aus den Bereichen des Gehirns, die nicht direkt vom Anfallsgeschehen beeinflußt werden (vgl. Abbildung 5.12). Das CNN wurde darauf trainiert, Synchronisationswerte aus dem Bereich $R \in [0.3, 0.9]$ zu approximieren, so dass die Synchronisationswerte der Testmenge TM2 sehr häufig die obere Auflösugsgrenze erreichen oder überschreiten und dann vom CNN korrekt mit $R^{approx} = 0.9$ approximiert werden. Damit läßt sich die hervorragende Approximationsqualität und die geringe Aufweitung der Differenzen der Testmenge TM2 im Vergleich zur Testmenge TM1 mit dem Erreichen der Auflösungsgrenze des gewonnenen CNN erklären. Synchronisationswerte aus dem Bereich $R \in [0.9, 1.0]$ wurden beim Training des CNN nicht betrachtet, da sie nur in hochsynchronisierten dynamischen Systemen von Interesse sind und keine zusätzliche Informationen zu der hier gestellten Aufgabe der Synchronisationsmessung in EEG-Zeitreihen von Epilepsiepatienten und damit der Möglichkeit eines Definition von prä-iktalen Zustands bieten.

Für eine weitere Untersuchung wurde die zeitliche Entwicklung der berechneten und approximierten Synchronisationswerte jeweils zwei Stunden vor Anfallsbeginn betrachtet. Diese Untersuchung wurde anhand der Testmenge TM1 durchgeführt. Die Abbildungen 5.14 und 5.15 stellen die jeweiligen Zeiträume separat dar. In diesen Abbildungen wurde die oben genannte Filterung nicht durchgeführt. Anfall 9 und Anfall 10 lagen nur etwa 90 Minuten auseinander, so dass in den entsprechenden Abbildungen eine Überschneidung von 30 Minuten enthalten ist.

Der durchschnittliche berechnete Synchronisationswert betrug für die interiktalen Zeiträume des Datensatzes LP $\overline{R}_{interiktal} = 0.7707$. Zur Berechnung dieses Wertes wurden die Anfälle, die hypothetischen prä-iktalen Zeiträume (jeweils zwei Stunden vor einem Anfall) und die post-iktalen Zeiträume (jeweils eine Stunde nach einem Anfall) ausgelassen. Die Wahl der Dauer dieser Zeiträume stützt sich auf die in [MAKRDEL03] publizierten Ergebnisse.

Die berechneten Synchronisationswerte unterschritten in drei Zeiträumen (vor den Anfällen 6, 7 und 10) 90 Minuten lang und länger deutlich den durchschnittlichen interiktalen Synchronisationswert. Für die approximierten Synchronisationswerte wurde solch ein Verhalten in vier Zeiträumen (vor den Anfällen 3, 4, 6 und 10) erkennbar. Die berechneten Synchronisationswerte unterschritten in zwei Zeiträumen (vor den Anfällen 5 und 9) je etwa 60 Minuten lang deutlich den durchschnittlichen interiktalen Synchronisationswert, während dieses Verhalten für die approximierten Synchronisationswerte in drei Zeiträumen (vor den Anfällen 5, 7 und 8) auftrat. Damit konnte das 60 Minuten und mindestens 90 Minuten lange Unterschreiten des durchschnittlichen interiktalen Synchronisationswerts sowohl bei den berechneten als auch bei den approximierten Werten ähnlich oft, aber bei teilweise unterschiedlichen Fällen beobachtet werden. Allerdings trat bei den approximierten Synchronisationswerten eine Aufweitung auf und minderte damit die Deutlichkeit des Unterschreitens. Während z.B. des Zeitraumes vor den vierten Anfall (vgl. Abbildung 5.14) wurden Abweichungen erkennbar, die mehrfach den durchschnittlichen interiktalen Synchronisationswert erreichten oder überschritten. Weiterhin wurden nur in drei Zeiträumen (vor den Anfällen 5, 6 und 7) ein ähnliches Verhalten der zeitlichen Entwicklung der berechneten und approximierten Synchronisationswerte festgestellt. Hier wurde wieder, durch die oben genannte Aufweitung der approximierten Synchronisationswerte, ein genaue Übereinstimmung verhindert.

Mit den approximierten Synchronisationswerten konnte ein Unterschreiten des durchschnittlichen interiktalen Synchronisationswerts in sieben von zehn hypothethischen präiktalen Zeiträumen nachgewiesen werden. Das kann als ein ermutigendes erstes Ergebnis angesehen werden. Allerdings verdeutlicht es ein weiteres Mal, dass im Training nicht die optimalen Parameter, sondern Parameter, die einem lokalen Minimum im Fehlergebirge entsprachen, gefunden wurden. Um genauere Aussagen tätigen zu können, müssen diese Ergebnisse an Daten eines größeren Patientenkollektiv überprüft werden.

Generalisierungsverhalten mit anderen Datenquellen

In der folgenden Untersuchung wurde die Approximationsqualität des gewonnenen CNN LP mit den Testmengen der Datenquellen GR, P1, P2 und P3 berechnet und die hieraus

gewonnen CNN auf die Testmenge TM1 der Datenquelle LP angewendet. Damit wurde, wie in der Einleitung des Kapitels erwähnt, ein weiteres mal das Generalisierungsverhalten untersucht, allerdings zwischen fünf verschiedenen Datenquellen und deren CNN. In der Tabelle 5.1 ist die Approximationsqualität der Testmengen der einzelnen Datenquellen LP, GR und P1 bis P3 unter Nutzung der fünf gewonnenen (siehe auch Tabelle 4.7) dargestellt.

	$\overline{\Delta R}$ LP TM1	*Cost* GR	*Cost* P1	*Cost* P2	*Cost* P3
CNN LP	0.0516	0.2674	0.1425	0.1928	0.0661
CNN GR	0.1937	0.1468	0.4458	0.1691	0.2571
CNN P1	0.2089	0.2608	0.1055	0.1596	0.2020
CNN P2	0.2326	0.3114	0.2924	0.0979	0.1734
CNN P3	0.1315	0.2601	0.1999	0.1849	0.0934

Tabelle 5.1: Generalisierungsverhalten der fünf gewonnenen CNN mit den Datenquellen LP, GR und P1 bis P3

Diese Untersuchung zeigte, dass die gewonnenen CNN mit einer Ausnahme nur die Testmengen der Datenquellen, aus denen ihre Trainingsmenge gewonnen wurde, erfolgreich approximieren konnten (vgl. Diagonale in der Tabelle 5.1). Die Approximationen der Testmengen der anderen Datenquellen wurden äußerst schlecht bewertet. Die Ausnahme bestand in der Anwendung des CNN LP auf die Testmenge der Datenquelle P3. Hier konnte eine hohe Approximationsqualität von $Cost = 0.0661$ erreicht werden. Damit wurde eine bessere Bewertung erreicht als bei der Anwendung des CNN P3 auf die Testmenge der Datenquelle P3 ($Cost = 0.0934$). Dieser Erfolg kann nicht mit patientenspezifischen Charakteristika erklärt werden (wie z.B. demographische Daten, Art der Pathologie, etc.), da diese für die fünf Patienten im wesentlichen vergleichbar waren. Statt dessen stützt dieses Ergebnis die Aussage, dass die in dieser Arbeit gewonnenen Einstellungen nicht den globalen Minima entsprechen und dass es weitere Konfigurationen gibt, die eine bessere Approximationsqualität bieten (CNN LP mit 3×3 Templates mit Punkt-Symmetrie und Polynome 2. Ordnung; CNN P3 mit 3×3 Templates mit Stern-Symmetrie und Polynome 3. Ordnung). Damit könnte also in der Datenquelle P3 auch unter Verwendung des CNN LP ein hypothetischer prä-iktualer Zustand definiert werden. Die Abbildungen 5.6 a.) und b.) stellen beispielhaft die berechneten Synchronisationswerte (Referenzwerte), die durch das CNN LP approximierten Synchronisationswerte und die Differenzwerte eines interiktalen (P3d) und eines peri-iktalen Datensatzes (P3f) der Datenquelle P3 dar.

Da sich die CNN-Topologie 1 beim Training durchgesetzt hatte, wurde die oben genannte Untersuchung des Generalisierungsverhaltens mit anderen Datenquellen erneut, jedoch unter Vertauschung der Zeitreihensegmente durchgeführt. Dabei wurde in diesem Fall das Zeitreihensegment 1 in den Status $X(0)$ und das Zeitreihensegement 2 in die Eingabe U geladen (siehe auch Abbildung 4.4). In Tabelle 5.2 ist die so gewonnene Approximationsqualität der einzelnen Testmengen der Datenquellen LP, GR und P1 bis P3 durch die Nutzung der fünf gewonnenen CNN (siehe auch Tabelle 4.7) dargestellt.

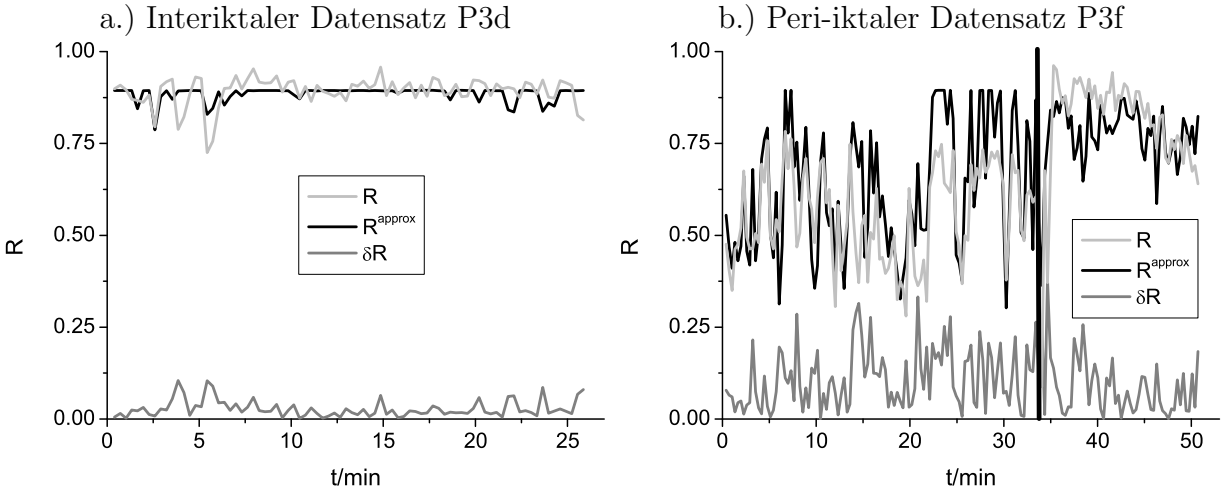


Abbildung 5.6: Zeitliche Verläufe der berechneten Synchronisationswerte (R, hellgrau), der approximierten Synchronisationswerte (R^{approx}, schwarz) und der Differenzwerte (ΔR, dunkelgrau) des interiktalen Datensatzes P3d (a.)) und des peri-iktalen Datensatzes P3f (b.)) der Datenquelle P3 unter Anwendung des CNN LP. Die senkrechte Linie in Abbildung b.) markiert den elektrischen Beginn eines Anfalls.

	$\overline{\Delta R}$ LP TM1	*Cost* GR	*Cost* P1	*Cost* P2	*Cost* P3
CNN LP	0.0056	0.2452	0.2026	0.1148	0.0972
CNN GR	0.1609	0.2528	0.4368	0.1559	0.2604
CNN P1	0.1430	0.2613	0.1941	0.3241	0.2098
CNN P2	0.2240	0.3112	0.3054	0.0911	0.1706
CNN P3	0.1528	0.2556	0.2838	0.1502	0.1636

Tabelle 5.2: Generalisierungsverhalten der fünf gewonnenen CNN mit den Datenquellen LP1, GR und P1 bis P3, mit vertauschter Eingabe U und Status $X(0)$

Die Untersuchung zeigte, dass die approximierten Synchronisationswerte der Testmengen bis auf zwei Ausnahmen mit einer schlechten Approximationsqualität bewertet wurden. Die zwei Ausnahmen CNN LP mit der Testmenge TM1 und CNN LP mit der Testmenge der Datenquelle P3 sind interessante Ergebnisse. Selbst unter der Vertauschung der Inhalte der Eingabe U und des Status $X(0)$, und damit unter einem erheblichen Eingriff in den Konfigurationen beider Fehlerräume, entsprachen die Parameter des CNN LP einem lokalen Minimum in beiden Fehlerräumen. Beide Fälle sind weitere Indizien für das Generalisierungsverhalten von CNN.

Damit konnte also in der Datenquelle P3 unter Verwendung des CNN LP auch unter Vertauschung der Eingabe U und des Status $X(0)$ ein hypothetischer prä-iktualer Zustand definiert werden. Die Abbildungen 5.7 a.) und b.) stellen beispielhaft die berechneten Synchronisationswerte (Referenzwerte), die durch das CNN LP approximierten Synchronisationswerte und die Differenzwerte eines interiktalen (P3d) und eines peri-iktalen Datensatzes

(P3f) der Datenquelle P3 dar.

Weiterhin konnte das CNN LP seine Testmenge TM1 mit einer hohen Qualität approximieren, trotz der Vertauschung der Inhalte der Eingabe U und des Status $X(0)$. Dabei wurde die Approximationsqualität sogar um eine Größenordnung übertroffen (CNN LP mit Testmenge TM1: $\overline{R} = 0.0516$; CNN LP mit Testmenge TM1 unter Vertauschung der Eingabe U und des Status $X(0)$: $\overline{R} = 0.0056$). Die Aufweitungen beider Fälle bewegten sich im gleichen Bereich ($\sigma = 0.1265$ bzw. $\sigma = 0.1287$). Abbildung 5.11 stellt den zeitlichen Verlauf der approximierten Synchronisationswerte (R^{approx}) und der Differenzen (ΔR) der berechneten und approximierten Synchronisationswerte der Testmenge TM1 dar. Auch für diese Abbildung wurde für eine übersichtlichere Darstellung der Ergebnisse ein *10 Punkte-Rückwärts-Filter* angewendet. Im Vergleich zu den approximierten Synchronisationswerten für den Fall ohne Vertauschung der Inhalte der Eingabe U und des Status $X(0)$ (vgl. Abbildung 5.10), wurden die Änderungen in der zeitlichen Entwicklung der berechneten Synchronisationswerte (vgl. Abbildung 5.9) besser approximiert. Dieses Ergebnis ist aufgrund der hohen Approximationsqualität zu erwarten. Genauso wie im dargestellten Fall der Anwendung des CNN LP auf die Testmenge TM1 ohne Vertauschung der Inhalte der Eingabe U und des Status $X(0)$ wiesen die Differenzen der berechneten und approximierten Synchronisationswerte im zeitlichen Verlauf der ersten 1.5 Tage der Datenaufzeichnung stabile Schwankungen um den mittleren Differenzwert ($\overline{R} = 0.0056$) auf. Danach wurde wieder ein Verhalten von lokaler Divergenz beobachtbar. Die Dauer dieser lokalen Divergenzen schwankte ähnlich zum vorher betrachteten Fall im Bereich von $[0.2, 0.6]$ Tagen. Sie hoben sich über den gesamten Zeitraum auf, so dass auch hier von einem stabilen Langzeitverhalten gesprochen werden kann. Damit konnte die Testmenge TM1 mit einer Änderung in der CNN-Konfiguration besser approximiert werden als im trainierten Fall. Die Abbildung 5.8 stellt die Verteilung der Differenzen der berechneten und approximierten Synchronisationswerte unter Vertauschung der Inhalte der Eingabe U und des Status $X(0)$ dar.

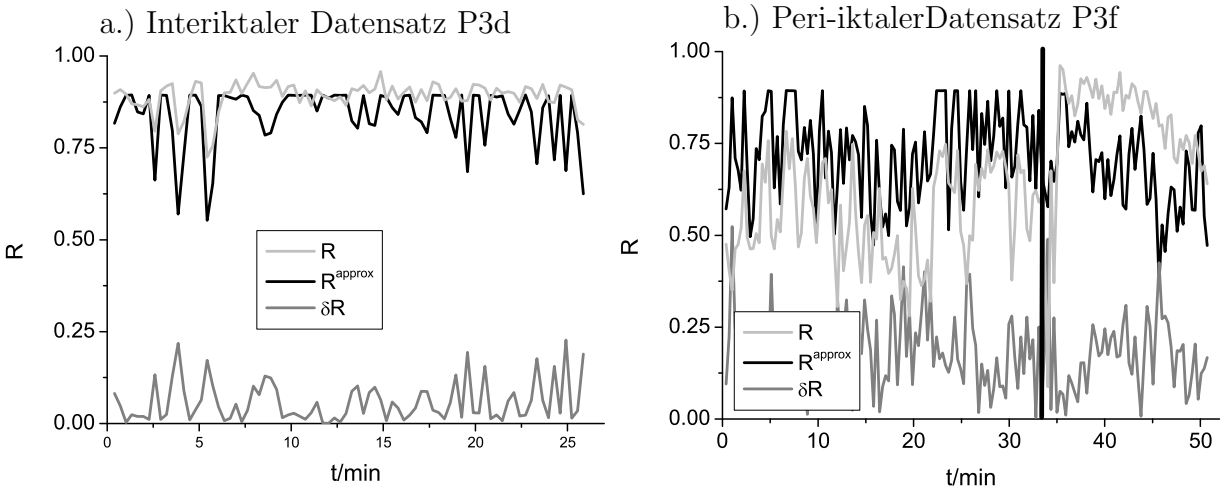


Abbildung 5.7: Wie in Abbildung 5.6, allerdings mit Vertauschung der Inhalte der Eingabe U und des Status $X(0)$.

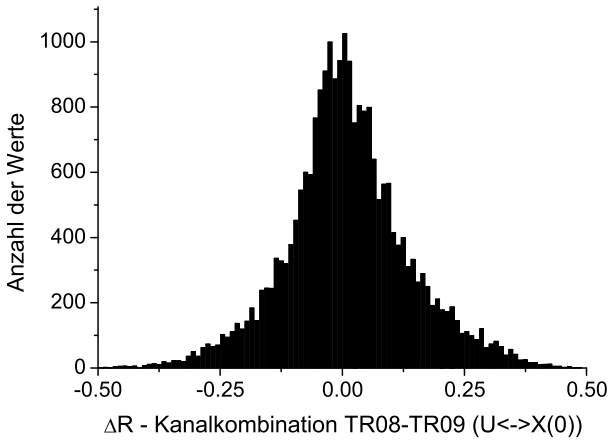


Abbildung 5.8: Verteilung der Differenzen (ΔR) der berechneten und der approximierten Synchronisationswerte der Testmenge TM1 (Elektrodenkombination TR08-TR09) der Datenquelle LP, approximiert mit den CNN LP unter Vertauschung der Inhalte der Eingabe U und des Status $X(0)$

Zusammenfassend läßt sich sagen: Der in Kapitel 4 beschriebenen Methodik und den daraus gewonnenen CNN konnte bei ersten Untersuchungen mit einem kontinuierlichen Datensatz von 4.4 Tagen Dauer ein gutes Generalisierungsverhalten zugesprochen werden. Durch die Annahme eines hypothetischen prä-iktalen Zustandes von zwei Stunden Dauer konnte in 7 von 10 Anfällen eine deutliche Unterschreitung des mittleren berechneten Synchronisationswertes aller interiktalen Zeiträume vor einem Anfall beobachtet werden.

Anhand dieser ermutigenden Ergebnisse kann ein exemplarischer CNN-Algorithmus entwickelt werden, mit dem alle Aufgaben, von der Approximation einer Kenngröße bis zur Entscheidungsfindung, ob ein prä-iktualer Zustand vorliegt oder nicht, in und mit einem CNN durchgeführt werden können. Solch ein Algorithmus wird im folgenden Kapitel dargestellt.

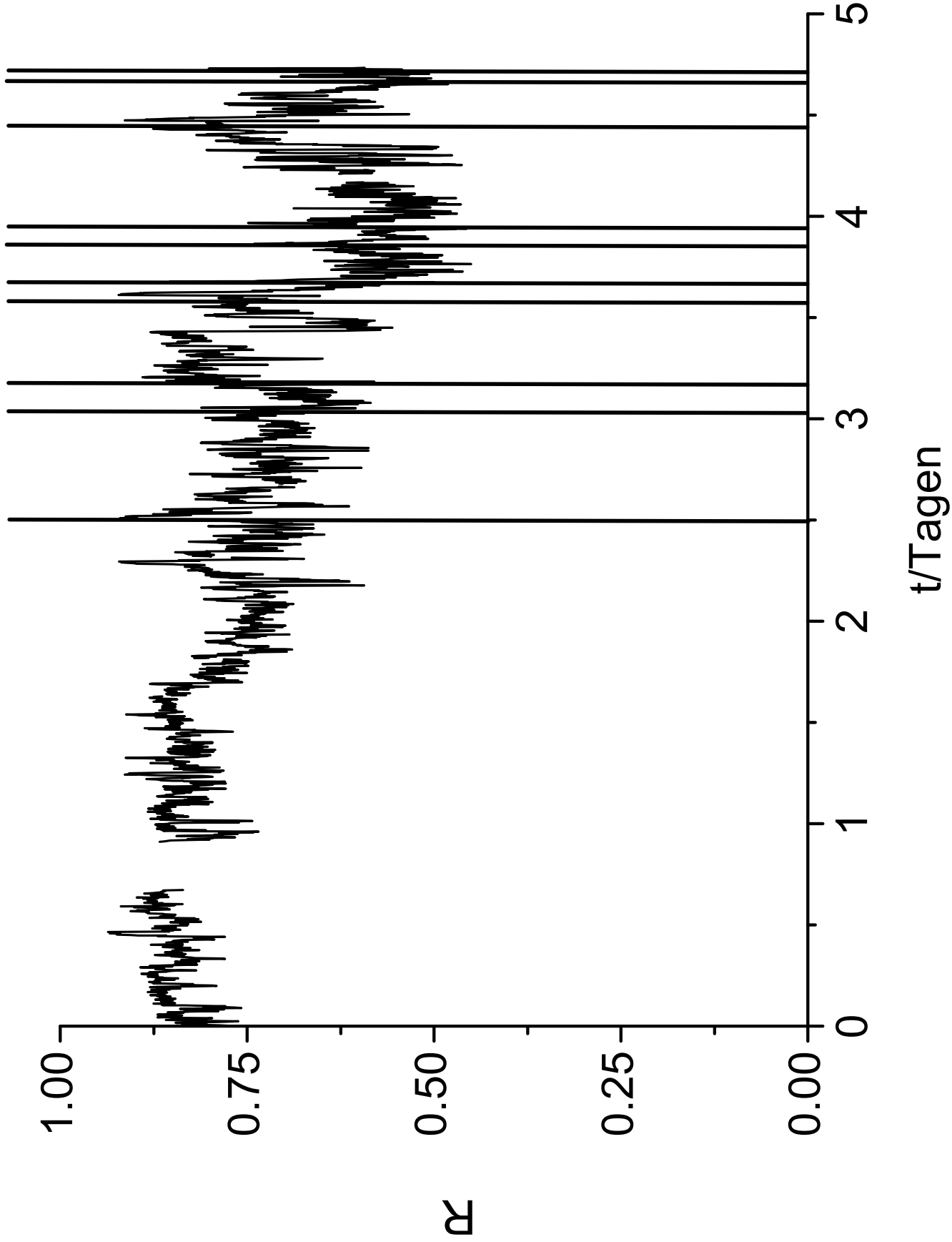


Abbildung 5.9: Zeitlicher Verlauf der berechneten Synchronisationswerte (R) der Testmenge TM1 für die Datenquelle LP unter Anwendung eines *10 Punkte-Rückwärts-Filters*. Die senkrechten Linien markieren jeweils den elektrischen Beginn eines Anfalls. Die leeren Bereiche markieren die Aufzeichnungslücken.

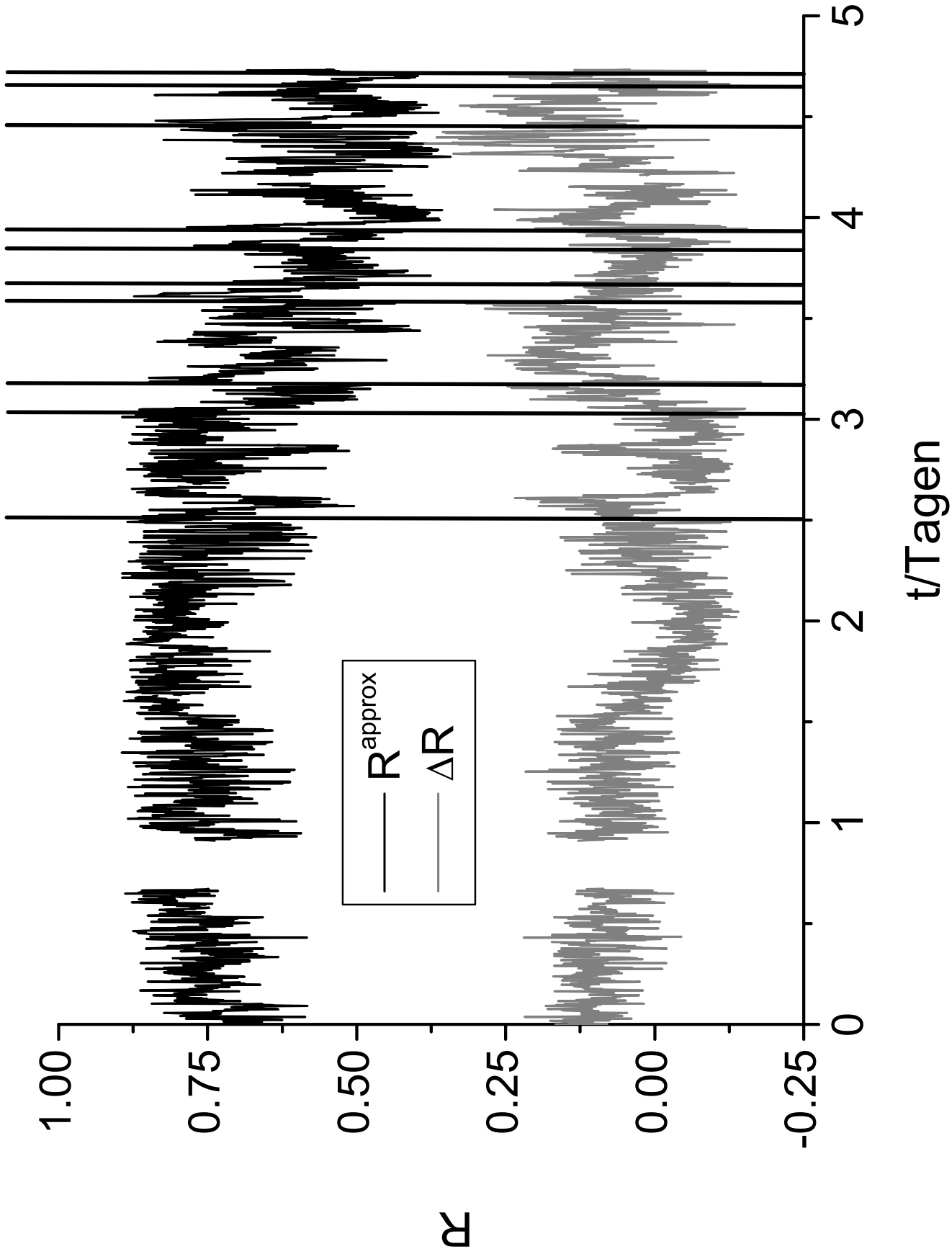


Abbildung 5.10: Zeitliche Verläufe der approximierten Synchronisationswerte (R^{approx}, schwarz) und der Differenzen (ΔR, grau) der berechneten und der approximierten Synchronisationswerte für die Testmenge TM1 der Datenquelle LP unter Anwendung eines *10 Punkte-Rückwärts-Filters*. Die senkrechten Linien markieren jeweils den elektrischen Beginn eines Anfalls. Die leeren Bereiche markieren die Aufzeichnungslücken.

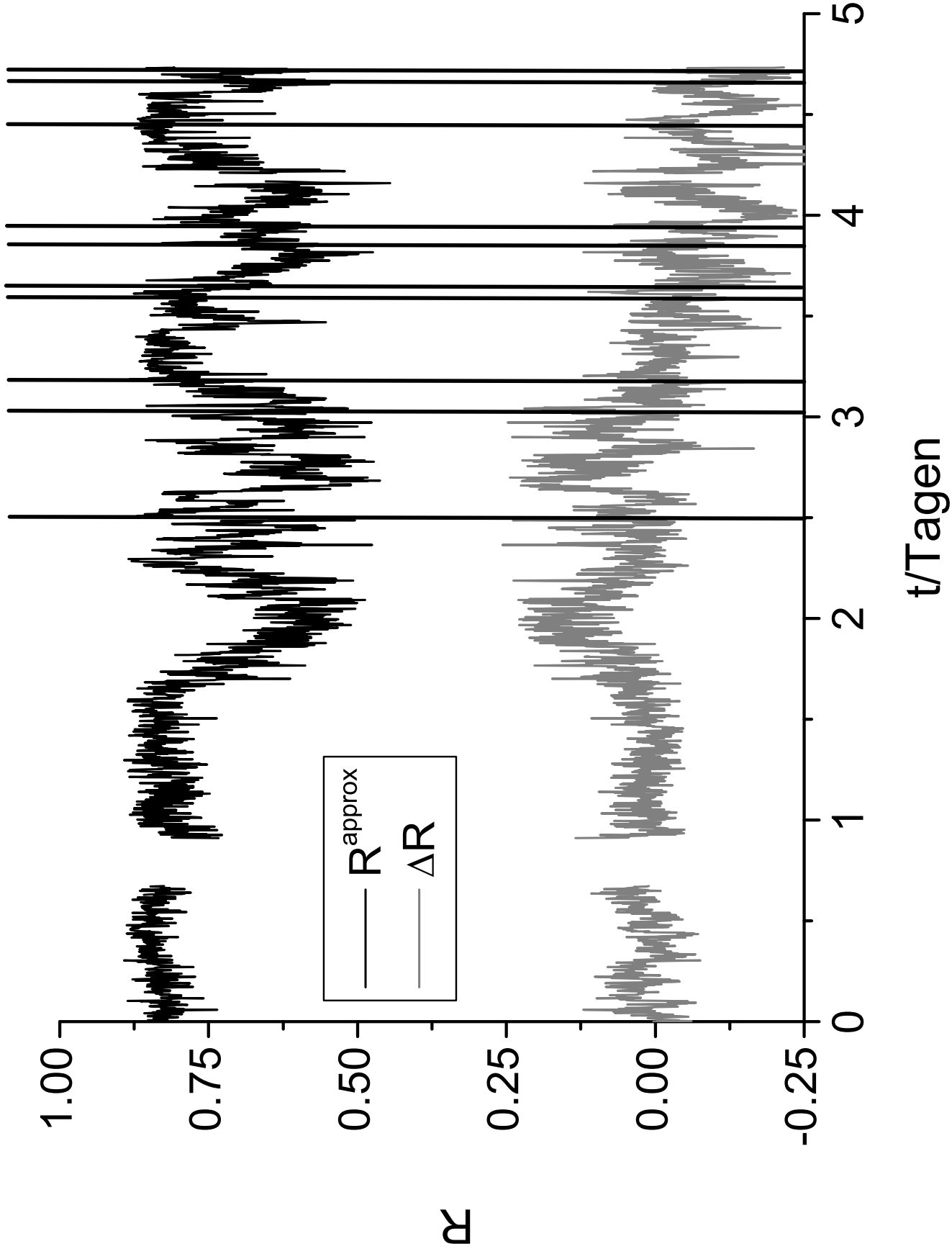


Abbildung 5.11: Wie in Abbildung 5.10, jedoch mit vertauschten Inhalten der Eingabe U und des Status $X(0)$.

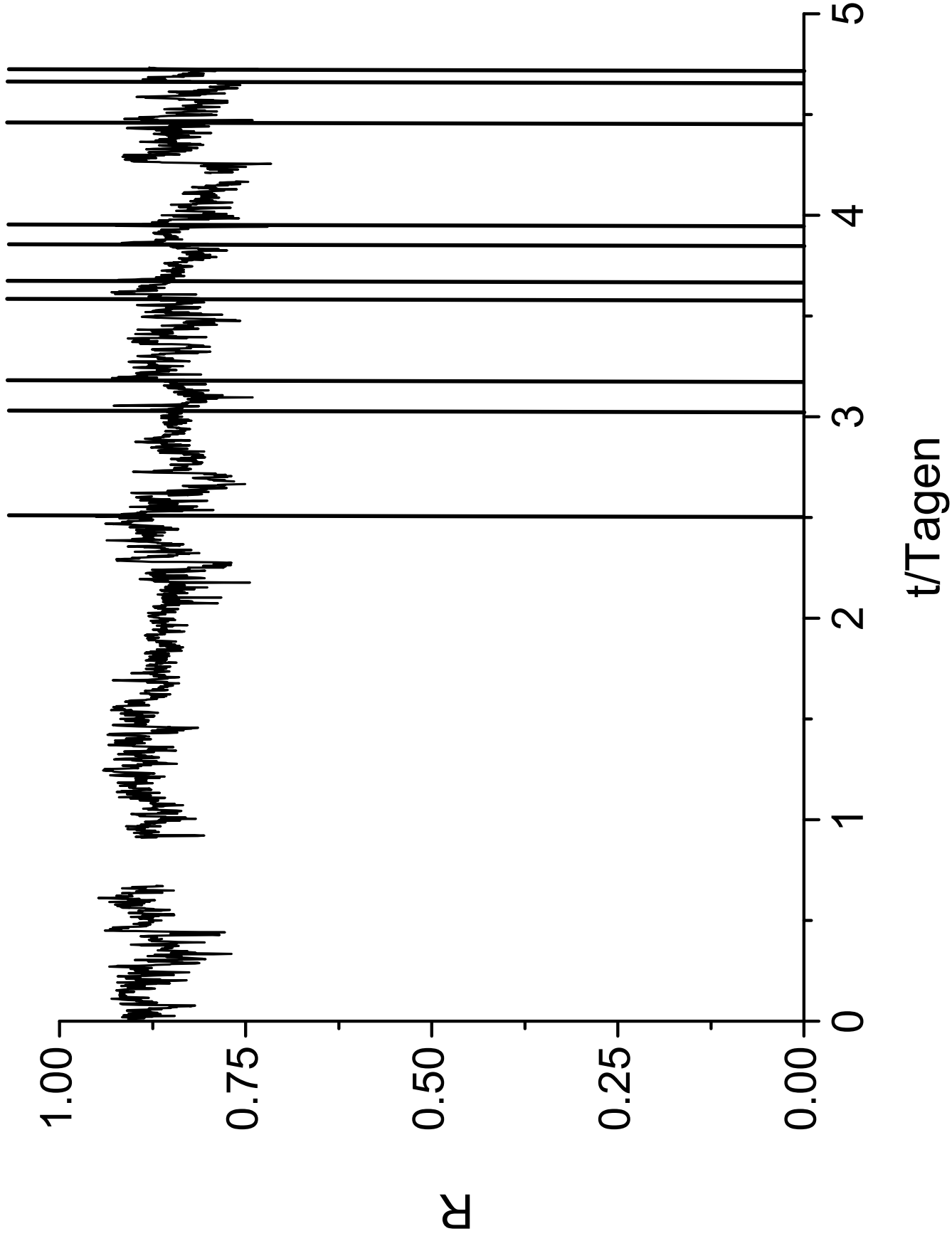


Abbildung 5.12: Wie in Abbildung 5.9, jedoch für die Testmenge TM2 der Datenquelle LP.

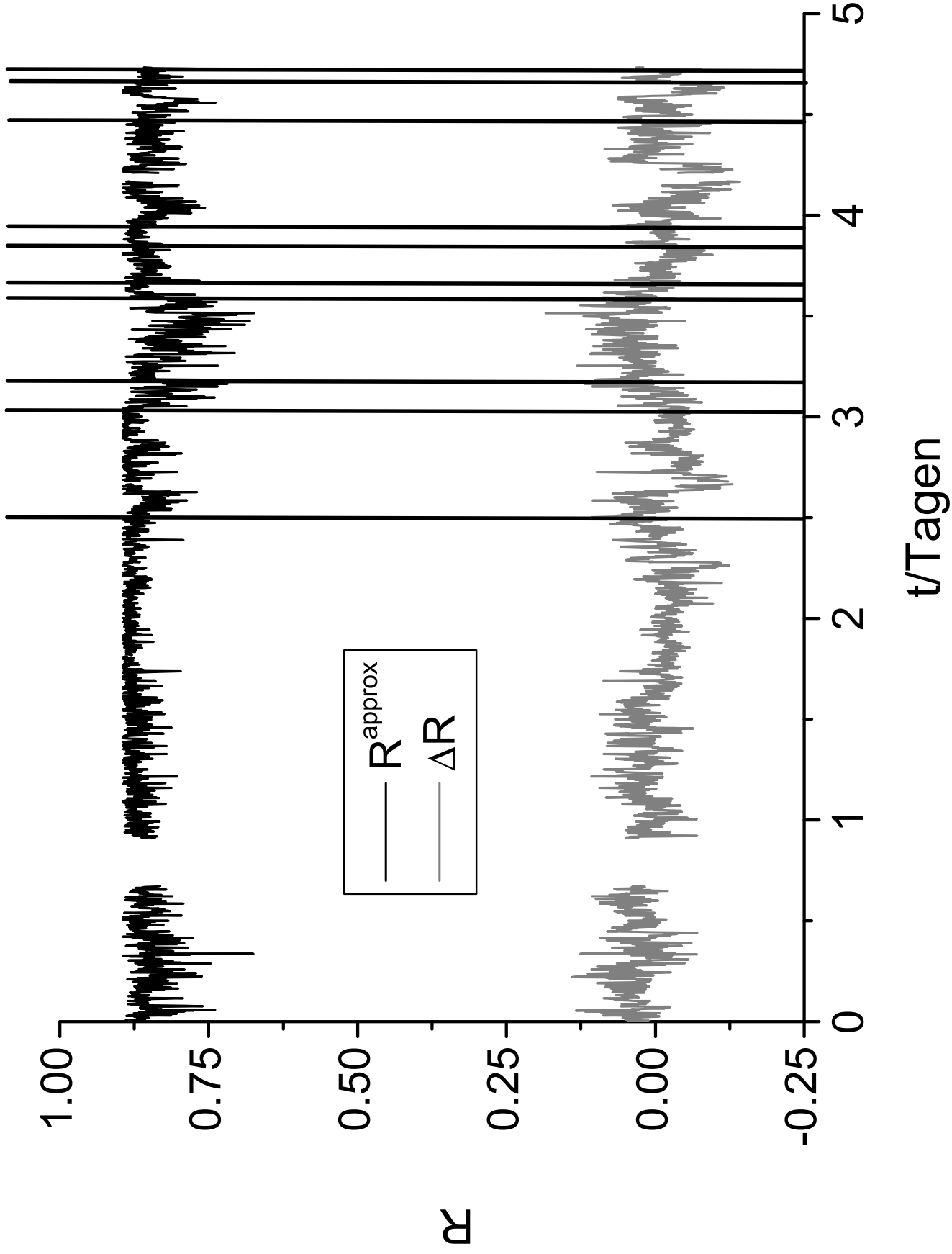


Abbildung 5.13: Wie in Abbildung 5.10, jedoch für die Testmenge TM2 der Datenquelle LP.

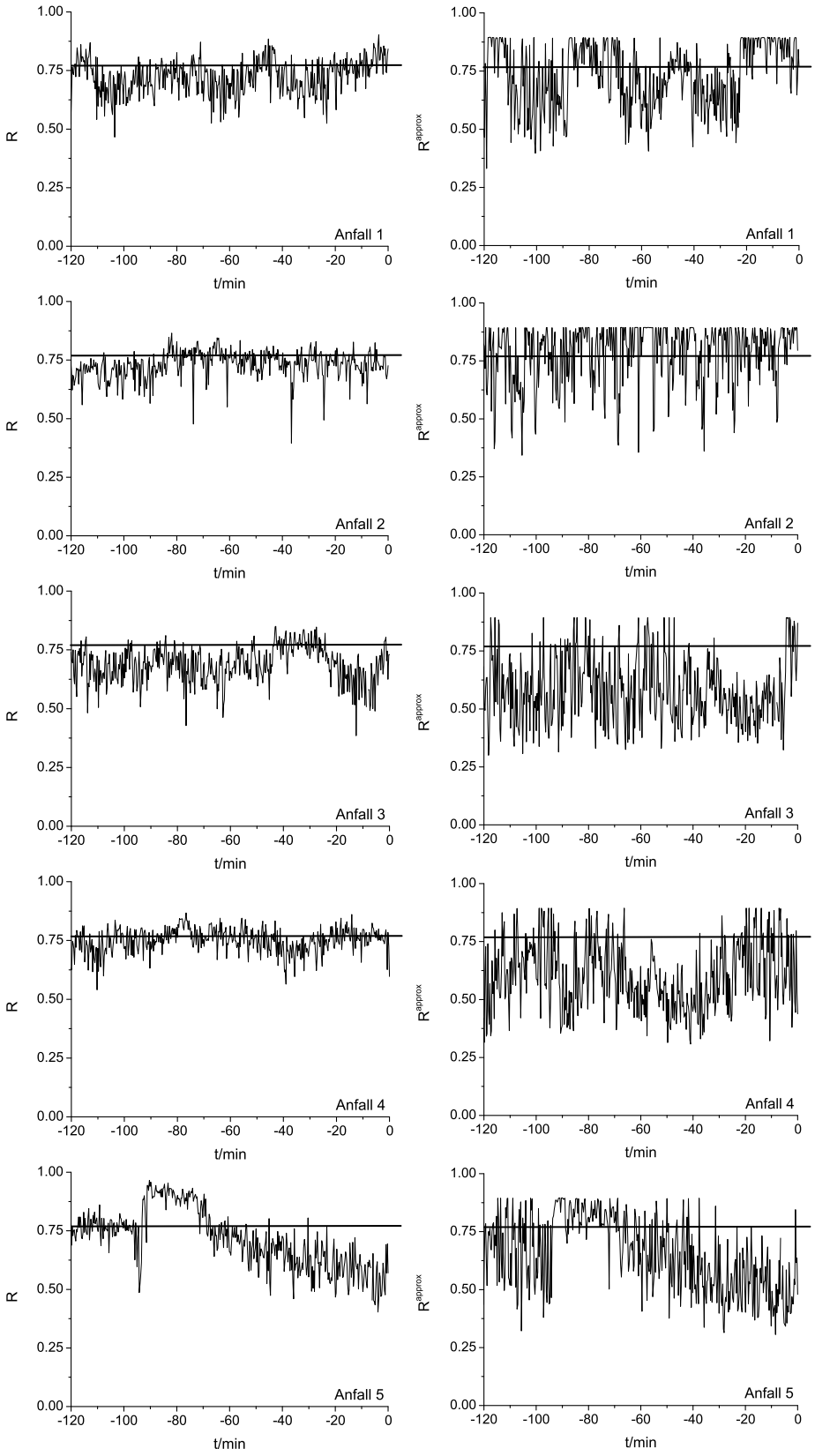


Abbildung 5.14: Zeitliche Verläufe der berechneten (R, links) und der approximierten Synchronisationswerte (R^{approx}, rechts) der Testmenge TM1, jeweils zwei Stunden vor den aufgezeichneten Anfällen 1 bis 5. Zeitpunkt 0 bezeichnet jeweils den elektrischen Anfallbeginn. Der durchschnittliche interiktale Synchronisationswert ($\overline{R}$) der Testmenge TM1 ist durch die waagerechten Linien angegeben.

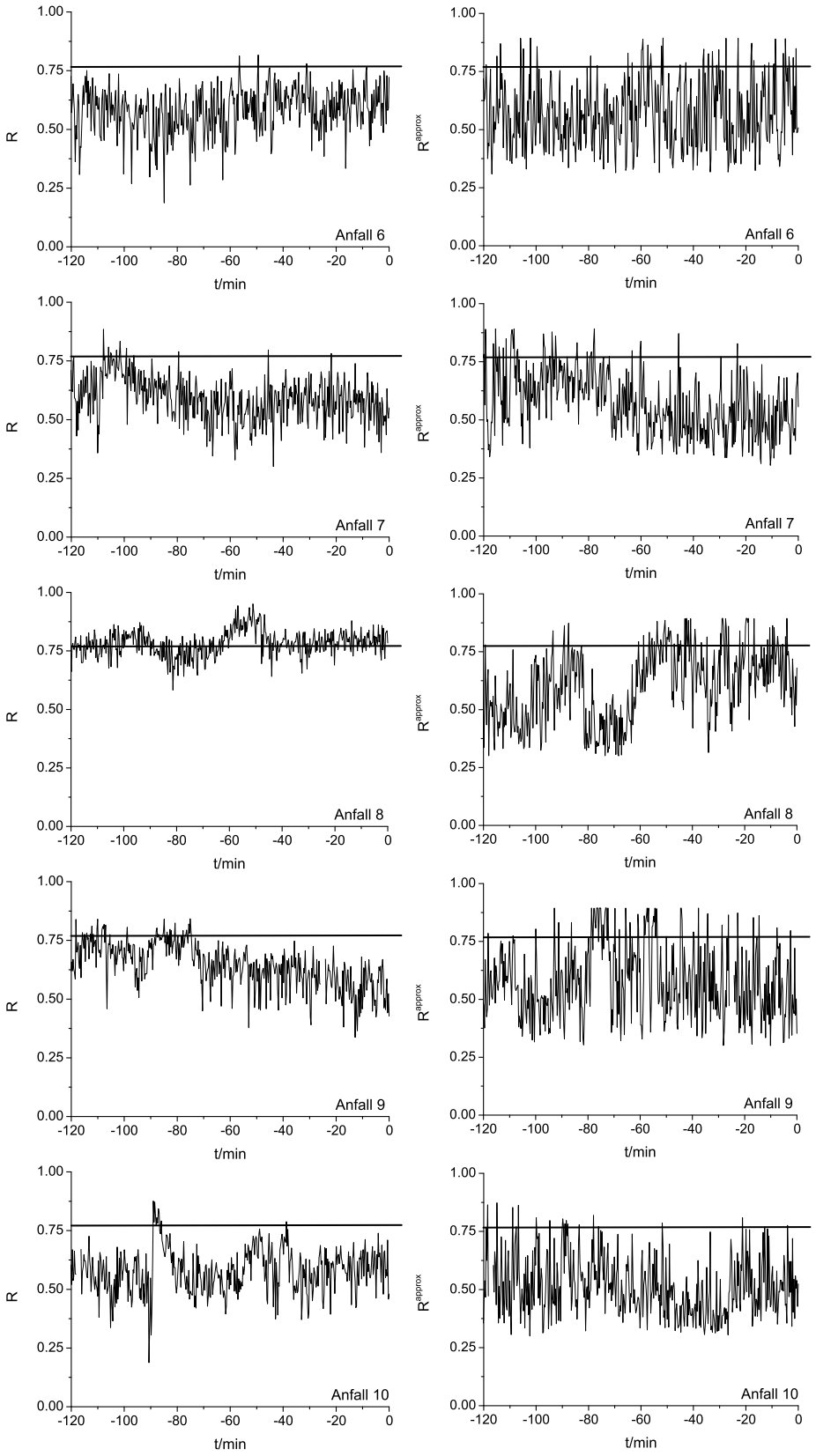


Abbildung 5.15: Wie in Abbildung 5.15, jedoch für die Anfälle 6 bis 10.

Kapitel 6

CNN-Algorithmus zur Datenverarbeitung und Zustandsdiskriminierung

In Kapitel 1.4.3 wurde die Möglichkeit erläutert, anhand der zeitlichen Entwicklung von charakterisierenden Kenngrößen in EEG-Daten von Epilepsiepatienten, einen prä-iktualen Zustand zu definieren. Die Ergebnisse der Kapitel 4 und 5 zeigen, dass mit CNN Synchronisationsmessungen durchgeführt werden können. Dabei wurden nicht nur synthetische Zeitreihen, sondern sogar EEG-Daten mit einer hohen Güte und Langzeitstabilität approximiert, womit auch eine Definition des genannten prä-iktalen Zustandes ermöglicht wird. Damit qualifiziert sich das CNN als ein *miniaturisierbares* Gerät, mit dem die Approximation von charakteristischen Kenngrößen durchgeführt werden kann. Was nun fehlt ist die Nachverarbeitung der approximierten Werte der Kenngrößen, damit dann im letzten Schritt eine Entscheidungsfindung (wurden Anzeichen für einen prä-iktalen Zustand gefunden oder nicht) durchgeführt werden kann.

Der im weiteren Text erläuterte CNN-Algorithmus ist ein Beispiel für eine vollständige Datenverarbeitung (die Analyse mit notwendiger Nachverarbeitung und Entscheidungsfindung), die mit *einem* CNN durchgeführt werden kann. Daraus könnte zukünftig ein miniaturisierbares oder sogar implantierbares EEG-Analyse-System entwickelt werden.

In der Abbildung 6.1 wird dieser Algorithmus mit sieben komplexen Arbeitsschritten und drei Schleifen, die ein mehrfaches Durchlaufen bestimmter Arbeitsschritte ermöglichen, dargestellt.

Im Arbeitsschritt (0) werden zunächst die gemessenen EEG-Daten in Zeitreihensegmente aufgeteilt, vorverarbeitet und in den CNN-Wertebreich $[-1, 1]$ konvertiert (siehe dazu auch Kapitel 4.1.1). Danach werden die Zeitreihensegmente in das CNN geladen.

Im Arbeitsschritt (1) erfolgt die Approximation der Kenngröße durch das CNN. Der Aufbau des CNN, seine Templates A und B und sein Schwellenwert Z wurden zuvor mit einem Parameter-Training ermittelt (siehe dazu Kapitel 4.3).

Im Arbeitsschritt (2) wird der Mittelwert über alle Zellwerte der Ausgabe $Y(\tau_{trans})$ berechnet. Hierfür wird zunächst die letzte Spalte Zellausgabe für Zellausgabe nach unten

verschoben und die Werte in einem mit 0 voreingestellten Register addiert. Danach werden die Zellausgaben aller Zeilen um eine Zelle nach rechts verschoben und die letzte Zeile wieder wie zuvor bearbeitet. Dies geschieht solange, bis alle Ausgabewerte $Y(\tau_{trans})$ aufaddiert sind. Der Inhalt des Registers wird durch die Anzahl der Zellen dividiert, um den Mittelwert zu ermitteln. Dieser stellt die approximierte Kenngröße dar.

Wird ein CNN mit ähnlichen Eigenschaften wie der ACE4K genutzt (siehe Kapitel 3.3), so besitzt es eine 64×64 Matrix und eine Auflösung von 7.6 Bit pro Zelle. Das Register, in dem der approximierte Synchronisationswert berechnet wird, muss also eine Breite von 20 Bit haben. Nur so kann der Sonderfall, dass alle Zellen der Ausgabe $Y(\tau_{trans})$ den Wert 1 haben (also haben sie Chipintern die Darstellung 255 bzw. 11111111 binär), abgespeichert werden. Die Division wird durch einen *shift* des Registers um 12 Bit nach rechts durchgeführt ($4096 = 2^{12}$), so dass der Mittelwert der Ausgabe $Y(\tau_{trans})$ als Resultat übrig bleibt.

Im Arbeitsschritt (3) wird nun diese approximierte Kenngröße in eine weitere CNN-Matrix (*Sicherungsmatrix*) gespeichert. Die zuvor gespeicherten Werte werden entsprechend der Abbildung 6.1 um eine Zellposition verschoben. Eine Sicherungsmatrix mit $64 \times 64 = 4096$ Zellen kann approximierte Kenngrößen über einen Zeitraum von maximal 20 Stunden enthalten (bei z.B. 200 Hz Abtastfrequenz, 4096 Elemente breiten Datenfenstern mit einer Überlagerung von 20%). Natürlich kann auch eine kleinere Matrix genutzt werden, falls weniger Synchronisationswerte aus der Vergangenheit benötigt werden. Sollen mehrere Kanalkombinationen approximiert werden, so kann die Arbeitsschrittfolge (0) bis (3) mehrfach wiederholt und weitere Sicherungsmatrizen, für jede Kanalkombination eine, erzeugt werden.

Im Arbeitsschritt (4) wird nun zur Weiterverarbeitung auf jede Sicherungsmatrix z.B. ein Schwellenwert-CNN angewendet (siehe Anhang C.1). Die Ausgabe $Y(\tau_{trans})$ enthält für jede Zelle, die die *Schwelle* überschritten hat, eine 1 (schwarz), sonst eine -1 (weiß). Das entspricht in der Darstellung eines ACE4K-CNN den Zellinhalten 255 bzw. 0. Diese Schwelle entspricht für das hier verwendete Synchronisationsmaß R der Schwelle, mit der zwischen interiktalen und prä-iktalen Zuständen unterschieden werden kann. Allerdings kann hier auch eine andere, der Kenngröße angepaßte Weiterverarbeitung der Sicherungsmatrix durchgeführt werden.

Im Arbeitsschritt (5) werden die schwarzen Pixel der zuvor gewonnenen Ausgabe $Y(\tau_{trans})$ gezählt. Dies erfolgt genauso wie im Arbeitsschritt (3). Allerdings erfolgt der *shift* des Registers um 8 Bit nach rechts, so dass im Register die Anzahl der schwarzen Zellen enthalten ist. Sollen z.B. mehrere Schwellen gesetzt bzw. Weiterverarbeitungsschritte durchgeführt werden, so können die Arbeitsschritte (4) und (5) mehrfach wiederholt werden. So entstehen für jede Sicherungsmatrix aus dem Arbeitsschritt (3) mehrere Register mit der Anzahl der schwarzen Zellen (abhängig von der Verarbeitungsart).

Die zeitliche Entwicklung von charakterisierenden Kenngrößen, wie z.B. der Synchronisationsmaßes R, beleuchtet nur ein Aspekt der untersuchten Dynamik. Durch Hinzunahme weiterer Kenngrößen könnten noch mehr Aspekte der Dynamiken untersucht werden, um z.B. im Fall von Epilepsie eine höhere Sensitivität und Spezifität bei der Suche nach dem hypothetischen prä-iktalen Zustand zu erreichen. Diese Möglichkeit der Wiederholung der Analyse mit verschiedenen Kenngrößen wird im Arbeitsschritt (6) dargestellt. Um mehrere Kenngößen mit diesem Algorithmus approximiern zu können, müssen die Arbeitsschritte

(1) bis (5) mehrfach wiederholt werden. So entstehen für jede Kenngröße ein oder mehrere Register mit den entsprechenden approximierten Werten. Das CNN bietet genügend Rechenkapazität für die Approximation mehrerer Kenngrößen, da nach jedem Eintreffen von Daten ein längerer Zeitraum übrig bleibt, in der das darauffolgende Fenster aufgezeichnet wird. Eine CNN-Operation benötigt eine Ausführungszeit von wenigen Milisekunden. Die EEG-Daten der im Kapitel 5 erwähnten Datenquelle LP wurden mit einer Abtastfrequenz von 200Hz aufgezeichnet. Wird z.B. eine Datenfensterbreite von 4096 Datenpunkten mit 20% Überlagerung angenommen, so bleibt dem CNN für die Arbeitsschritte (1) bis (6) 16.384 Sekunden Zeit.

Im Arbeitsschritt (7) wird aus den gewonnenen Werten in den Registern eine Entscheidung gefällt. Im Fall von EEG-Daten, die von Epilepsiepatienten stammen, kann z.B. entschieden werden ob ein hypothetischer prä-iktaler Zustand vorliegt oder nicht. Ein Anzeichen dafür wären z.B. im Fall der Synchronisationsmessung wenig gezählte schwarze Zellen in den Registern, also ein deutliches Unterschreiten der Schwelle.

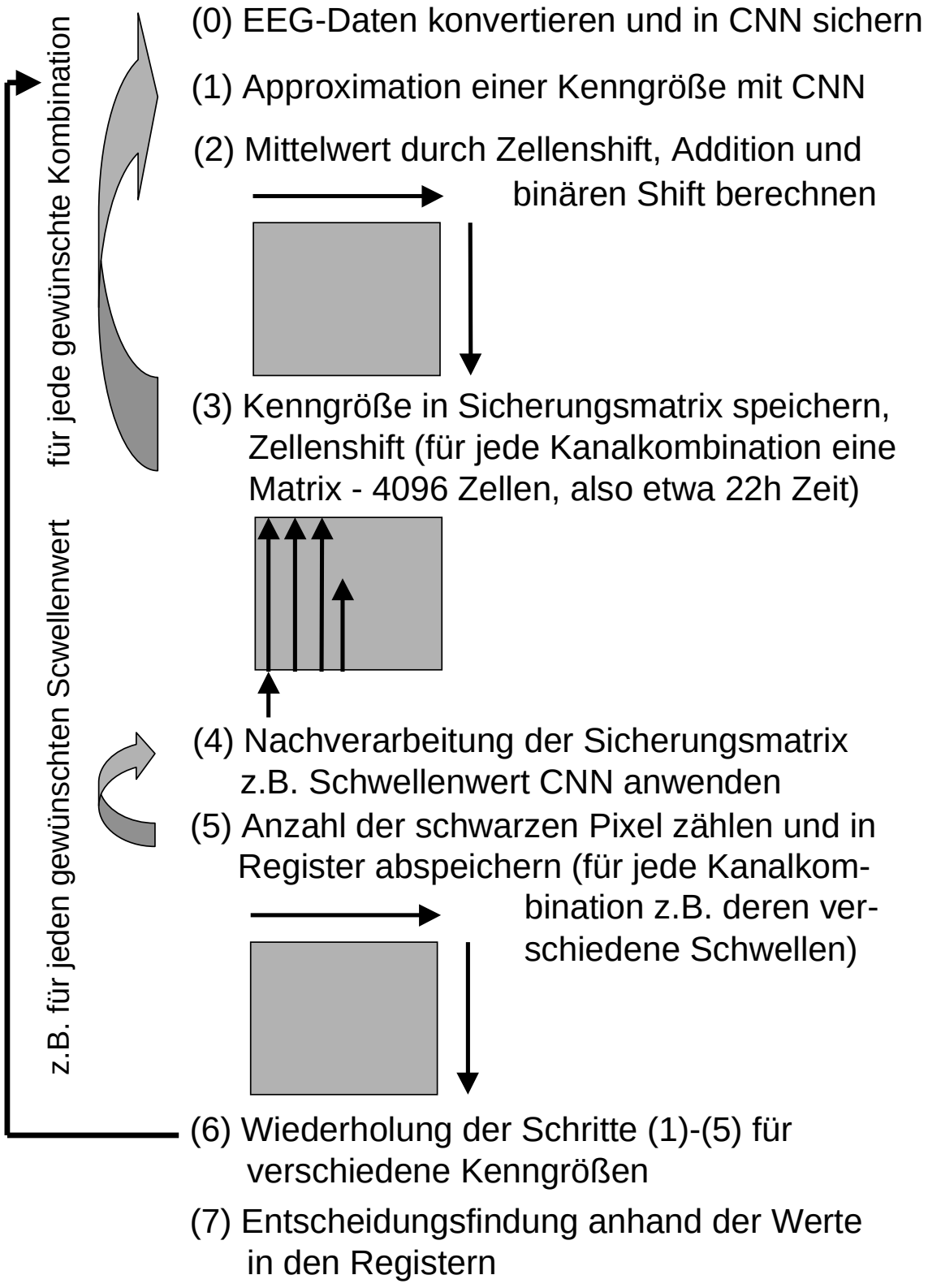


Abbildung 6.1: Beispielhafter CNN-Algorithmus zur Datenverarbeitung von EEG-Daten und Zustandsdiskriminierung

Kapitel 7

Diskussion

Technische neuronale Netze werden seit einigen Jahren als Instrumente zur Analyse von verschiedenartigsten Daten untersucht. In der vorliegenden Arbeit wurden zunächst synthetische Daten und, als mögliche Anwendung, Daten aus klinischen Applikationen mit CNN analysiert.

Die folgende Aufzählung soll einen Überblick über die Möglichkeiten geben, die technische neuronale Netze in klinischen Applikationen ermöglichen: In [LSI04], [CFCCCPS02], [LZY02] und [NG04] wurden Methoden beschrieben, mit denen große EEG-Datenmengen (Vielkanalableitungen, die mehrere Tage dauern und von mehreren Patienten stammen) automatisiert klassifiziert werden können, um damit Ärzten bei der Durchsicht eine Hilfestellung zu geben. Dabei wurden die Daten z.T. mit Filtern vorverarbeitet, um interessante Aspekte im Voraus zu extrahieren.

Eine wichtige Aufgabe der prächirurgischen Epilepsiediagnostik ist die Lokalisierung des hypothetischen Ursprungsortes der Anfälle im Gehirn. In [Ha02], [LLKLCLP00] wurde von technischen neuronalen Netzen berichtet, mit denen durch Analyse von Daten, die durch bildgebende Systeme gewonnen wurden, eine hohe Lokalisierungsqualität erreicht werden konnte.

Eine weitere wichtige Aufgabe ist die Definition eines bisher hypothetischen Voranfallszustands. Solch ein Zustand wurde in [PPHDW00] durch die Nutzung eines rekurrenten neuronalen Netzes definiert. In [SWKD01] und [SWKD02] hingegen wurde durch die Nutzung eines neuronalen Zellmodells eine Kenngröße entwickelt, die die Definition eines hypothetischen Voranfallszustands ermöglichte.

In den oben genannten Beispielen wurde eine direkte Klassifikation der Daten mit neuronalen Netzen angestrebt. Dabei wurden die Daten z.T. kompliziert vorverarbeitet. Die vorliegende Arbeit ging inhaltlich einen Schritt zurück. Es wurde keine direkte Klassifikation versucht, statt dessen sollte die Approximation von charakterisierenden Kenngrößen erfolgen. Diese Kenngrößen spiegeln verschiedene Aspekte der untersuchten Dynamiken wider. Sie können durch weitere Schritte in einem CNN-Algorithmus zusammengefaßt werden, wie in dieser Arbeit vorgeschlagen. Im letzten Schritt würde dann eine Klassifikation erfolgen. Dabei geschehen alle Arbeitsschritte innerhalb eines CNN, nur die Gewichte und der Schwellenwert müssen entsprechend der gewünschten Aufgabe rekonfiguriert werden. Mit dieser Vorgehensweise könnte also eine große Menge von Eigenschaften untersucht wer-

den, die *gemeinsam* in eine Klassifikation einfließen. CNN als technische Realisation bieten Höchstleistungs-Rechenkapazitäten und können diese Aufgaben in Echtzeit lösen.

Im Rahmen dieser Arbeit konnte die Abschätzung der bivariaten Kenngröße mittlere Phasenkohärenz R erfolgreich auf ein CNN übertragen werden. Die Skalierung der Amplitudenwerte wurde als einziger direkter Vorverarbeitungsschritt durchgeführt. Es konnte eine Methodik entwickelt werden, die zukünftig als Ansatz zur Übertragung weiterer bivariater und anderer Maße in CNN genutzt werden kann. Aufgrund der Äquivalenz von CNN zu *endlichen Automaten* existieren theoretisch keine Hürden bei der Übertragbarkeit der Algorithmen.

Das Phasensynchronisationsmaß R ist nicht das einzige, das sich zur Messung von Synchronisation in gekoppelten dynamischen Systemen eignet. Weitere interessante Ansätze wie z.B. zustandsraumbasierte Interdependenz-Maße ([AGLE99]) können im überwachten Training als Referenzwerte genutzt werden. Dabei erschließen sich neue Möglichkeiten, da durch das Einsortieren der Daten in die Zellmatrix eine Art *Zustandsraum-Einbettung* vorgenommen werden kann. Natürlich können auch völlig andere Ansätze der nichtlinearen Zeitreihen-Analyse in CNN umgesetzt wurden. Dabei kann die in dieser Arbeit entwickelte Methodik erweitert werden. Polynomielle Templates höherer Ordnung oder ganz andere Funktionen, die die Rolle der Gewichte übernehmen, können realisiert werden. Ein Beispiel dafür ist in [Ca03] beschrieben. Hier wurde durch die Nutzung von CNN ein nichtlineares erregbares Medium erzeugt, welches auf bestimmte Charakteristika von Anregungen mit einer raumzeitlichen Musterbildung instantan reagiert. Weitere Möglichkeiten erschließen sich in der Variation der Größe der Zell-Matrix des CNN, um z.B. die Einflüsse verschiedener Skalen innerhalb der Daten zu berücksichtigen. Das Hauptproblem in der hier entwickelten Methodik ist die erfolgreiche Durchführung der Parameter-Optimierung. Es gibt weitere interessante Ansätze, die nicht im Rahmen dieser Arbeit untersucht wurden. Beispiele dafür sind evolutionäre Ansätze (z.B. [KT00]), die z.B. die in der *Vergangenheit* gewonnenen besten Parameter mit den *zukünftigen* verschränken (Vermehrung, Mutation und Selektion), sowie konnektionistische Verfahren und Modelle, in denen neben den Parametern vor allem ihre Verknüpfungsstrukturen und Basisfunktionen untersucht werden (z.B. Bestimmung von Verknüpfungsstrukturen in neuronalen Netzen in [Ya99]). Allerdings wird die Komplexität dieser Suche massiv durch die zusätzliche Suche im Raum der Verknüpfungsstrukturen und Basisfunktionen erhöht.

Ein weiterer ausführlicher Vergleich mit anderen Studien ist nicht möglich, da bisher eine solche Arbeit nicht durchgeführt wurde. Allerdings wurde in [TKAW99] von einer weiteren, jedoch *univariaten* Kenngröße berichtet (der Korrelationsdimension D_2), deren Abschätzung erfolgreich auf ein CNN übertragen werden konnte.

Trotz der vielversprechenden Ergebnisse dieser Arbeit müssen noch weitere Untersuchungen folgen. Die Synchronisationsmessung mit CNN kontinuierlicher EEG-Aufzeichnungen (von mehreren Tagen Dauer) muss auf ein größeres Patientenkollektiv ausgedehnt werden, um weitere Aussagen über das Approximationsverhalten zu erhalten. Dabei ist die Untersuchung der Generalisierbarkeit der Synchronisationsmessung von besonderem Interesse. Durch einen anderen Aufbau der Trainingsmenge, in der Zeitreihenpaare aller verwendeten Datenquellen enthalten sind, kann möglicherweise *ein einzelnes* CNN entwickelt werden, mit dem in *allen* verwendeten Datenquellen die Synchronisation mit einer hohen

Approximationsqualität gemessen werden kann. Aufgrund der universellen Eigenschaften der CNN ist dieses Vorhaben theoretisch möglich. Unter Umständen müssen auch weitere Konfigurationen in der Template-Topologie und dem Polynomgrad der Templates gesetzt werden. Natürlich können auch andere Einstellungen im CNN verändert werden, allerdings sollte dabei stets die technische Realisierbarkeit gewährleistet werden.

Es existieren allerdings schon mehrere interessante schaltungstechnische Realisationen, wie der ACE4K (ein CNN mit einer 64×64 Zell-Matrix), der ACE8K (ein CNN wie der ACE4K, allerdings bestehend aus zwei miteinander verbundenen Zell-Matrizen), der ACE16K (ein CNN mit einer 128×128 Zell-Matrix) und weitere interessante Entwicklungen, wie z.B. das in [LPKH04] vorgestellte CNN mit 72×72 Zell-Matrix und kubischen Template-Polynomen. Diese Realisationen könnten eine direkte Übertragung der entwickelten CNN-Einstellungen erleichtern. Allerdings müssen dafür auch die unvermeidbaren Einflüsse der schaltungstechnischen Realisationen, wie z.B. die endliche Genauigkeit, intensiv in zukünftigen Studien untersucht werden.

Kapitel 8

Zusammenfassung

Die Analyse von Synchronisationsphänomenen, also Ähnlichkeiten und wechselseitigen Abhängigkeiten zwischen komplexen Dynamiken, spielt in vielen Wissenschaftsbereichen wie Physik, Ökologie, Biologie, aber auch in den Neurowissenschaften eine wichtige Rolle. Gegenstand dieser Arbeit war die Untersuchung von Synchronisationsphänomenen in dynamischen Systemen mit Hilfe von Zellularen Neuronalen Netzen (CNN).

Zunächst wurde der Begriff der Synchronisation in dynamischen Systemen vorgestellt. Dabei wurde auf die verschiedenen, in der Literatur genannten, Definitionen von Synchronisation eingegangen (siehe Kapitel 1.1). Im Anschluß wurde das Phasensynchronisationsmaß *mittlere Phasenkohärenz* R vorgestellt (siehe Kapitel 1.2). R ist ein nichtlineares Synchronisationsmaß, mit dem nicht nur in gekoppelten dynamischen Modellsystemen Synchronisation nachgewiesen werden kann, sondern es ist auch robust genug, um es auf gemessene Zeitreihen von Systemen mit weitestgehend unbekannter Dynamik anzuwenden. Eine wichtige Anwendung von R ist die Untersuchung von Aufzeichnungen der hirnelektrischen Aktivität im Rahmen der prächirurgischen Epilepsiediagnostik. Mormann und Kollegen ([MKADLE03], [MAKRDEL03]) berichteten von Abnahmen der Synchronisation vor epileptischen Anfällen. Damit ermöglicht dieses Maß, neben weiteren Synchronisationsmaßen, die nicht in dieser Arbeit aufgeführt wurden, einen hypothetischen Voranfallszustand zu definieren (siehe Kapitel 1.4.3).

Die Komplexität der Berechnung von nichtlinearen Kenngrößen wie z.B. R wird häufig durch polynomielle Terme bestimmt. Insbesondere wird diese Komplexität durch eine hohe Anzahl von Meßsonden drastisch erhöht (z.B. in klinischen Applikationen bis zu 256), so dass eine Berechnung mit heutigen Personal-Computern in Echtzeit kaum realisiert werden kann. Im Hinblick auf zukünftige, neuartige Analyse-Apparaturen sind aber auch weitere Faktoren wie niedriger Energieverbrauch, Miniaturisierbarkeit und Robustheit von Bedeutung.

Deshalb wurde im Folgenden die Architektur von technischen neuronalen Netzen (siehe Kapitel 2) und im speziellen von Zellularen Neuronalen Netzen (siehe Kapitel 3) vorgestellt. Technische neuronale Netze haben den Vorteil, dass bekannte Eigenschaften von natürlichen neuronalen Netzen, wie dem menschlichen Gehirn, integriert werden können. So können Lösungswege der Natur, wie z.B. das *Lernen*, in die technische Welt übertragen werden. Technische neuronale Netze ermöglichen eine völlig neue Herangehensweise an Probleme,

die mit herkömmlichen Computern ein polynomielles Laufzeitverhalten mit hohem Polynomgrad oder sogar ein exponentielles Laufzeitverhalten aufweisen. Durch ihr Paradigma der lokalen Kopplung entsteht ein beherrschbarer Verdrahtungsaufwand. Sie ermöglichen als *VLSI-Realisationen* (*Halbleiter-Bausteine*) Höchstleistungsberechnungen und verbrauchen dabei einen Bruchteil der Energie und des notwendigen Raumes, im Vergleich zu konventionellen Personal-Computern. Aufgrund der Äquivalenz zu *endlichen Automaten* (siehe Kapitel 3.1.2) kann theoretisch jeder auf einem Personal-Computer entwickelte Algorithmus in ein CNN bzw. einen CNN-Algorithmus übertragen werden.

Das zentrale Ziel dieser Arbeit war, die Messung von Synchronisationsphänomenen mit R auf ein CNN zu übertragen. Allerdings erfordert die *Programmierung* eines CNN ein Umdenken. Die Konfigurationen der wenigsten CNN, die für spezielle Probleme genutzt werden, lassen sich analytisch berechnen, statt dessen muss eine Parameter-Optimierung durchgeführt werden (siehe Kapitel 3.2.3 und Anhang B). Weiterhin wurde eine Methodik entwickelt, mit der das Problem in eine Darstellungsform gebracht wurde, die im CNN abbildbar ist (siehe Kapitel 4).

Es wurden zwei CNN-Werkzeuge genutzt. Das SCNN-System ist ein universelles Simulationssystem, mit einer Vielzahl von CNN-Konfigurationsmöglichkeiten. Es bietet die Möglichkeit der *überwachten* Parameter-Optimierung (siehe Kapitel 3.2.3). Das ACE4K-System ist eine schaltungstechnische Realisation und wurde als Koprozessor für Personal-Computer realisiert. Die im Rahmen dieser Arbeit erfolgte Anbindung an das SCNN-System ermöglichte eine einfache Nutzung des ACE4K. Allerdings wurde die Leistung durch Einschränkungen, vorgegeben durch die schaltungstechnischen Realisationen, beeinträchtigt.

Die Übertragung des Synchronisationsmaßes in CNN wurde zunächst anhand von synthetischen Zeitreihen durchgeführt. Die Generierung der synthetischen Zeitreihen ermöglichte die Variation der Kopplungsstärke und damit auch des Synchronisationsgrades. Als Ergebnis wurde eine hohe Approximationsqualität in der Messung des Synchronisationsgrades mit CNN erreicht (siehe Kapitel 4.4.1). Aufgrund dieses ermutigenden Ergebnisses wurden reale Daten, am Beispiel von EEG-Daten von drei Epilepsiepatienten, untersucht. Auch hier wurde eine eine hohe Approximationsqualität erreicht. In zwei von drei Fällen genügten die approximierten Werte, um einen hypothetischen Voranfalls-Zustand innerhalb der entsprechenden Datensätze zu definieren.

Somit konnte gezeigt werden, dass Synchronisationsmessungen mit CNN auch in realen Daten möglich sind. Allerdings konnte die Methodik nicht erfolgreich auf das ACE4K-System übertragen werden, da polynomielle Templates 2. und 3. Ordnung benötigt wurden (siehe Kapitel 4.5). Diese Vorgaben wurden bisher nicht von der schaltungstechnischen Realisation erfüllt. Aufgrund der universellen Eigenschaften des SCNN-Systems konnten unvermeidbare schaltungstechnische Toleranzen modelliert werden (siehe Kapitel 4.6). Es konnte gezeigt werden, dass die Approximationsqualität einer Synchronisationmessung mit CNN empfindlich auf Schwankungen in den Templates reagierte (aufgrund des hohen Polynom-Grades der gewonnenen CNN). Schwankungen im Schwellenwert wurde mit einer deutlich geringeren Empfindlichkeit begegnet.

Im weiteren wurden Untersuchungen zur Generalisierbarkeit der gewonnenen CNN durchgeführt (siehe Kapitel 5). Es konnten mehrere Arten von Generalisierungsverhalten nachgewiesen werden: Die zeitliche (Anwendung auf Datensätze von mehrtägiger Dauer) und

räumliche Stabilität (Anwendung auf Datensätze von anderen Meßsonden) der Approximation in einer Datenquelle, sowie die problemlose Übertragung auf eine weitere Datenquelle ohne zusätzliche Optimierungsmaßnahmen.

Aufgrund dieser ermutigenden Ergebnisse wurde ein exemplarischer CNN-Algorithmus entwickelt, mit dem die Messung von Kenngrößen, die Nachverarbeitung und die Entscheidungsfindung in und mit einem CNN durchgeführt werden können (siehe Kapitel 6).

Zusammenfassend kann geschlossen werden, dass das in dieser Arbeit vorgestellte Verfahren zur Übertragung einer Kenngröße in CNN Möglichkeiten eröffnet, dynamische Systeme auf eine neue Art und Weise zu untersuchen.

Danksagung

Bedanken möchte ich mich bei allen, die mich während des Studiums und der Dissertation unterstützt haben und somit zu derer Gelingen beigetragen haben.

Besonders danken möchte ich

- meinen Eltern, die mir das Studium und damit den Weg in die Promotion ermöglicht haben.
- meiner Freundin Sara Pfeiffer, die immer für alle meine Anliegen viel Zeit und Geduld hat.
- meiner Familie, insbesondere meiner Schwester Isabella Sowa, die mich immer unterstützt hat.
- Prof. Dr. P.David, Prof. Dr. C.E.Elger und PD Dr. Klaus Lehnertz, die mir die Dissertation ermöglicht haben.
- Prof. P.David, PD Dr. Klaus Lehnertz, Dr. Florian Mormann, Dipl.-Phys. Andy Müller, Dipl.-Phys. Hannes Osterhage und Dr. Christoph Rieke für die hervorragende Betreuung und Zusammenarbeit während der Dissertation!
- den NEUROPhysikern, für die Möglichkeit in einer hervorragenden Gruppe mitarbeiten zu dürfen!
- meinen Freunden (insbesondere Jochen Cammin), die immer für gute Abwechslung gesorgt haben.

Anhang A

Verwendete Integrationsalgorithmen

Die Dynamik eines physikalischen Systems wird durch seine *Bewegungsgleichung* festgelegt. Diese Bewegungsgleichung ist eine Differentialgleichung mit Variablen, die bestimmten Messgrößen des physikalischen Systems zugeordnet sind. Aber auch die Simulation eines CNN geschieht über ein System einfacher Differentialgleichungen. Diese spiegeln die *Entwicklung* des Status der einzelnen Zellen wider. Üblicherweise können solche Differentialgleichungen aufgrund ihrer Komplexität nicht analytisch gelöst werden. Daher werden numerische Verfahren angewendet, die im Folgenden erläutert werden. Diese numerischen Verfahren wurden zur Generierung des Rössler-Modellsystems und innerhalb der SCNN Software genutzt.

A.1 Euler-Verfahren

Gegeben sei eine Differentialgleichung der Form

$$\frac{d}{dt}x(t) = f(x(t)) \tag{A.1}$$

mit dem Anfangswert $x(0) = x_0$ und der Ableitung von $x(t)$ an der Stelle t_0

$$\frac{d}{dt}x(t)|_{t=t_0} = \lim_{h\to 0}\frac{x(t_0+h) - x(t_0)}{h} \tag{A.2}$$

Aus den beiden oben genannten Formeln kann das Eulersche Integrationsverfahren in der folgenden Form hergeleitet werden

$$x_{n+1} = x_n + f(x_n)h \tag{A.3}$$

wobei h den zeitlichen Abstand zwischen zwei Werten, also den Kehrwert der endlichen Abtastrate, bezeichnet.

Würde die Abtastrate unendlich groß (Grenzübergang $h \to 0$), so würde dieses Verfahren die Differentialgleichung A.1 exakt lösen. Computer können nur eine bestimmte Genauigkeit erreichen, dementsprechend ist h begrenzt. Zusätzlich wird bei jedem Integrationsschritt ein systematischer Fehler begangen, da bei der endlichen Integration über dem Intervall h nur die Ableitung am Beginn des Intervalls in die Formel eingeht. Damit tritt also ein Fehler der Größenordnung $O(h^2)$ auf.

A.2 Runge-Kutta Verfahren

Wird nun im A.1 genannten Verfahren ein Zwischenpunkt im Intervall h mit berücksichtigt, so verändert sich die Berechnungsvorschrift folgendermaßen

$$x_{n+1} = x_n + f(x_z)h \tag{A.4}$$

$$\text{mit} \quad x_z = x_n + \frac{1}{2}f(x_n)h \tag{A.5}$$

Dieses Verfahren wird Integrationsverfahren nach Runge-Kutta zweiter Ordnung genannt. Der zusätzliche Schritt verdoppelt bei gleichbleibendem h die Rechenzeit. Jedoch wird die Genauigkeit ebenso erhöht. Der Fehler in Runge-Kutta zweiter Ordnung bewegt sich in der Größenordnung $O(h^3)$. Werden nun weitere Zwischenpunkte im Intervall h berücksichtigt, so kann die Berechnungsvorschrift des Integrationsverfahren Runge-Kutta 4ter Ordnung entwickelt werden. Für diese gilt:

$$x_{n+1} = x_n + \left[\frac{1}{6}f(x_{z1}) + \frac{1}{3}f(x_{z2}) + \frac{1}{3}f(x_{z3}) + \frac{1}{6}f(x_{z4})\right]h \tag{A.6}$$

$$\text{mit} \quad z_{z1} = x_n \tag{A.7}$$

$$x_{z2} = x_n + \frac{1}{2}f(x_{z1})h \tag{A.8}$$

$$x_{z3} = x_n + \frac{1}{2}f(x_{z2})h \tag{A.9}$$

$$x_{z4} = x_n + f(x_{z3})h \tag{A.10}$$

Dieses Verfahren vervierfacht, bei gleich bleibendem h, die Rechenzeit im Vergleich zum Euler-Verfahren. Gleichzeitig wird aber die Genauigkeit stark erhöht, da sich der Fehler nur noch in der Größenordnung $O(h^5)$ bewegt.

Zusammenfassend läßt sich sagen: Das Eulersche Integrationsverfahren kann mit einem entsprechend gewählten h für eine schnelle Integration genutzt werden. Das Runge-Kutta Verfahren 4ter Ordnung hat einen Berechnungsfehler, der um drei Größenordnungen kleiner ist als der des Eulerschen Integrationsverfahrens. Allerdings erfolgen für jedes Intervall h vier Berechnungsschritte. Bei gleicher Wahl von h ist also das Runge-Kutta Verfahren viel genauer als das Euler Verfahren, jedoch auch viermal langsamer.

Die Wahl des Integrationsverfahrens, der Schrittweite h und der Integrationsdauer $t_d = Nh$ (mit N als Anzahl der Integrationsschritte) ist damit stark von dem gestellten Problem abhängig. Eine ungünstige Wahl kann Ergebnisse verfälschen und z.B. nicht existente Lösungen produzieren ([Mor98]) oder die Integrationszeit unnötig erhöhrn und damit die Berechnungszeit drastisch verlängern.

Weitere Informationen zu den oben genannten Verfahren finden sich in [PTVF92].

Anhang B

Verwendete Optimierungsalgorithmen

Optimierungsalgorithmen werden eingesetzt, um globale Minima oder Maxima einer nicht analytisch lösbaren Funktion zu finden. Mit diesen Algorithmen wird versucht, das *Optimierungsproblem* zu lösen. Im Fall von neuronalen Netzen wird mit diesen Algorithmen versucht, die Gewichte so einzustellen, dass das neuronale Netz seine Aufgaben möglichst optimal erfüllt.

Der folgende Ablauf ist eine allgemeine Skizze eines Optimierungsverfahrens: Zunächst werden eine Trainingsmenge und eine Ergebnismenge definiert. Durch Anwendung eines Optimierungsverfahrens sollen die Parameter eines neuronalen Netzes so eingestellt werden, dass die Ausgaben des neuronalen Netzes, auf die Trainingsmenge angewendet, möglichst genau der Ergebnismenge entsprechen. Eine Trainingsmenge TM kann folgendermaßen aussehen

$$TM = ((EW_i, SW_i), GW) \tag{B.1}$$
$$\text{mit Paaren } EW_i = \text{Werte, die der Eingang annimmt} \tag{B.2}$$
$$\text{und } SW_i = \text{Werte, die der Status annimmt} \tag{B.3}$$
$$\text{und mit } GW = \text{Werte, die die Gewichte zu Beginn annehmen} \tag{B.4}$$
$$\tag{B.5}$$

Zusätzlich existiert eine Ergebnismenge

$$RM = (RW_i) \tag{B.6}$$

mit Ergebnissen RW_i, die jeweils dem gewünschten Ergebnis des Paares (EW_i, SW_i) entsprechen. Natürlich muss sowohl TM als auch RM der Topologie des gewählten neuronalen Netzes entsprechen, d.h. die Wertemengen müssen im Neuronalen Netz abbildbar sein.

Diese Trainingsmenge beschreibt ein Problem, das mit einem neuronalen Netz (NN) gelöst werden soll. Gilt nun

$$NN((EW_i, SW_i), GW) = NN(TM_i) = AW_i \tag{B.7}$$

d.h. das neuronale Netz berechnet als Ausgabe die Werte AW, so kann mit der Ausgabemenge

$$AM = (AW_i) \tag{B.8}$$

eine Bewertungsfunktion entwickelt werden. Diese hat die folgenden Haupteigenschaften

$$FCompare(AW_i, RW_i) = \begin{cases} 0 & : \quad AW_i = RW_i \\ > 0 & : \quad \text{sonst} \end{cases} \tag{B.9}$$

Das Trainings- bzw. Optimierungsverfahren wird mit einer bestimmten Anzahl von Schritten angewendet. Dabei wird beim ersten Schritt die Anfangskonfiguration TM genutzt und mit $FCompare$ eine Bewertung berechnet. In den folgenden Schritten versucht nun das Verfahren durch Variation der Werte von GW, der Gewichte, das Minimum von $FCompare$ zu finden. Ist dieses Minimum gefunden, dann gilt die Problemstellung, soweit sie in der Trainingsmenge abgebildet wurde, als gelöst. Die Einschränkung bezieht sich darauf, ob das neuronale Netz nur noch die Trainingsmenge abbilden kann, also *übertrainiert* wurde, oder ob es auch weitere Tripel (EW_i, SW_i, RW_i) korrekt berechnet. Dieses spezielle Problem wird in den Kapiteln 2.1.4 und 3.2.3 besprochen. Dabei ist

$$CM = (EW_i, SW_i, RW_i) \quad \text{mit} \quad (EW_i, SW_i) \notin TM \tag{B.10}$$

die Testmenge für das Problem.

Die folgenden Abschnitte erläutern die in dieser Dissertation genutzten Optimierungsalgorithmen, die im Training in SCNN genutzt wurden. Diese Algorithmen ermöglichen die Extremwertsuche in N-dimensionalen Räumen. In dem oben genannten Beispiel wurden die Gewichte optimiert. Natürlich können weitere Parameter eines künstlichen neuronalen Netzes optimiert werden, wie z.B. die Topologie oder die Ausgabefunktion.

B.1 Iterative-Annealing Optimierungsverfahren

B.1.1 Algorithmusskizze

Iterative-Annealing basiert auf der Idee eines sich immer wiederholenden *Abkühlvorgangs*. Der Parameterraum dieses Verfahrens ist genauso groß wie die Anzahl der zu optimierenden Parameter in der Problemstellung.

Zunächst wird der Parametervektor mit Schätzern für einen Startwert gefüllt (im einfachsten Fall Zufallszahlen). Dieser Vektor wird noch einmal verändert, indem jeder Komponente Zufallszahlen hinzu addiert werden. Diese Werte stammen jetzt aus einem Wertebereich, der von der Starttemperatur T_0 abhängig ist. Falls jetzt der Funktionswert an dem neuen Parametervektor kleiner ist als zuvor (Bewegung hin zum möglichen globalen Minimum), wird dieser Parametervektor gespeichert. Jetzt werden die Komponenten des Parametervektors erneut durch Addition von Zufallszahlen verändert. Diese stammen allerdings aus einem kleineren Wertebereich, also mit einer verminderten Temperatur $T < T_0$. Wieder wird der Funktionswert verglichen. Dieser Vorgang wiederholt sich so lange, bis

eine bestimmte Anzahl von Durchläufen erreicht wird. Der Parametervektor befindet sich in einem möglicherweise lokalen Minimum. D.h. der Funktionswert des Parametervektors hat den kleinsten möglichen Wert erreicht, der in diesem Abkühlvorgang möglich war.

Um sicher zu stellen, dass nicht ein lokales Minimum erreicht wurde, wird diese Prozedur mit dem bisher besten Parametervektor und der Starttemperatur T_0 mehrfach wiederholt (siehe auch Abbildung B.1). Der folgende Ablauf illustriert den Algorithmus im Detail:

1. Initialisierung
   - D als Dimension des Fehlerraums
   - Startwerte $x_0^k \quad \forall \quad k \in [1, D]$ wählen
   - $k = 0$
   - maximale Schrittanzahl N_{train} setzen
   - Anzahl der Wiederholungen j_{max} setzen
   - Starttemperatur T_0 setzen
   - minimale Temperatur τ setzen
2. Schrittweite $\nu = (\frac{\tau}{T_0})^{\frac{j_{max}}{N_{train}}}$ setzen
3. $T = T_0$ und $i = 0$ setzen
4. $y_i^k = x_i^k + u^k T \quad \forall \quad k \in [1, D]$ und $u^k \in U[-0.5, 0.5]$ (gleichverteilt)
5. wenn $f(\vec{y}_i) < f(\vec{x}_i)$ setze $\vec{x}_{i+1} = \vec{y}_i$
6. verringere Temperatur $T = \nu T$
7. $i = i + 1$
8. wenn $i < \frac{N_{train}}{j_{max}}$, gehe zu Schritt 4.
9. $j = j + 1$
10. wenn $j < j_{max}$, gehe zu Schritt 3.

Dieser Ablauf verdeutlicht, dass N_{train}, j_{max} und T_0 wichtige Systemparameter sind und sorgfältig gewählt werden müssen, um sinnvolle Ergebnisse zu erhalten.

Weitere Informationen zu diesem Thema sind in [FTIA01], [FTFE01] und [Sch02] zu finden.

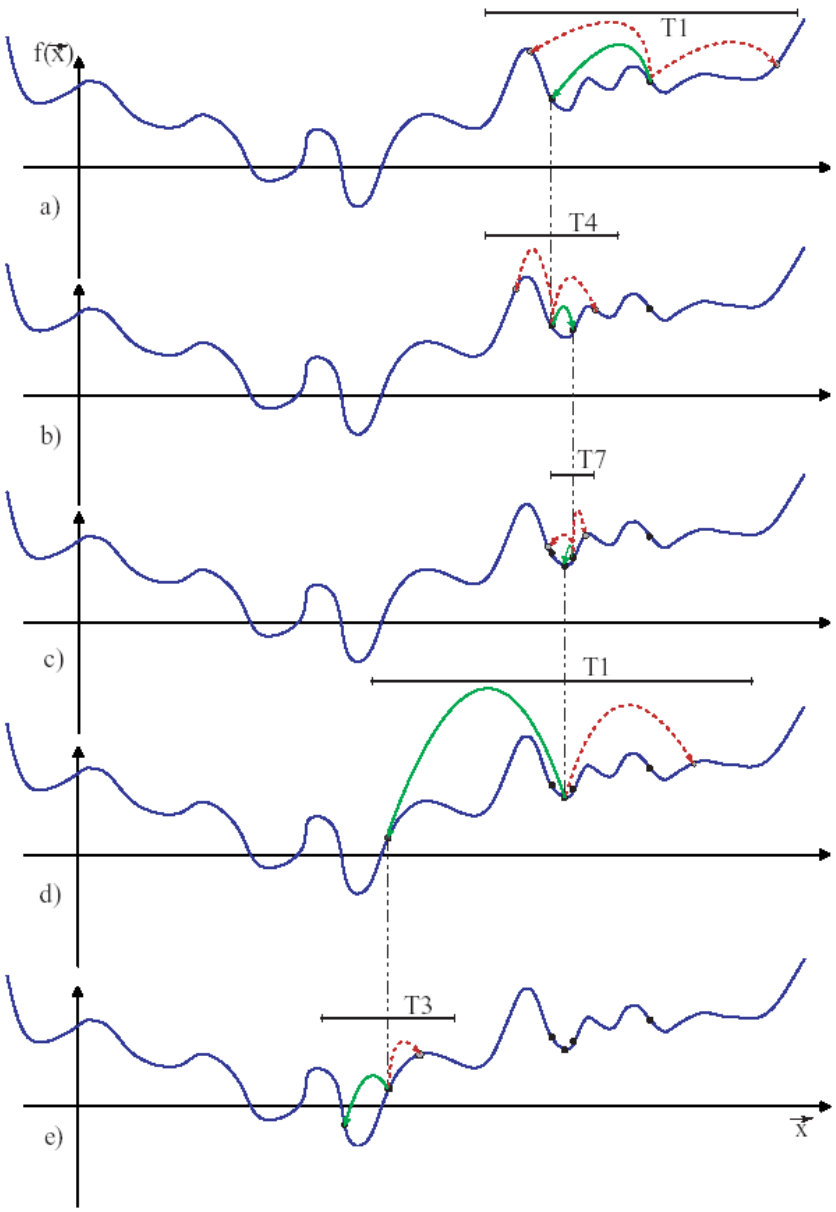


Abbildung B.1: Die Graphen a.) bis e.) zeigen den Verlauf der Suche nach dem globalen Minimum der Funktion $f(x)$ ([Sch02])

B.2 Powell's Linienminimierungsverfahren

B.2.1 Algorithmusskizze

Die folgende Algorithmusskizze illustriert das Vorgehen:

- die Vektoren P und n sind die gegebenen Eingabewerte, f die zu minimierende Funktion
- Finden des Skalars λ, das $f(P + \lambda * n)$ minimiert
- Ersetzen von P mit $P + \lambda * n$ und n mit $\lambda * n$
- die Minimierung ist abgeschlossen

Dieses Verfahren basiert auf der Vereinfachung eines multidimensionalen in ein eindimensionales Problem. D.h. in der verbleibenden, ausgewählten Richtung kann nun ein Minimierungsverfahren angewendet werden, welches nur bei eindimensionalen Funktionen funktioniert.

So kann das Problem in zwei Unterprobleme zerlegt werden:

- Richtungsfindung
- Minimierung einer eindimensionalen Funktion

Weitere Informationen zu diesem Thema sind in [PTVF92] zu finden.

B.2.2 Richtungsfindung

Die hier verwendete Richtungsfindung geschieht durch die Methode des *Nicht-Nutzens der Richtung mit dem stärksten Gradienten.*

Die Idee in dieser Methode basiert auf der *quadratisch konvergierenden* Methode. Hier werden die Richtungen n_i mit den Basis-Vektoren initialisiert:

$$n_i = e_i \quad \text{mit} \quad i = 1, ..., N \tag{B.11}$$

Der folgende Ablauf wird so lange durchgeführt, bis kein Abstieg mehr geschieht:

- setzen des Startpunktes auf P_o (Schätzer für einen Startpunkt oder z.B. Zufallszahlen)
- für alle $i = 1, ..., N$: P_{i-1} in Richtung n_i zum Minimum bewegen und den Punkt als P_i setzen
- für alle $i = 1, ..., N - 1$: $n_i \leftarrow n_{i+1}$ setzen
- $n_N \leftarrow P_N - P_0$ setzen
- P_N in Richtung n_N zum Minimum bewegen und den Punkt als P_N setzen

Dieser Ablauf enthält jedoch ein Schwäche. Nach jedem Durchlauf wird n_1 überschrieben. Dies kann Richtungen schaffen, die aufeinander gefaltet sind, also lineare Abhängigkeiten erzeugen. Dementsprechend kann der Optimierungsalgorithmus falsche Ergebnisse liefern, da u.U. nur ein Unterraum nach den Minimum untersucht wird. Diese Schwäche kann z.B. durch Reinitialisieren der n_i durch e_i nach z.B. N Schritten verhindert werden.

In diesem Fall wird jedoch die oben skizzierte Richtungsfindung abgeändert. Die Hauptidee ist, dass die Richtung in der f den stärksten Abstieg erfährt, ausgeschlossen wird. Dadurch, dass es der stärkste Abstieg ist, wird es auch als Hauptkomponente in der neuen Richtung sein, die hinzugefügt wird. Der Ausschluss des stärksten Abstiegs bewahrt vor einem Aufbau linearer Abhängigkeiten.

B.2.3 Minimierung einer eindimensionalen Funktion

Die folgenden beiden Möglichkeiten einer Minimierung stehen in *SCNN* zur Verfügung. In den hier durchgeführten Trainingsszenarien wurde nur die erste Möglichkeit genutzt. Der Vollständigkeit halber werden aber beide erwähnt.

Die erste Möglichkeit, eine eindimensionale Funktion zu minimieren besteht darin, die Methode der *parabolischen Interpolation* und *Brent's Methode in einer Dimension* zu nutzen. Die hier genutzte Technik wird *inverse parabolische Interpolation* genannt. Gegeben sei die eindimensionale Funktion f und die Werte a, b und c auf der Abszisse. Um nun den Wert x zu bestimmen, welcher das Minimum der Parabel durch $f(a)$, $f(b)$ und $f(c)$ auf der Abszisse repräsentiert, genügt es die folgende Formel zu nutzen:

$$x = b - \frac{1}{2}\frac{(b-a)^2[f(b)-f(c)] - (b-c)^2[f(b)-f(a)]}{(b-a)[f(b)-f(c)] - (b-c)[f(b)-f(a)]} \tag{B.12}$$

Diese Formel lässt sich einfach herleiten. Die Abbildung B.2 illustriert das Vorgehen.

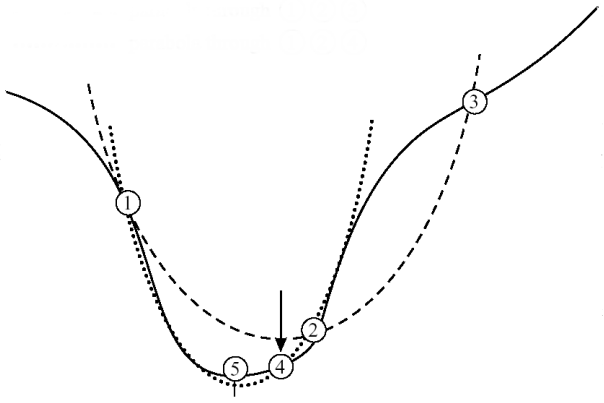


Abbildung B.2: Die Punkte 1, 2 und 3 stellen die erste Parabel dar, die an $f(x)$ angelegt wurde. Die Parabel durch die Punkte 1, 2 und 4 geht aus dem vorher gewonnenen Punkt 4 (vorläufiges Minimum) hervor. Punkt 5 ist das nächste vorläufige Minimum ([PTVF92]).

Dieses Verfahren sucht nun iterativ nach dem Minimum, bezieht dabei aber nur die Werte und die Funktionswerte ein.

Als zweite Möglichkeit, eine eindimensionale Funktion zu minimieren, kann die Methode der *eindimensionalen Suche mit ersten Ableitungen* genutzt werden. In diesem Fall werden

die Ableitungen zum Finden des Minimums genutzt. Gegeben sei wieder die eindimensionale Funktion f und die Werte a, b und c auf der Abszisse. Der Wert x wird folgendermaßen bestimmt: Das Vorzeichen der Ableitung des mittleren Punktes des Triplets (a, b, c) gibt vor, ob der nächste Testpunkt aus dem Intervall (a, b) oder (b, c) stammen soll. Dieses Verfahren bewegt sich iterativ zum Minimum hin und bricht ab, sobald es nur noch aufwärts und nicht mehr abwärts geht.

B.3 Downhill-Simplex Optimierungsverfahren

B.3.1 Algorithmusskizze

Dieses Optimierungsverfahren bedient sich der geometrischen Figur *Simplex*. Ein Simplex ist eine Figur in einem N dimensionalen Raum, bestehend aus $N + 1$ Punkten. In zwei Dimensionen ist ein Simplex ein Dreieck, in drei Dimensionen ein Tetraeder etc.

Diese Methode startet nicht in einem Punkt, sondern in $N + 1$, die die Eckpunkte des ersten Simplex bestimmen. Diese Startwerte können einfach bestimmt werden: Sei P_0 ein Startpunkt, dann sind $P_i = P_0 + \lambda * e_i$ die anderen N Punkte, für $i = 1, ..., N + 1$ (mit $\lambda = const$, passend zum Problem).

Die Methode vollführt nun weitere Schritte, z.B. eine Reflektion. Dabei wird am Simplex der Eckpunkt mit dem größten Funktionswert gespiegelt und neu gesetzt. Wird z.B. ein Talbereich erreicht, können auch Schritte wie Kontraktion und Expansion angewendet werden, um dem Extremwert näher zu kommen.

Um sicher zu gehen, dass tatsächlich ein globales Minimum gefunden wurde, ist es ratsam, am gefundenen Minimum die Methode noch einmal zu starten. Dabei sollten N von den $N + 1$ Eckpunkten des Simplex ausgetauscht werden und nur P_0 als Element des möglichen Minimum-Simplex beibehalten werden. Ist dieses Minimum tatsächlich relevant, wird der Algorithmus es schnell wieder finden.

Die Abbildungen B.3 a.) bis d.) illustrieren die Transformationen. Weitere Informationen zu diesem Thema sind in [PTVF92] zu finden.

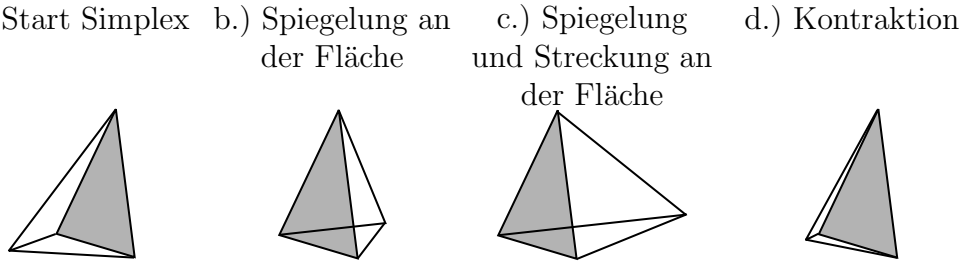


Abbildung B.3: Beispiel für mögliche Simplex-Operationen (in diesem Fall ein Tetraeder) ([PTVF92])

B.4 Evolutionäre Optimierungsverfahren

B.4.1 Algorithmusskizze

Dieser Algorithmus basiert auf der Idee, den Vorgang der *biologischen Evolution* nachzubilden. Dabei werden die Methoden der *Vermehrung*, *Mutation* und *Selektion* genutzt. Der Parameterraum dieses Verfahrens ist genau so groß wie die Anzahl der zu optimierenden Parameter in der Problemstellung.

Zunächst wird der Parametervektor mit Zufallszahlen gefüllt. Dieser Vektor wird jetzt mehrfach verändert. Es wird eine *Population* von verschiedenen Parametervektoren aus dem ersten Parametervektor generiert, indem gleichverteilte Zufallszahlen den Komponenten hinzu addiert werden. Dabei entspricht dieser Schritt einer Mutation. Die Parametervektoren mit den niedrigsten Funktionswerten (Bewegung zum möglichen globalen Minimum hin) werden als Eltern für die nächste Generation genutzt. Dabei entspricht dieser Schritt einer Selektion mit anschließender Vermehrung. Jetzt werden den Komponenten der Eltern-Parametervektoren wieder gleichverteilte Zufallszahlen hinzu addiert. Dabei entsteht eine neue Population.

Dieser Vorgang wiederholt sich so lange bis die maximale vorgegebene Anzahl von Schritten erreicht wird. Einzelne Parameter der Individuen können dabei im Gegensatz zur biologischen Evolution zwischen zwei oder mehreren Individuen ausgetauscht, gemittelt oder auch anders miteinander verknüpft werden. Eine Generation geht aus der vorherigen durch Vermehrung der Eltern mittels Mutation und/oder Vererbung hervor. Dabei können die Eltern, zwecks Verbesserung der zu optimierenden Parameter, in der neuen Generation beibehalten bzw. ausgelassen werden. Allerdings wird im zweiten Fall riskiert, dass die nachfolgende Generation u.U. nur über schlechtere (also weiter vom möglichen globalen Minimum entfernte) Funktionswerte verfügt.

Weitere Informationen zu diesem Thema sind in [Ge98] und in [KT00] zu finden.

Zusammenfassend läßt sich sagen: Die oben genannten Optimierungsverfahren haben Vor- und Nachteile, die sich hauptsächlich in Effizienz und Rechenzeit widerspiegeln. Da die Dimension der zu optimierenden Funktionen im Fall von CNN klein ist, $N < 100$, ist der Speicherplatzbedarf zu vernachlässigen.

Downhill-Simplex und Powell's Linienminimierung sind zwei recht robuste und einfach zu implementierende bzw. zu nutzende Verfahren. Im aufgezeigten Fall kommen sie nur mit den Funktionswerten aus und lassen die Ableitungen außer acht. Downhill-Simplex ist etwas langsamer, dafür jedoch robuster als Powell's Linienminimierung. Trotzdem können beide Verfahren in lokalen Minima zum Abbruch kommen. Vergleiche dazu sind z.B. in [FTIA01] zu finden.

Das Iterative-Annealing und das evolutionäre Optimierungsverfahren sind zwei Algorithmen, die theoretisch immer das globale Minimum finden. Allerdings kann das Training durchaus sehr viele Schritte, u.U. sogar unendlich viele umfassen. Wenn beiden Optimierungsalgorithmen die gleiche Anzahl von Trainingsschritten gewährt werden soll, so ist zu beachten, dass das evolutionäre Verfahren M mal so lange braucht, mit M als Anzahl der Kinder, die pro Optimierungsschritt generiert werden.

Anhang C

CNN-Typen

Weitere Informationen zu den im folgenden Text beschriebenen CNN-Typen sind in [RK99] zu finden. Die Ecken von Bildern mit großen weißen Flächen werden durch schwarze Punkte dargestellt, damit sie von der Seite unterschieden werden können.

C.1 Schwellenwert-CNN

Aufbau des CNN Das Schwellenwert-CNN wird durch folgende Gewichte A, B und den Schwellenwert Z bestimmt:

$$A = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{Bmatrix} \quad B = \begin{Bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{Bmatrix} \quad Z = \{ \; -\dot{z} \; \} \quad \text{mit} \quad -1 < \dot{z} < 1 \qquad \text{(C.1)}$$

Aufgabe bzw. Ablauf Gegeben ist ein statisches Graustufenbild P und eine Schwelle $\dot{z}$. Das Eingabe-Bild U kann beliebig gewählt werden oder per Standardeinstellung $U = 0$ sein.
Das erste Status-Bild wird mit dem Graustufenbild geladen, $X(0) = P$.

Für die Ausgabe gilt nach einer zeitlichen Entwicklung ($Y(t) \Rightarrow Y(\infty)$) Folgendes: Das Ausgabe-Bild ist binär und entspricht P. Dabei entsprechen die schwarzen Bildpunkte den Bildpunkten in P, für dessen Grauwerte $p_{ij} > \dot{z}$ gilt. Die anderen Bildpunkte werden auf die Farbe weiß gesetzt. Die Abbildungen C.1 a.) bis c.) zeigen ein Beispiel für das Schwellenwert-CNN.

C.2 Lücken-Finder-CNN

Aufbau des CNN Das Lücken-Finder-CNN wird durch folgende Gewichte A, B und den Schwellenwert Z bestimmt. Der hier vorgestellte Lücken-Finder arbeitet horizontal und bildet nach rechts ab. Aufgrund der Symmetrie der Problemstellung kann durch das

Drehen von A analog ein vertikaler oder diagonaler Lücken-Finder konstruiert werden.

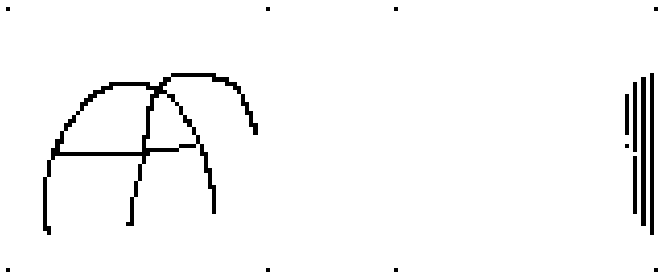
$$A = \left\{ \begin{array}{ccc} 0 & 0 & 0 \\ 1 & 2 & -1 \\ 0 & 0 & 0 \end{array} \right\} \quad B = \left\{ \begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \right\} \quad Z = \left\{ \begin{array}{c} 0 \end{array} \right\} \tag{C.2}$$

Aufgabe bzw. Ablauf Gegeben ist ein statisches binäres Bild P.
Das Eingabe-Bild U kann beliebig gewählt werden oder per Standardeinstellung $U = 0$ sein.
Das erste Status-Bild wird mit dem binären Bild geladen, $X(0) = P$.
Die Randwerte, die virtuellen Zellen, werden auf 0 gesetzt.

Für die Ausgabe gilt nach einer zeitlichen Entwicklung ($Y(t) \Rightarrow Y(\infty)$) Folgendes:
Das Ausgabe-Bild ist ein binäres. Es stellt die Anzahl der horizontalen Löcher in jeder Zeile von P durch genauso viele schwarze Bildpunkte am rechten Rand dar. Die Abbildungen C.2 a.) bis c.) zeigen ein Beispiel für das Lücken-Finder-CNN.

a.) Eingabe-Bild U b.) Ausgabe-Bild $Y(\infty)$ mit $z = 0.3$ c.) Ausgabe-Bild $Y(\infty)$ mit $z = -0.3$

Abbildung C.1: Beispielanwendung des Schwellenwert-CNN mit zwei verschiedenen Schwellenwerten



a.) Eingabe-Bild b.) Ausgabe-Bild

Abbildung C.2: Beispielanwendung des Lücken-Finder-CNN

Anhang D

Rössler-Modellsystem

Das Rössler-Modellsystem ist ein deterministisch-chaotisches System, dessen Trajektorie einen bestimmten Teilbereich seines durch seine Variablen aufgespannten Vektorraumes nicht verlässt. Dieser Vektorraum wird üblicherweise *Zustandsraum* genannt. Der Teilbereich heißt *Attraktor* des Systems.

Das System (siehe Abbildung D.1) wird üblicherweise durch das folgende Differentialgleichungssystem beschrieben:

$$\dot{x} = -0.89\, y - z \tag{D.1}$$

$$\dot{y} = 0.89\, x + 0.165\, y \tag{D.2}$$

$$\dot{z} = 0.2 + z\,(x - 10) \tag{D.3}$$

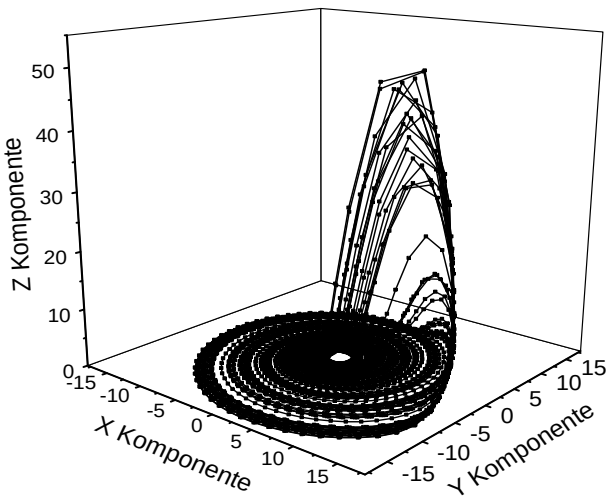


Abbildung D.1: Attraktor des Rössler-Modellsystems

Für das in dieser Arbeit genutzte Rössler-System wurden folgende Anfangsbedingungen genutzt:

$$x(0) = y(0) = z(0) = 1 \tag{D.4}$$

Zur symmetrischen Kopplung zweier nicht identischer Rössler-Systeme wurde ein zusätzlicher linearer Kopplungsterm genutzt. Dabei gibt ϵ die Stärke der Kopplung an.

$$\dot{x}_{1,2} = -\omega_{1,2}y_{1,2} - z_{1,2} + \epsilon(x_{2,1} - x_{1,2}) \tag{D.5}$$

$$\dot{y}_{1,2} = \omega_{1,2}x_{1,2} + 0.165y_{1,2} \tag{D.6}$$

$$\dot{z}_{1,2} = 0.2 + z_{1,2}(x_{1,2} - 10) \tag{D.7}$$

$$\text{mit} \quad \omega_1 = 0.89 \quad \text{und} \quad \omega_2 = 0.85 \tag{D.8}$$

Zur Integration wurde das Runge-Kutta Verfahren vierter Ordnung genutzt (siehe Anhang A.2). Dabei wurde eine Schrittweite von $h = 0.1$ gewählt. Um negative Effekte des *Einschwingvorgangs* (auch als *Transienten* bezeichnet) zu vermeiden, wurde ein Integrationsvorlauf von 1000 Datenpunkten gewählt.

Weitere Informationen zu diesem Thema sind in [Mor98] und in [Mor03] zu finden.

Anhang E

Historische Entwicklung von Neuronalen Netzen

Die folgende Auflistung zeigt einige wichtige Daten in der Entwicklung künstlicher neuronaler Netze in den letzten 60 Jahren:

- 1943 - Warren McCulloch und Walter Pitts (*A logical calcus of the ideas immanent in nervous activity*)
  - zeigten, dass neuronale Netze basierend auf *McCulloch-Pitts-Neuronen* prinzipiell jede arithmetische und logische Funktion berechnen können
  - gaben den Anstoß für weitere Forschung auf diesem Gebiet
- 1949 - Donald O.Hebb (*The Organization of Behavior*)
  - definierte die *Hebb'sche Lernregel* als einfaches, universelles Lernkonzept individueller Neuronen
- 1951 - Marvin Minsky (*Snark*)
  - entwickelte den ersten bekannten Neurocomputer
- 1957/1958 - Frank Rosenblatt und Charles Wightman (*Mark I Perceptron*)
  - entwickelten den ersten erfolgreichen Neurocomputer, der mit einem 20×20 Pixel großen Bildsensor einfache Ziffern erkennt (Gewichte durch 512 motorgetriebene Potentiometer realisiert)
- 1959 - Frank Rosenblatt (*Principles of Neurodynamics*)
  - beschrieb verschiedene Varianten des *Perzeptrons*
  - bewies im *Perzeptron Konvergenz Theorem*, dass das Perzeptron alles, was es repräsentieren kann, auch lernen kann
- 1969 - Marvin Minsky und Seymour Papert

  - analysierten das Perzeptron mathematisch
  - zeigten, dass das Perzeptron z.B. beim XOR-Problem versagt (siehe Anhang E.3, im Abschnitt über das Perzeptron)
  - schlussfolgerten, dass mächtigere Modelle als das Perzeptron die gleichen Probleme haben

Diese Schlussfolgerung ist aus heutiger Sicht falsch. 15 Jahre lang gab es damals kaum Forschungsgelder. Statt dessen wurde massiv das Forschungsgebiet der *Künstlichen Intelligenz* gefördert. Allerdings wurden in dieser Zeit auch die theoretischen Grundlagen für die aktuelle Renaissance geschaffen.

- 1972 - Teuvo Kohonen (*Correlation Matrix Memories*)
  - stellte ein Modell des linearen Assoziators, eines speziellen Assoziativspeichers, vor
- 1974 - Paul Verbos
  - entwickelte das *Backpropagations-Verfahren*
- 1982 - Teuvo Kohonen (*Self-organized Formation of Topologically Correct Feature Maps*)
  - stellte die selbstorganisierten Abbildungen vor
- Stephen Grossberg
  - stellte Modelle (ART-1, ART-2, ART-3, ARTMAP und Fuzzy-ART) der *Adaptive Resonance Theory* vor
- 1982 - John Hopfield (*Neural Networks and physical systems with emergent collective computational abilities*)
  - untersuchte binäre Hopfield-Netze als neuronales Äquivalent zu Ising-Modellen in der Physik
- 1983 - K. Fukushima, S. Miyake und T. Ito (*Necognitron: A Neural Network Model for A Mechanism of Visual Pattern Recognition*)
  - stellten ein neuronales Modell zur positions- und skalierungsinvarianten Erkennung von handgeschriebenen Zeichen vor
- 1986 - D.E. Rumelhart, G.E. Hinton und R.J. Williams (*Learning Internal Representations by Error Propagation*)
  - griffen die Backpropagation als schnelles und robustes Lernverfahren für mehrstufige, vorwärts gerichtete Netze auf

Seit Mitte der 80er Jahre im 20 Jh. entwickelte sich dieses Forschungsgebiet wieder mit hoher Geschwindigkeit. Rumelhart, Hinten und Williams schafften den erneuten Aufschwung für die Erforschung künstlicher neuronaler Netze.

Die folgenden Abschnitte gehen auf einige wichtige Aspekte künstlicher neuronaler Netze ein.

E.1 McCulloch und Pitts

McCulloch und Pitts entwickelten ein Modell mit folgenden Annahmen:

- ein Neuron ist ein aktives oder passives binäres Schaltelement
- ein Neuron besitzt einen festen Schwellenwert
- ein Neuron empfängt Eingaben sowohl von erregenden Synapsen (alle mit gleichem Gewicht), als auch von hemmenden Synapsen
- eine einzige hemmende Synapse verhindert die Neuronaktivierung

Dieses Modell wurde unter Anwendung von endlichen Automaten und Bool'schen Funktionen untersucht. Alle Bool'schen Funktionen lassen sich durch ein dreischichtiges Netz realisieren. Wird die XOR-Funktion weggelassen, reichen auch zwei Schichten. Das folgt aus der Tatsache, dass alle Bool'schen Funktionen in konjunktiver[1] oder disjunktiver[2] Normalform darstellbar sind.

Dadurch, dass das McCulloch-Pitts-Neuron keine Gewichtung der Eingabe beinhaltet, ist es ein statisches Modell. Selbstmodifikation ist nicht möglich, damit ist auch die Möglichkeit des Lernens ausgeschlossen.

E.2 Hebb'sche Lernregel

Die erste Lernregel wurde von dem Psychologen Donald Hebb im Jahr 1949 formuliert. Sein Algorithmus, mit dem er die Lernfähigkeit des Gehirn zu erklären versuchte, sieht folgendermaßen aus:

`Wenn ein Axon einer Zelle A nahe genug ist, um eine Zelle B zu erregen und sich wiederholt oder dauerhaft am Feuern beteiligt, geschieht ein Wachstumsprozess oder eine metabolische Änderung in einer oder beiden Zellen dergestalt, dass A's Effizienz als eine der auf B feuernden Zelle anwächst.`

Auf künstliche Neuronen übertragen kann der Algorithmus folgendermaßen formuliert werden:

[1] Darstellung der Bool'schen Funktion als Terme, bestehend aus logischen *Oder*-Verknüpfungen, die mit logischen *Und*-Verknüpfungen verbunden sind.

[2] Darstellung der Bool'schen Funktion als Terme, bestehend aus logischen *Und*-Verknüpfungen, die mit logischen *Oder*-Verknüpfungen verbunden sind.

```
Wenn Zelle j eine Eingabe von Zelle i erhält und beide gleichzeitig und
stark aktiviert sind, erhöht sich das Gewicht zwischen den beiden
Zellen (von i in Richtung j).
```

Die folgende Formel ist die mathematische Formulierung der Lernregel:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \Delta\omega_{ij} \qquad \text{mit} \qquad \Delta\omega_{ij} = \eta * o_i * a_j \tag{E.1}$$

Dabei ist $\Delta\omega_{ij}$ die Änderung des Gewichtes ω_{ij}, $\eta > 0$ die Lernrate (konstant), o_i die Ausgabe der Vorgängerzelle i und a_j die Ausgabe der Nachfolgerzelle j.
Problematisch bei dieser Lernregel ist, dass Neuronen in diesem Fall nicht vergessen können. ω_{ij} wächst bei dauerhafter Aktivierung ins Unendliche.

E.3 Perzeptron

Das erste effektive künstliche neuronale Netzwerk wurde 1958 von Frank Rosenblatt entwickelt, das *Perzeptron*. Das Perzeptron ist ein zweischichtiges vorwärts gerichtetes Netzwerk. Eine künstliche Retina ist mit der Eingabeschicht verbunden. Diese Verbindungen sind unveränderlich und zufällig, wobei jedes Neuron auch mit mehreren Bildpunkten verbunden werden kann. Zwischen der Eingabeschicht und der Ausgabeschicht existieren Verbindungen mit veränderbaren Gewichten. Die Ausgabeschicht besteht aus einem Neuron. Sie addiert alle Impulse aus der Eingabeschicht und wird aktiv, wenn ein veränderbarer Schwellenwert überschritten wird. Dabei kann sie nur zwei Zustände annehmen, *erkannt* und *nicht erkannt*. Eine Anmerkung zur Mächtigkeit des Perzeptrons: Ein wie oben betrachtetes, einstufiges Perzeptron kann z.B. das XOR-Problem nicht lösen. Einstufige Perzeptronen können nur linear separierbare Mengen lösen. Diese Mengen sind durch eine *Hyperebene* trennbar. Durch Einführung weiterer Ebenen kann die Mächtigkeit erhöht werden, Probleme mit höherer Komplexität können gelöst werden.

Anhang F

SCNN - ein universeller Software Simulator

F.1 Vorarbeiten

Die Portierung von SCNN nach Linux wurde ohne große Probleme realisiert. Da Linux auch ein Unix-ähnliches Betriebssystem ist, mussten nur Verzeichnispfade, ein anderer Compiler und Linker und deren Optionen neu eingestellt werden. Als Compiler wurde der zur Kompilationszeit aktuelle GCC 3.2 genutzt. Die hiermit übersetzte Version erwies sich als stabil und wurde unter Linux primär für das Training genutzt.
Die Portierung von SCNN nach Windows gestaltete sich komplizierter. Hier muss zwischen zwei Portierungen unterschieden werden, die für unterschiedliche Zwecke genutzt wurden.

1. CYGWIN

   - Der Quellcode wurde mit dem Werkzeugsystem CYGWIN[1] kompiliert. Dabei konnten die gleichen Einstellungen genutzt werden wie bei der Linux Portierung, da CYGWIN ein Unix-ähnlicher Aufsatz für Windows ist. Allerdings müssen die Programme für die Nutzung unter Windows neu übersetzt werden. Danach starten sie mit einer Bibliothek, die bestimmte Befehle von Unix emuliert. Die hiermit übersetzte Version erwies sich als stabil und wurde unter Windows zur Simulation und für das Training genutzt.

2. Borland C++ Builder 6.0

   - Die unter Windows Nutzern weit verbreitete Entwicklungsumgebung Borland C++ Builder 6.0 (*BCB*) wurde für eine native Portierung von SCNN nach Windows genutzt. Diese Version wurde allerdings nicht produktiv genutzt. Statt dessen wurde häufig von dem hervorragenden integrierten *Debugger* Gebrauch gemacht. Reparaturen und Erweiterungen, aber auch Funktionsabläufe konnten direkt im Quelltext während des Ablaufs beobachtet und analysiert werden.

[1] www.cygwin.org - Ein Unix-ähnlicher Aufsatz für Windows, mit dem Unix-Programme mit kleinen Änderungen direkt unter Windows übersetzt und genutzt werden können.

F.2 Simulation

Dieses Unterkapitel zeigt anhand des AND-CNN (siehe Kapitel 3.1.2) wie die Simulation im SCNN beschrieben und ausgeführt wird.

Zunächst muss das Netz konfiguriert werden. Dies geschieht durch ein Skript, die Templates und drei Bilder, die die Werte für die Eingabe, den Status und den Schwellenwert enthalten. Die Templates und die Bilder werden üblicherweise in ASCII, also *Klartext*, abgelegt. Die folgenden Quelltexte sind ausführlich dokumentiert.

Das Format für die Templates, am Beispiel des A Templates:

```
#TEMPLATE
# Generated by SCNN_Remote (C) Robert Sowa

name=xyz

invariant

neighbourx=1
# Nachbarn in X Richtung
neighboury=1
# Nachbarn in Y Richtung
linear=1
# Polynomkoeffizient D

#also ein einfaches 3x3x1 Template

values:
0 0 0
0 2 0
0 0 0
# die Werte
```

Das Format für die Bilder:

```
#RFDATA
# Generated by SCNN_Remote (C) Robert Sowa

dataname=E:\bildname.rfnet
sizex=64
# Anzahl der Elemente in X-Richtung
sizey=64
# Anzahl der Elemente in Y-Richtung
sizez=1
# Anzahl der Elemente in Z-Richtung
```

```
# also ein planes 64x64x1 Bild

values:
-1.0   1.0   ...
-1.0   1.0   ...
 ...   ...   ...
# Zahlenwerte der einzelnen Zellen, ueblicherweise
# im Wertebereich -1..1
```

Die restliche Konfiguration geschieht im *Steuer-Skript*:

```
SCNN

set var "scnnmode"=2;
# Kommandozeilen Modus
set var "simsteps"=500;
# 500 Schritte beim Integrationsalgorithmus
set var "outputfunc"=1;
# Sigmoidale Ausgabefunktion
set var "calcmethod"=1;
# Runge-Kutte 4ter Ordnung Integrationsmethode

# Weitere Werte wie die Integrations-Schrittweite
# (h=0.1), das Beta in der Sigmoidalen Funktion
# (Beta=4) oder die Randzelleneinstellung (alle
# Randzellen=0) werden automatisch per Standardeinstellung
# gesetzt, koennen aber auch veraendert werden

load template 0 n state "ta.tem"
load template 0 n input "tb.tem"
# das A und B Template laden

load img pbm 0 input "input.pbm"
load img pbm 0 state "state.pbm"
load img rfnet 0 bias "bias.rfnet"
# die Bilder, fuer die Eingabe, den Status und
# den Schwellenwert, laden

start sim
# !!!!!!!!!!!!!!!!!!!!!!!!!
# ! die Simulation starten !
# !!!!!!!!!!!!!!!!!!!!!!!!!

save img rfnet 0 output "out.rfnet"
```

```
save img pgm 0 output "out.pgm"
# die Ausgabe in zwei verschiedenen
# Formaten wegschreiben, also Fließkommazahlen
# Tabelle und als Graustufenbild

end
```

Durch den Befehlsaufruf *scnn.exe -I scriptname.scr* wird der Simulator gestartet. Zu beachten ist, dass das Ausgabebild *vollkommen* abhängig ist von der *Ausgangskonfiguration* des Simulators. Damit ist auch die Architektur des Computers gemeint (z.B. Auflösung der Fliesskommadatentypen), schließlich wird der Simulator in dieser gestartet.

F.3 Bewertungsfunktionen

Folgende Bewertungsfunktionen stehen in SCNN zur Verfügung:

- Bewertungsfunktionen $FCompare(Y(\tau_{trans}), Y^{Ref})$ (mit $N_{Bildpunkte}$ als Anzahl der Bildpunkte)

  0 - MSE (normiert) (MSEn) $\frac{1}{4N_{Bildpunkte}} \sum_{i=1}^{N_{Bildpunkte}} \left(y_i(\tau_{trans}) - y_i^{Ref}\right)^2$

  Mean-Square-Error normiert

  1 - MSE $\frac{1}{N_{Bildpunkte}} \sum_{i=1}^{N_{Bildpunkte}} \left(y_i(\tau_{trans}) - y_i^{Ref}\right)^2$

  Mean-Square-Error

  2 - ME $\frac{1}{N_{Bildpunkte}} \sum_{i=1}^{N_{Bildpunkte}} \left|y_i(\tau_{trans}) - y_i^{Ref}\right|$

  Betragsfehler

  3 - UNMSE $\sum_{i=1}^{N_{Bildpunkte}} \left(y_i(\tau_{trans}) - y_i^{Ref}\right)^2$

  Absoluter Mean-Square-Error

  4 - RMSE $\sum_{i=1}^{N_{Bildpunkte}} \frac{\left(y_i(\tau_{trans}) - y_i^{Ref}\right)^2}{y_i(\tau_{trans})^2}$

  Relativer Mean-Square-Error

  5 - PE $\frac{1}{2N_{Bildpunkte}} \sum_{i=1}^{N_{Bildpunkte}} \left[\max\left(y_i(\tau_{trans}), y_i^{Ref}\right) - \min\left(y_i(\tau_{trans}), y_i^{Ref}\right)\right]$

  Prozentueller Fehler (Fehler im Bereich [0,1])

  6 - PEMAX

  maximaler prozentueller Wert

  7 - MEMAX

  maximaler Betragswert

  8 - GAREA

Fläche um die Diagonale in der Grauwertkorrelationmatrix beider Bilder - Flächenberechnung mit linearen Verbindungsstücken der Punkte ober- und unterhalb der Grauwertkorrelationsmatrix gelöst

9 - GMAX

maximaler Abstand von der Diagonalen in der Grauwertkorrelationsmatrix (maximale Lotlänge)

10 - MEDIAN

Median der Grauwertabstände (Fehler) aller Bildpunkte

11 - RANGE

Abstand zwischen kleinstem und größtem Fehler aller Bildpunkte

12 - QUARTILE

Abstand zwischen 25 und 75 Prozent Median-Wert, problematisch, wenn die Anzahl Bildpunkte nicht durch vier teilbar ist, mindestens vier Bildpunkte sind erforderlich

13 - DECLILE

Abstand zwischen 10 und 90 Prozent Median-Wert, problematisch, wenn die Anzahl Bildpunkte nicht durch 10 teilbar ist, mindestens 10 Bildpunkte sind erforderlich

14 - CORRELATION COEFFICIENT $\frac{\overline{Y(\tau_{trans})Y^{Ref}}-\left(\overline{Y(\tau_{trans})}\right)\left(\overline{Y^{Ref}}\right)}{\sqrt{\overline{Y(\tau_{trans})^2}-\overline{Y(\tau_{trans})}^2}\sqrt{\overline{(Y^{Ref})^2}-\overline{Y^{Ref}}^2}}$

Normierter Korrelation-Koeffizient zwischen den Grauwerten der Bildpunkte

15 - RANK CORRELATION COEFFICIENT

Normierter Rank-Korrelation-Koeffizient zwischen den Grauwerten der Bildpunkte

16 - SUM SQUARE DIFFERENCE OF RANKS

Normierter (auf die Anzahl der benutzten Bildpunkte) SUM SQUARE DIFFERENCE OF RANKS zwischen den Grauwerten der Bildpunkte

Diese Bewertungsfunktionen können in mehrere Klassen von Maßen zusammengefaßt werden:

Klasse1 (Bewertungsfunktionen 0,1,2,3,4,5) Die Mittelwerte der linearen oder quadrierten Grauwertabstände der zu vergleichenden Bilder werden als Fehlermaße verwendet. Das Problem bei diesen Maßen ist, dass Strukturen in den Bildern nicht verglichen werden können. Bilder können als gut bewertet werden, auch wenn einige Bildpunkte einen sehr großen Grauwertabstand haben. Die Mittelung wischt über alle Bildpunkte und wirkt noch stärker, wenn die Bildgröße erhöht wird.

Klasse2 (Bewertungsfunktionen 6,7) Der Maximum-Wert der Grauwertabstände der zu vergleichenden Bilder wird als Fehlermaß verwendet. Dies ist ein sehr hartes Kriterium. Erst wenn der Abstand der beiden *schlechtesten* Bildpunkte verkleinert wurde, wird auf andere

eingegangen. Das Problem hierbei ist, dass es zu einem Rundlauf in einem Bild kommen kann und nicht alle Bildpunkte optimiert werden.

Klasse3 (Bewertungsfunktionen 10,11,12,13) Der Median selbst lässt, wie die Max-Fehlermaße, einzelne Bildpunkte optimieren. Allerdings finden in diesem Fall die Bildpunktwechsel häufiger statt. Die folgenden drei Maße (11,12,13) liefern einen Fehler, der relativ zu zwei Grauwertabständen in den Bildern ist. Sie ziehen die Grauwertabstände zueinander, trotzdem können beide recht hoch sein. Damit ist das Erreichen der Ähnlichkeit zwischen den beiden Bildern nicht gewährleistet.

Klasse4 (Bewertungsfunktionen 8,9) Diese Maße arbeiten mit den Grauwertwerten der zu vergleichenden Bilder. Werden diese in ein 2D-Histogramm mit den Grenzen (-1,1,-1,1), eingetragen, so entsteht eine Punkte-Wolke. Wenn beide Bilder ähnlich oder gleich sind, zieht sich diese Wolke in Richtung der Diagonalen zusammen. Entsteht z.B. eine Gerade parallel über bzw. unter der Diagonalen, ist das zu vergleichende Bild heller bzw. dunkler als das andere. (8) misst die Fläche, die von den Punkten (wenn man die äußersten mit einer Linie verbindet) um die Diagonale herum eingenommen wird. Dies ist auch ein recht hartes Maß, da es direkt auf alle Details der zu vergleichenden Bilder eingeht. (9) misst die maximale Länge, die ein Lot (senkrechte Verbindung zwischen einem Punkt und der Diagonalen) auf der Diagonalen hat. Auch hier kann entsprechend zu oben ein Bild heller oder dunkler sein.

Klasse5 (Bewertungsfunktionen 14,15,16) Auch diese Maße sind gut geeignet, um das Vorhandensein von Details in den zu vergleichenden Bildern zu prüfen. Die letzten beiden Maße nutzen eine andere Menge von Zahlen als Grauwertwerte. Das letzte Maß kann in das vorletzte überführt werden. Allerdings entstehen auch hier Probleme, wenn z.B. eines der Bilder eine Fläche ist. Dann ist der Nenner der Gleichung = 0, weil einer der *Sigma-Werte* im Nenner 0 ist. Daher muss dieser Fall anders bewertet werden, was wiederum für den Optimierungsalgorithmus einen *Wechsel* des Fehlergebirges bedeutet. Tritt dieses Problem nicht auf, dann liegt immer noch ein sehr hartes Maß vor, da es direkt auf *alle* Details in den zu vergleichenden Bildern eingeht.

Die Bewertungsfunktionen 5 bis 16 wurden im Rahmen dieser Arbeit in das SCNN System implementiert.

F.4 Zusammenführungsfunktionen

Folgende Zusammenführungsfunktionen stehen in SCNN zur Verfügung:

- Zusammenführungsfunktion $FCompose(i \in [1, N_m], Err_i^{local})$ (mit N_m als Mächtigkeit der Trainingsmenge)

  Nr.0 $\frac{1}{N_m} \sum_{i=1}^{N_m} Err_i^{local}$

  Mittelwert

  Nr.1 $\frac{N_m}{\sum_{i=1}^{N_m} \frac{1}{Err_i^{local}}}$

  Harmonisches Mittel

Nr.2 $\sqrt{\frac{1}{N_m}\sum_{i=1}^{N_m}(Err_i^{local})^2}$

Root-Mean-Square

Nr.3 der mittlere Wert nach Sortierung

Median

Nr.4 der maximale Wert

Max Value

Folgende Probleme können bei Nutzung der Zusammenführungsfunktionen auftreten:

Bei einem Trainingsset mit mehr als einer Eingabe U, einem Status $X(0)$ und einer Referenz Y^{Ref} muss für jedes Tripel das eigene Fehlergebirge gleichermaßen nach einem (globalen) Minimum durchsucht werden. Dies ist problematisch, da es durchaus vorkommen kann, dass ein Minimum eines Tripels mit einem (lokalen) Maximum eines anderen Sets von der Position her übereinstimmt.

Die Zusammenführungsfunktion Nr.0 verwischt alle Fehlergebirge durch eine Mittelung. Der oben genannte Fall kann eintreten: Ein Minimum wird durch die Mittelung angehoben und dadurch bei der Optimierung übersehen.

Die Zusammenführungsfunktion Nr.3 springt von dem Fehlergebirge eines Tripels in das nächste des folgenden Tripels. Die Optimierungsverfahren können u.U. keinen gleichmässigen Abstieg schaffen, wenn die Fehlergebirge sich stark unterscheiden.

Die Zusammenführungsfunktion Nr.4 bleibt im Fehlergebirge des Tripels mit dem schlechtesten Fehlerwert. Das Optimierungsverfahren verweilt so lange im Fehlergebirge, bis ein anderes Tripel schlechter bewertet wird. Dies kann zu Rundläufen führen, nicht alle Tripel werden gleich gut optimiert.

Die Zusammenführungsfunktionen Nr.1 und Nr.2 erwiesen sich als nicht praktikabel. Die Zusammenführungsfunktionen Nr.0, Nr.3 und Nr.4 basieren entweder auf einer Mittelung (mischen) der Fehlergebirge oder auf dem Herauspicken eines speziellen Fehlergebirges. Die Zusammenführungsfunktionen Nr.1 und Nr.2 vermischen die Fehlergebirge derart, dass ihre Grundstruktur nicht mehr erhalten bleibt. Die Optimierungsverfahren können somit keine erfolgreiche Suche nach dem (globalen) Minimum durchführen.

Die Zusammenführungsfunktionen Nr.1 bis Nr.4 wurden im Rahmen dieser Arbeit in das SCNN System implementiert.

F.5 Optimierung

Die folgenden dokumentierten Steuer-Dateien stellen Beispiele für Training mit einer und mit mehreren Referenzen dar.

Training mit einer Referenz

```
SCNN

set var "scnnmode"=2;
```

```
set flag "train_feedforward"=1;
# das B Template trainieren
set flag "train_feedback"=1;
# das A Template trainieren
set flag "train_bias"=1;
# den Schwellenwert trainieren

set flag "writevector"=1;
# Status Ausgabe
set flag "epoch_train_error"=1;
# Status Ausgabe

set var "simsteps"=400;
set var "trainsteps"=500;
# Anzahl der Trainingsschritte waehlen

set var "outputfunc"=1;

set var "trainmethod"=4;
# Trainingsmethode waehlen

set var "errortype"=5;
# Bewertungsfunktion waehlen


load template 0 n state "ta.tem"
load template 0 n input "tb.tem"
load img rfnet 0 bias "b.rfnet"
# zu trainierende Werte vorbelegen


addtotrainlist pgm 0 ref "mln.pgm"
# Referenzbild laden
addtotrainlist pgm 0 input "ml.pgm"
# Eingabebild laden
addtotrainlist simnow rfnet 0 state "6464null.rfnet"
# Statusbild laden

start learn
# !!!!!!!!!!!!!!!!!!!!!!
# ! das Training starten !
# !!!!!!!!!!!!!!!!!!!!!!
```

```
save template 0 state "gentema.tem"
save template 0 input "gentemb.tem"
save img rfnet 0 bias "calcbias.rfnet"
# die bestimmten Werte fuer das A und B Template und
# den Schwellwert speichern

end
```

Training mit mehreren Referenzen

```
SCNN

set var "scnnmode"=2;

set flag "train_feedforward"=1;
set flag "train_feedback"=1;
set flag "train_bias"=1;

set flag "writevector"=1;
set flag "epoch_train_error"=1;

set var "simsteps"=400;
set var "trainsteps"=500;

set var "outputfunc"=1;

set var "trainmethod"=4;

set var "errortype"=5;

set var "epoch_mean_type"=5;
# Zusammenfuehrungsfunktion waehlen

load template 0 n state "ta.tem"
load template 0 n input "tb.tem"
load img rfnet 0 bias "b.rfnet"

addtotrainlist pgm 0 ref "mln1.pgm"
addtotrainlist pgm 0 input "ml1.pgm"
addtotrainlist simnow rfnet 0 state "6464null.rfnet"
# Erste Referenz, die beruecksichtigt werden soll

addtotrainlist pgm 0 ref "mln2.pgm"
addtotrainlist pgm 0 input "ml2.pgm"
```

```
addtotrainlist simnow rfnet 0 state "6464null.rfnet"
# Zweite Referenz, die beruecksichtigt werden soll

start learn

save template 0 state "gentema.tem"
save template 0 input "gentemb.tem"
save img rfnet 0 bias "calcbias.rfnet"

end
```

Anhang G

Aladdin CNN System

G.1 Anbindung an den PC

Ein Betriebssystem der WindowsNT Familie (WindowsNT, Windows2000, WindowsXP) wird auf dem PC benötigt, um den Aladdin Visual Computer Stack anzusteuern. *CNN-Edit* und *CNN-Run* sind die beiden Haupt-Programme, mit denen der Stack gesteuert werden kann. Weiterhin existiert noch eine Anbindung an *Matlab*, die als Toolbox realisiert wird. Analogic arbeitet auch an einer Anbindung an die Borland CBuilder Entwicklungsumgebung, die den direkten Zugriff auf den Stack, bis hin zur direkten Steuerung des DSP, möglich macht.

G.2 Anbindung an SCNN

Die von Analogic mitgelieferte Software kann zwar den ACE4K zum Berechnen von vielfältigen CNN-Operationen nutzen, aber keine Training-Szenarien ausführen. Auch bei Nutzung der Matlab-Toolbox können erst nach einiger Programmierarbeit Trainingszenarien durchgeführt werden. Zudem ist von großem Nachteil, dass Erfahrungen und Ergebnisse von SCNN nicht direkt mit den Ergebnissen der Toolbox verglichen werden können.

Deshalb wurde im Rahmen dieser Dissertation ein Möglichkeit entwickelt, beide Systeme, das SCNN und den ACE4K, zu vereinen. Der Vorteil liegt darin, dass die Ergebnisse aufgrund der identischen Infrastruktur vergleichbar sind. Nur das CNN, genauer der Software-Simulator des SCNN, der ACE4K oder der Software-Simulator des Stacks, werden entsprechend ausgetauscht.

Technische Realisation

Die Steuerung der Simulation und des Trainings erfolgt im SCNN System genauso wie bisher (siehe Kapitel 3.2.2 und Kapitel 3.2.3). Wird der ACE4K oder der Software-Simulator des Stacks als CNN genutzt, dann ergibt sich folgende Änderung in den Abläufen für eine CNN-Operation (also eine Simulation bzw. für einen Trainingsschritt):

1. Wegschreiben der Eingabe U und Status $X(0)$ und der Parameter (A und B Templates und Schwellenwert)
2. Starten von CNN-Run, welches die unter 1. weggeschriebenen Daten an den Stack zur Ausführung weiterreicht
3. Einlesen des Ausgabe-Bildes $Y(\tau_{transChip})$ vom Stack in die SCNN Strukturen
4. Fortfahren des Ablaufes im SCNN-System

Der Vorteil ist, dass auch hier durch zuschaltbare Protokolle Einblick genommen werden kann und sogar die Kommunikations-Daten zwischen dem SCNN und dem Stack eingesehen werden können. Werden die Ausgabe- und Einlese-Operationen in einer *RAM-Disk* durchgeführt, so können etwa zwei bis drei CNN-Operationen pro Sekunde durchgeführt werden (AthlonXP 2 Ghz PC mit 256 MB DDRAM, Windows2000). Die gleichen Aufgaben erreichen mit dem Simulator des SCNN-Systems auf entsprechend ausgerüsteten PCs unter Linux, je nach Einstellung mehrere 1000 bis 10000 CNN-Operationen pro Tag. Daher ist es trotz der zeitaufwändigen Ausgabe- und Einlese-Operationen bei jedem Schritt interessant, den ACE4K zu Trainingszwecken zu nutzen. Denn der ACE4K ist immer noch eine bis zwei Größenordnungen schneller als der Simulator des SCNN-Systems, bei vergleichbaren CNN-Topologien. Durch direkte Anbindung des ACE4K an das SCNN-System kann die Geschwindigkeit noch weiter gesteigert werden.

Möglichkeiten und Einschränkungen

Der ACE4K und der Simulator der Stacks ermöglichen die Bearbeitung von mindestens 64×64 großen Bildern. Der ACE4K kann *keine* kleineren verarbeiten. Größere Bilder werden durch eine Kachelfunktion (siehe Kapitel 3.3.2) realisiert. Interessant ist, dass die Ausführung genauso schnell ist wie bei einem 64×64 Bild. Dieser Effekt entsteht durch die Tatsache, dass die Ein- und Ausgabe in den Chip viel mehr Zeit kosten als die tatsächliche Berechnung. Allerdings bleibt noch zu untersuchen, wie sich die Kachelung auf das Training im ACE4K auswirkt.

Die A und B Templates müssen genau 3×3 Elemente groß sein (1er Nachbarschaft). Kleinere Templates können nur realisiert werden, indem die entsprechenden Gewichte auf 0 gesetzt werden. Für den ACE4K gilt die Einschränkung, dass die Templatewerte auf den Wertebereich $[-3, 3]$ beschränkt sind.

Der Schwellenwert wird in der aktuellen Realisation der Anbindung invariant gesetzt. Es ist aber problemlos möglich, eine Erweiterung zum varianten Schwellenwert zu programmieren. Damit wird ein 64×64 großes Graustufenbild als variante Schwellenwert-Matrix geladen.

G.3 Berechnung und Simulation

Im Aladdin CNN System kann entweder der ACE4K zur Berechnung oder der Simulator des Stacks zur Simulation genutzt werden. Beide CNN-Realisationen werden über dieselbe

Programmiersprache angesprochen, nur das Ziel (der ACE4K oder der Simulator) muss entsprechend gesetzt werden. Die Programmierung des Aladdin CNN Systems geht folgendermaßen von statten:

1. Der Algorithmus wird in einem Flussdiagramm aufgebaut. Dabei werden die notwendigen Templates und Subroutinen berücksichtigt.

2. Das Flussdiagramm wird in *ALPHA-Source-Code* übersetzt. ALPHA ist eine lesbare Sprache, die die Steuerung des Stacks erlaubt ([ACA02]).

3. Der *ALPHA-Compiler* erzeugt den *AMC* (Aladdin Macro Code). Diese Sprache ist ebenfalls lesbar, allerdings ist sie sehr schaltungstechnisch orientiert. Natürlich kann der Algorithmus direkt in AMC umgesetzt werden.

4. Die AMC-Datei wird nun in *ABC* gewandelt. ABC ist eine Maschinensprache, mit der der Stack angesteuert wird.

5. Per ABC wird nun, durch CNN-Run, der ACE4K oder der Software-Simulator angesteuert.

Der folgende AMC-Quelltext ist ein Beispiel für typische CNN-Operationen. Natürlich können auch komplexere Abläufe mit mehreren CNN-Operationen realisiert werden.

```
host.load.tem d:\testml\my.tem tem33
;Templates und Schwellenwert Parameter fuer z.B. AND Operation laden
mov.tem.tem tem33 TEM1
;Parameter in TEM1 auf den Stack speichern

host.load.pic d:\testml\inp.bmp lam5
;Eingabe-Bild laden
mov.lam.lam lam5 LAM1
;Eingabe-Bild in LAM1 auf den Stack speichern

host.load.pic d:\testml\sta.bmp lam6
;Status-Bild laden
mov.lam.lam lam6 LAM2
;Status-Bild in LAM2 auf den Stack speichern

ar.tem TEM1 LAM1 LAM2 LAM3 500 0 0 NIL NIL
;CNN starten

mov.lam.lam LAM3 lam8
;Ergebnis-Bild vom Stack holen
host.save.pic d:\testml\outp.bmp lam8
;Ergebnis-Bild speichern
end
```

Literaturverzeichnis

[AC00] Analogic Computer LTD., *"Aladdin V1.3 - SimCNN multi-layer CNN simulator for visual mouse platform reference manual - Version 4.1"*, Budapest, Hungary, 2000

[ACS00] Analogic Computer LTD., *"Aladdin V1.3 - System specification - list of documents - Version 4.1"*, Budapest, Hungary, 2000

[ACI00] Analogic Computer LTD., *"Aladdin V1.3 - System identification: Objectives, architecture and components - Version 4.1"*, Budapest, Hungary, 2000

[ACT00] Analogic Computer LTD., *"Aladdin V1.3 - TemMaster - Template design and optimization tool for binary input-output CNNs - Version 4.1"*, User's Guide, Budapest, Hungary, 2000

[ACC99] Analogic Computer LTD., *"CADETWin-99 - Algorithm design using high level description and visual input via the ALPHA language and compiler - Version 3.0"*, User's Guide, Budapest, Hungary, 1999

[ACO02] Analogic Computer LTD., *"Aladdin Professional - Overview - Version 2.1"*, Budapest, Hungary, 2002

[ACAMC02] Analogic Computer LTD., *"Aladdin Professional - Extended Analogic Macro Code (AMC) - Version 2.1"*, Reference Manual, Budapest, Hungary, 2002

[ACA02] Analogic Computer LTD., *"Aladdin Professional - CNN ALPHA language and compiler - Version 2.1"*, Reference Manual, Budapest, Hungary, 2002

[ACC02] Analogic Computer LTD., *"Aladdin Professional - Algorithm design using high level description and visual input via the ALPHA language and compiler - Version 2.1"*, User's Guide, Budapest, Hungary, 2002

[ACCR02] Analogic Computer LTD., *"Aladdin Professional - CNNRun - Manual - Version 2.1"*, Budapest, Hungary, 2002

[ACCE02] Analogic Computer LTD., *"Aladdin Professional - CNNEdit - Manual - Version 2.1"*, Budapest, Hungary, 2002

[AGLE99] J. Arnhold, P. Grassberger, K. Lehnertz, C.E. Elger, *"A robust method for detecting interdependences: application to intracranially recorded EEG"*, Physica D, 134:419-430, 1999

[Ame98] C. Ames, *"Klassifikation von Neurosignalen mit künstlichen Zellularen Neuronalen Netzwerken"*, Diplomarbeit am Institut für Angewandte Physik der Johann Wolfgang Goethe-Universität Frankfurt am Main, 1998

[AVR86] V.S. Afraimovich, N.N. Verichev, M.I. Rabinovich, *"General synchronization"*, Izv. Vyssh. Uch. Zav. Radiozifika, 29:795-803, 1986

[BCG82] E.R. Berlekamp, J.H. Conoway, H.K. Guy, *"Winning Ways for your Mathematical Plays"*, Academical Press, New York, USA, 1982

[BS95] I.N. Bronstein, K.A. Semendjajew, *"Taschenbuch der Mathematik"*, B.G. Teubner Verlagsgesellschaft, Frankfurt am Main, 1995

[Ca03] A. Chernihovskyi, *"Time series analysis with excitable nonlinear media simulated with cellular neural networks"*, Diplomarbeit am Helmholtz-Institut für Strahlen- und Kernphysik und an der Klinik für Epileptologie der Rheinischen Friedrich-Wilchelms-Universität Bonn, Dezember 2003

[CFCCCPS02] C. Castellaro, G. Favaro, A. Castellaro, A. Casagrande, S. Castellaro, D.V. Puthenparampil, C.F. Salimbeni, *"An artificial intelligence approach to classify and analyse EEG traces"*, Clinical Neurophysiology, 32(3):193-214, 2002

[Ch98] L.O. Chua, *"CNN: A paradigm for complexity"*, World Scientific Publishing Co. Ltd., Singapur, Serie A, Vol. 31, 1998

[CY88] L.O. Chua, L. Yang , *"Cellular Neural Networks: Theory and Applications"*, IEEE Trans. Neural Networks, 35:1257-1272, 1988

[ECDR96] S. Espejo, R. Carmona, R. Dominguez-Castro, A. Rodriguez-Vazquez, *"A CNN universal chip in CMOS technology"*, Int. J. Circ. Theor. Appl., 24:93-109, 1996

[El01] C.E. Elger, *"Future Trends in epileptology"*, Curr. Opin. Neurol., 14:185-186, 2001

[EP97] J. Engel Jr., T.A. Pedley, *"Epilepsy: A comprehensive Text-Book"*, Philadelphia: Lippicott-Raven, USA, 1997

[ER85] J.-P. Eckmann, D. Ruelle, *"Ergodic theory of chaos and strange attractors"*, Rev. Mod. Phys., 57:617-656, 1985

[FCRL93] L. Fabiny, P. Colet, R. Roy, D. Lenstra, *"Coherence and phase dynamics of spatially coupled solid-state lasers"*, Phys. Rev. A, 47:4287-4296, 1993

[FTFE01] D. Feiden, R. Tetzlaff, *"Feature extraction in motion estimation with cellular neural networks using iterative annealing"*, Proc. ECCTD 01, 3:413-416, Espoo, Finnland, 2001

[FTIA01] D. Feiden, R. Tetzlaff , *"Iterative Annealing: A new efficient optimization method for Cellular Neural Networks"*, Proc. IEEE International Conference on Image Processing (ICIP 2001), Thessaloniki, Griechenland, 2001

[Ga46] D. Gabor, *"Theory of communications"*, Proc. IEE London 93:429-457, 1946

[GP93] P. Grassberger, I. Procaccia, *"Characterization of strange attractors"*, Phys. Rev. Lett. 50:346-349, 1983

[Ge98] G.C. Geis, *"Untersuchung der Auswirkungen von Parameterabweichungen auf das Verhalten von Zellularen Neuronalen Netzwerken"*, Diplomarbeit am Institut für Angewandte Physik der Johann Wolfgang Goethe-Universität Frankfurt am Main, 1998

[Haeng] M. Hänggi, *"Java CNN Simulator Applet"*, http://www.isi.ee.ethz.ch/~haenggi/CNNsim.html

[Ha78] F.J. Harris, *"On the use of windows for harmonic analysis with the discrete Fourier Transform"*, Proc. IEEE, 66:51-83, 1978

[Ha02] C. Hand, *"Epicenter location bay analysis of interictal spikes - A case study for the use of artificial neural networks in biomedical engineering"*, Techniques in Bioinformatics and Medical Informatics Annals of the New York Academy of Sciences, 980:306-313, 2002

[HCP94] J.F. Heagy, T.L. Carrol, L.M. Pecora, *"Synchronous chaos in coupled oscillatory systems"*, Phys. Rev. E, 50:1874-1885, 1994

[HKMa00] P.W. Hawkes, B. Kazan, T. Mulvey, *"Advances in imaging and electron physics"*, Academic Press Volume 114, San Diego, USA, 2000

[HKMb00] P.W. Hawkes, B. Kazan, T. Mulvey, *"Advances in imaging and electron physics"*, Academic Press Volume 109, San Diego, USA, 2000

[HKK95] S.K. Han, C. Kurrer, Y. Kuramoto, *"Dephasing and bursting in coupled neural oscillators"*, Phys. Rev. Lett., 75:3190-3193, 1995

[Hu1673] C. Hugenii, *"Horoloquim Oscilatorium"*, Paris, Frankreich, 1673

[KLT00] R. Kunz, A. Loncar, R. Tetzlaff, *"SCNN 2000, Part I and II"*, Proc. 6th Int. Workshop on CNNs and their Application (CNNA 2000), 407-419, Catania, 2000

[KP95] L. Kocarev, U. Parlitz, *"General approach for chaotic synchronization with applications to communication"*, Phys. Rev. Lett., 74:5028-5031, 1995

[KS97] H. Kantz, Th. Schreiber, *"Nonlinear Time Series Analysis"*, Cambridge Univ. Press, Cambridge, UK, 1997

[KTW96] R. Kunz, R. Tetzlaff, D. Wolf, *"SCNN: A universal simualtor for cellular neural networks"*, Proc. IEEE CNNA 96, Sevilla, 255-260, 1996

[KTW00] R. Kunz, R. Tetzlaff, D. Wolf *"Brain electrical activity in epilepsy: Characterization of the spatio-temporal Dynamics with cellular neural networks based on a correlation dimension analysis "*, Proc. ISCAS 2000, Genf, 1024-1027, 2000

[KT00] R. Kunz, R. Tetzlaff, *"Evolutionary learning strategies for cellular neural networks"*, Proc. of IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA 2000), 293-299, Catania, ISBN 07-7803-6344-2, 2000

[KTAGFLPSW00] R. Kunz, R. Tetzlaff, C. Ames, G. Geis, D. Feiden, A. Loncar, F. Puffer, R. Schoenmeyer, D.S. Weiß, *"SCNN 2000 - Dokumentation"*, Institut für Angewandte Physik der Johann Wolfgang Goethe-Universität Frankfurt am Main, 2000

[Ku96] R. Kunz, *"Simulation Zellularer Neuronaler Netzwerke"*, Diplomarbeit am Institut für Angewandte Physik der Johann Wolfgang Goethe-Universität Frankfurt am Main, 1996

[La01] T.N. Lal, *"Support Vector Machines: Theorie und Anwendung auf Prädiktion epileptischer Anfälle auf der Basis von EEG-Daten"*, Diplomarbeit am Institut für Angewandte Mathematik der Rheinischen Friedrich-Wilhelms Universität Bonn, 2001

[LAGE00] K. Lehnertz, J. Arhold, P. Grassberger, C.E. Elger, *"Chaos in Brain?"*, World Scientific, Singapur, 2000

[LE95] K. Lehnertz, C.E. Elger, *"Spatio-temporal dynamics of the primary epileptogenic are in temporal lobe epilepsy characterized by neuronal complexity loss"*, Electroencephal. clin. Neurophysiol., 95:108-117, 1995

[LE98] K. Lehnertz, C.E. Elger, *"Can epileptic seizures be predicted? Evidence from nonlinear time series analyses of brain electrical activity"*, Phys. Rev. Lett., 80:5019-5022, 1998

[LL02] B. Litt, K. Lehnertz, *"Seizure prediction and the preseizure period."*, Curr. Opin. Neurol., 15:173-177, 2002

[LLKLCLP00] J.S. Lee, D.S. Lee, S.K. Kim, S.K. Lee, J.K. Chung, M.C. Lee, K.S. Park, *"Localization of epileptogenic zones in F-18FDG brain PET of patients with temporal lobe epilepsy using artificial neural network"*, IEEE Trans. Med. Imag., 19(4):347-355, 2000

[LMKARDE03] K. Lehnertz, F. Mormann, T. Kreuz, R.G. Andrzejak, C. Rieke, P. David, C.E. Elger, *"Seizure prediction by nonliner EEG analysis"*, IEEE EMB Mag., 22:57-63, 2003

[LPKH02] M. Laiho, A. Paaiso, A. Kananen, K. Halonen, *"A mixed-mode polynomial-type CNN for analysing brain electrical activity in epilepsy"*, Int. J. Circ. Theor. Appl., 30:165-180, 2002

[LPKH04] M. Laiho, A. Paaiso, A. Kananen, K. Halonen, *"A mixed-mode polynomial-type cellular processor hardware realization"*, IEEE Trans. on Circuits and Systems, 51(2):286-297, 2004

[LSI04] B.L. Lu, J.H. Shin, M. Ichikawa, *"Massively parallel classification of single-trial EEG signals using a min-max modular neural network"*, IEEE Trans. Biomed. Eng., 51(3):551-558 , 2004

[LZY02] H.S. Liu, T. Zhang, F.S. Yang, *"A multistage, multimethod approach for automatic detevtion and classification of epileptiform EEG"*, IEEE Trans. Biomed. Eng., 49(2):1557-1566, 2004

[Ma72] K.V. Mardia, *"Probability and Mathematical Statistics: Statistics of Directional Data"*, Academy Press, London, UK, 1972

[MAKRDEL03] F. Mormann, R.G. Anrzejak, T. Kreuz, C. Rieke, P. David, C.E. Elger, K. Lehnertz, *"Automated detection of a pre-seizure state based on a decrease in synchronization in intracranial EEG recordings from epilepsy patients"*, Phys. Rev. E, 67:021912, 2003

[MAQBCRV98] J. Martinerie, C. Adam, M. Le Van Quyen, M. Baulac, S. Clemenceau, B. Renault, F.J. Varela, *"Epileptic seizures can be anticipated by non-linear analysis"*, Nature Medicine, 4:1173-1176, 1998

[MLDE00] F. Mormann, K. Lehnertz, P. David, C.E. Elger, *"Mean phase coherence as a measure for phase synchronization and its application to the EEG of epilepsy patients"*, Physica D, 144:358-369, 2000

[MKADLE03] F. Mormann, R.G. Andrzejak, P. David, K. Lehnertz,C.E. Elger, *"Epileptic seizures are preceded by a decrease in synchronization"*, Epilepsy Res., 53:173-185, 2003

[Mor98] F. Mormann, *"Synchronisationsphänomene in synthetischen Zeitreihen und Zeitreihen hirnelektrischer Aktivität"*, Diplomarbeit am Helmholz-Institut für Strahlen- und Kernphysik und in der Klinik für Epileptologie der Medizinischen Einrichtungen der Universität Bonn, 1998

[Mor03] F. Mormann, *"Synchronization phenomena in the human epileptic brain"*, Dissertation an der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn, 2003

[NG04] V.P. Nigam, D. Graupe, *"A neural-network-based detection of epilepsy"*, Neurol. Res., 26(1):55-60, 2004

[Ot93] E. Ott, *"Chaos in dynamical systems"*, Cambridge Univ. Press, Cambridge, UK, 1993

[PCJMH97] L.M. Pecora, T.L. Carrol, G.A. Jhonson, D.J. Mar, J.F. Heagy, *"Fundamentals of synchronization in chaotic systems, concepts and applications"*, Chaos, 7:520-543, 1997

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, *"Numerical Recipies in C, Second Edition"*, Cambridge Univ. Press, Cambridge, UK, 1992

[PRK01] A. Pikovsky, M.G. Rosenblum, J. Kurths, *"Synchronization: A universal concept in nonlinear sciences"*, Cambridge Univ. Press, Cambridge, UK, 2001

[Pu97] F. Puffer, *"Cellular neural networks with nonlinear weight functions - Applications to texture classification"*, Proc. European Conf. on Circuit Theory and Design, Budapest, 192-166, 1997

[PTW96] F. Puffer, R. Tetzlaff, D. Wolf, *"Modelling nonlinear systems with cellular neural networks"*, Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 96), Atlanta, 3513-3516, USA, 1996

[PTW95] F. Puffer, R. Tetzlaff, D. Wolf, *"Learning algorithm for cellular neural networks (CNN) solving nonlinear partial differential equations"*, Proc. Int. Symp. on Signals, Systems and Electronics (ISSSE 95), San Francisco, 501-504, USA, 1995

[PYW95] D.W. Peterman, M. Ye, P.E. Wigen, *"High frequency synchronization of chaos"*, Phys. Rev. Lett., 74:1740-1742, 1995

[PPHDW00] A. Petrosian, D. Prokhorov, R. Homan, R. Dasheiff, D. Wunsch, *"Recurrent neural network based prediction of epileptic seizures in intra- and extracranial EEG"*, Neurocomputing, 30:201-218, 2000

[QAG00] R. Quian Quiroga, J. Arnhold, P. Grassberger, *"Learning driver-response relationships from synchronization patterns"*, Phys. Rev. E, 61:5142-5148, 2000

[Ra99] I. Rauhut, *"Entwicklung einer Digitalschaltung zur schnellen Berechung von Korrelationssummen für nichtlineare Zeitreihenanalysen"*, Diplomarbeit am Helmholz-Institut für Strahlen- und Kernphysik und in der Klinik für Epileptologie der Medizinischen Einrichtungen der Universität Bonn, 1998

[RC93] T. Roska, L. O. Chua, *"The CNN Universal Machine: An analogic array computer"*, IEEE Trans. Circ. Syst.II: Analog and Digital Signal Processing, 40:163-173, 1993

[RK98] M.G. Rosenblum, J. Kurths, *"Analysing synchronization phenomena from bivariate data by means of the Hilbert Transform"*, in Nonlinear analysis of Physiological data, edited by H. Kantz et al., Springer Series for Synergetics, 1998

[RK99] T. Roska, L. Kék , *"CSL: CNN Software Library"*, Version 7.3, Budapest, 1999

[RPK96] M.G. Rosenblum, A.S. Pikovsky, J. Kurths, *"Phase synchronization of chaotic oscillators"*, Phys. Rev. Lett., 78:1804-1807, 1996

[RPK97] M.G. Rosenblum, A.S. Pikovsky, J. Kurths, *"From phase to lag synchronization in coupled chaotic oscillators"*, Phys. Rev. Lett., 78:4193-4196, 1997

[RR93] R. Rojas, *"Theorie der neuronalen Netze"*, Springer-Lehrbuch, Berlin 1993

[RT94] R. Roy, K.S.Jr. Thornburg, *"Experimental Synchronization of chaotic lasers"*, Phys. Rev. Lett., 72:2009-2012, 1994

[Sch02] R. Schönmeyer, *"Analyse und Optimierung von schaltungstechnischen Realisierungen Zellularer Neuronaler Netze zur nichtlinearen Informationsverarbeitung"*, Diplomarbeit am Institut für Angewandte Physik der Johann Wolfgang Goethe-Universität Frankfurt am Main, 2002

[Sch94] H.G. Schuster, *"Deterministisches Chaos - Eine Einführung"*, VCH Verlagsgesellschaft mbh, Weinheim, 1994

[SDEAS96] M. Schiek, F.R. Drepper, R. Engelbert, H.H. Abel, K. Suder, *"Cardiorespiatory synchronization"*, in Nonlinear analysis of Physiological data, edited by H. Kantz et al., Springer Series for Synergetics, 1996

[SRKA98] C. Schäfer, M.G. Rosenblum, J. Kurths, H.H. Abel, *"Heartbeat synchronized with ventilation"*, Nature, 392:239-240, 1998

[SWKD01] K. Schindler, R. Wiest, M. Kollar, F. Donati, *"Using simulated neural cell models for detection of epileptic seizures in foramen ovale and scalp EEG"*, Clin. Neurophysiol., 112:1006-1017, 2001

[SWKD02] K. Schindler, R. Wiest, M. Kollar, F. Donati, *"EEG analysis with simulated neuronal cell models helps to detect pre-seizure changes"*, Clin. Neurophysiol., 113:604-614, 2002

[TKAW99] R. Tetzlaff, R. Kunz, C. Ames, D. Wolf, *"Analysis of brain electical activity in epilepsy with cellular neural networks (CNN)"*, Proc. European Conf. on Circuit Theory and Design (ECCTD 99), Stresa, 1007-1010, Italien, 1999

[TKGW98] R. Tetzlaff, R. Kunz, G. Geis, D. Wolf, *"Minimizing the effects of tolerance faults on hardware realizations of cellular neural networks"*, Proc. CNNA 98, London, 254-258, UK, 1998

[TRWKPVSF98] P. Tass, M.G. Rosenblum, J. Weule, J. Kurths, A.S. Pikovsky, J. Volkmann, A. Schnitzler, H.J. Freund, *"Detection of n:m phase locking from noisy data: application to magentoencephalography"*, Phy. Rev. Lett., 81:3291-3294, 1998

[We03] D.S. Weiß, *"Nichtlineare Prädiktion hirnelektrischer Aktivität mit Zellularen Nichtlinearen Netzwerken bei Epilepsie"*, Diplomarbeit am Institut für Angewandte Physik der Johann Wolfgang Goethe-Universität Frankfurt am Main, 2001

[Ya99] X. Yao, *"Evolving artificial neural networks"*, Proc. IEEE, 87(9):14231447, 1999

[Zh95] Z. Zhang, *"Parameter estimation techniques: A tutorial with application to conic fitting"*, Rapport de recherche No. 2676, Institute National de Recherche en Informatique et en Automatique (INRIA) Sophia Antipolis, Frankreich, 1995

Eigene Veröffentlichungen

Buchartikel

[EMKARSFDL02] C.E. Elger, F. Mormann, T. Kreuz, R.G. Andrzejak, C. Rieke, R. Sowa, S. Florin, P. David, K. Lehnertz, *"Characterizing the spatio-temporal dynamics of the epileptogenic process with nonlinear EEG analysis"*, R. Tetzlaff (Hrsg.) Proceesings of the 7th IEEE International Workshop on Cellular Neural Networks and Their Applications, World Scientific, Singapur, 228-242, 2002

Konferenzbeiträge

[CSNTEL04] A. Chernihovskyi, R. Sowa, C. Niederhöfer, R. Tetzlaff, C.E. Elger, K. Lehnertz, *"Real-time seizure detection with cellular neural networks"*, eingereicht für American Epilepsy Society 58th Annual Meeting, New Orleans, Dezember 2004

[CSFEL04] A. Chernihovskyi, R. Sowa, S. Florin, C.E. Elger, K. Lehnertz, *"Pattern recognition in noisy and non-stationary time series with cellular neural networks"*, eingereicht und akzeptiert für 8th IEEE International Workshop on Cellular Neural Networks and Their Applications in Budapest, Ungarn, Juli 2004

[SMCFEL04] R. Sowa, F. Mormann, A. Chernihovskyi, S. Florin, C.E. Elger, K. Lehnertz, *"Estimating synchronization in brain electrical activity from epilepsy patients with cellular neuronal networks"*, eingereicht und akzeptiert für 8th IEEE International Workshop on Cellular Neural Networks and Their Applications in Budapest, Ungarn, Juli 2004

[SMCNTEL04] R. Sowa, F. Mormann, A. Chernihovskyi, C. Niederhöfer, R. Tetzlaff, C.E. Elger, K. Lehnertz, *"Seizure prediction: Measuring EEG phase synchronization with cellular neuronal networks"*, eingereicht für American Epilepsy Society 58th Annual Meeting, New Orleans, Dezember 2004