Einsatz neuronaler Netze

zur Verbesserung der

Dickentreffsicherheit in Walzwerken

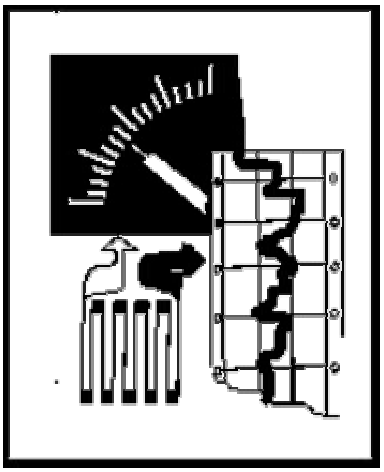
Von der Fakultät Maschinenbau

der Universität Dortmund

genehmigte Dissertation

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften



FACHGEBIET MESSTECHNIK

von

Carsten Wick

Witten

Tag der mündlichen Prüfung: 21.02.2003

Referent: Univ. Prof. Dr.-Ing. Mathias Uhle

Korreferent: Univ. Prof. Dr.-Ing. Matthias Kleiner

Danksagung

Diese Arbeit entstand während meiner Tätigkeit als freier Doktorand am Fachgebiet Messtechnik der Universität Dortmund. Allen, die hierzu beigetragen haben, gilt mein persönlicher Dank.

Mein besonderer Dank gilt Herrn Univ. Prof. Dr.-Ing. Mathias Uhle, dem Leiter des Fachgebietes Messtechnik, der sich großzügig bereit erklärte, das Referat für diese Arbeit zu übernehmen, trotz der Umstände, die mit der externen Anfertigung der Arbeit verbunden waren. Ich danke ihm für die damit verbundene Mühe und für die regelmäßigen, anregenden fachlichen Diskussionen und Hinweise, die für diese Arbeit richtungweisend waren.

Herrn Prof. Dr.-Ing. Matthias Kleiner, dem Leiter des Lehrstuhls Umformtechnik, danke ich für sein Interesse an der Arbeit und die Übernahme des Korreferats.

Allen Mitarbeitern des Fachgebietes Messtechnik danke ich für ihren Einsatz und für ihre Leistungen, die zum Gelingen der Arbeit mit beigetragen haben.

Mein Dank gilt auch der ThyssenKrupp Stahl AG, ohne deren Unterstützung die Durchführung der praktischen Versuche nicht möglich gewesen wäre. Stellvertretend für alle Mitarbeiter des Bochumer Warmbreitbandwerkes der ThyssenKrupp Stahl AG möchte ich Herrn Dr. Tamler und vor allem Herrn Dr.-Ing. Ingolf Jäckel danken.

Die Untersuchungen auf dem Gebiet der neuronalen Netze und der Prozessmodelle wurden in enger Zusammenarbeit mit der Siemens AG in Erlangen durchgeführt. Hier möchte ich vor allem Herrn Dr.-Ing. Michael Jansen und Herrn Dr.-Ing. Einar Broese danken.

Meinem Vater, Herrn Dipl.-Ing. Dietmar Wick, danke ich für die Anregung des Themas und für eine Vielzahl engagiert geführter Diskussionen und meiner Mutter danke ich für die Durchsicht des Textes.

Mein Dank gilt auch meinen Kolleginnen und Kollegen, welche mir viel Verständnis entgegenbrachten und es mir gestatteten, neben dem Alltagsgeschäft wissenschaftlich zu arbeiten.

Mein besonderer Dank gilt natürlich meiner Ehefrau Anke Wick für Geduld, Ruhe und Nachsicht und die gewissenhafte Durchsicht des Textes.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Allgemeine Formelzeichen

x	Skalar, Vektor
A	Matrix
x^T, A^T	Zeilenvektor, transponierte Matrix
x	Istwert
x^*	Messwert
$\hat{x}$	neuronal geschätzter Wert
$\tilde{x}$	berechneter Wert; Sollwert
$\bar{x}$	Mittelwert
$\|x\|$	Betrag; euklidsche Norm
Σ	Summenzeichen
Π	Produktzeichen
$\frac{\partial f(x)}{\partial x_i}$	partielle Ableitung von f nach x_i
∇	Nabla-Operator; entspricht $\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, ..., \frac{\partial}{\partial x_n}\right)^T$

Indizes

A	Austrittsebene im Walzspalt
AW	Arbeitswalze (j=1)
B	Band; Walzgut
BP	Backpropagation
C	Cluster
DN	Diagnosenetz
E	Eintrittsebene im Walzspalt
i,j	Allgemeine Indizes (z.B. für Neuronen)
i	Gerüstnummer
j	Index zur Identifikation der Walze: j=0 : Walzgut j=1 : Arbeitswalze j=2 : Stützwalze (4-Walzen) bzw. Zwischenwalze (6-Walzen) j=3 : Stützwalze (6-Walzen-Gerüst)
MM	Mathematisches Modell
NN	Neuronales Netz
p	Muster (Pattern)
SW	Stützwalze (hier: j=2)
W	Walze

WD	weight decay
WS	Walzspalt (j=0)
α_j, β_j	Linkes bzw. rechtes Lager der j´ten Walze

Häufig verwendete Symbole

x,y,z	Ortkoordinaten; der Koordinatenursprung liegt per Definition im Schwerpunkt des Walzgutes
a	Aktivierungszustand
A_d	Gedrückte Fläche
A_j	Querschnitt der j´ten Walze; $A_i=A_i(x)=\pi \cdot r_j^2$
b	Bandbreite
e	Residuum
E	Elastizitätsmodul (Mechanik)
E	Gesamtfehler (Neuronale Netze)
E_e	Externer Fehler
E_i	Interner Fehler
f_{act}	Aktivierungsfunktion
f_{out}	Ausgabefunktion
$f_{W,\alpha}$	Materialabhängige Korrektur der Walzkraft
$f_{W,\gamma}$	Gerüstabhängige Korrektur der Walzkraft
F_{um}	Umschaltpunkt der Walzkraft im Ständerdehnungsmodell
F_W	Walzkraft
$F_{W,MM}$	Mathematisches Modell für die Walzkraft
G	Gleitmodul der Walze (Hook´sches Gesetz $G=\frac{E}{2\cdot(1+m)}$)
h	Banddicke
h_G	Gaugemeterdicke (auch DHI)
h_m	Mittlere Banddicke im Walzspalt
Δh	Absolute Dickenänderung im Walzspalt
Δh_G	Gaugemeterfehler (auch DHG)
I_j	Axiales Flächenträgheitsmoment der j´ten Walze
l_d	Gedrückte Länge
p	Linienlast (z.B. im Walzspalt)
q	Relative Dickentreffsicherheit
Q	Absolute Dickentreffsicherheit
r	Relative Änderung
r_j	Radius der j´ten Walze
r_j'	Abgeplatteter Radius der j´ten Walze
s	Anstellung
s_0	Nullpunktanstellung
Δs_G	Anstellungsänderung durch Gerüstauffederung

Δs_L	Anstellungsänderung durch Lageraufschwimmen (auch DFB3)
Δs_{Rest}	Adaptive Nullpunkkorrektur (auch DFBI)
Δs_W	Anstellungsänderung durch Walzenmodelle
S	Sensitivität
t	Teaching input
t_B	Berührzeit
T	Bandtemperatur (meist θ)
v	(Walzgut-)Geschwindigkeit
v_u	Walzenumfangsgeschwindigkeit
v_{rel}	Relativgeschwindigkeit (zwischen Walze und Walzgut)
w_j	Abplattung der j´ten Walze
$w_{i,j}$	Gewicht zwischen zwei Neuronen i und j
Δw	Gewichtsänderung
x_N	Fließscheidenlage
z_C	Clusterzentrum
α	Walzwinkel
α_0	Greifwinkel
α	Trägheitsrate bei neuronalen Netzen („Momentum“)
δ	Fehler für die „Delta“-Regel
ε	Fehler
ε_i	Relative Stichabnahme im i´ten Walzgerüst
Φ	Volumenstrom
η	Lernrate
φ	Umformgrad
$\dot{j}$	Formänderungsgeschwindigkeit
κ_A	Bezogene Voreilung
κ_E	Bezogene Nacheilung
λ	Korrekturfaktor bei neuronalen Netzen
μ	Mittelwert (Statistik)
μ	Reibwert (Mechanik)
θ	(Band-)Termperatur (auch T)
$\Delta\theta$	Temperaturänderung
$\Delta\theta_L$	Temperaturänderung durch Wärmeleitungsverluste
$\Delta\theta_S$	Temperaturänderung durch Strahlung
$\Delta\theta_u$	Temperaturänderung durch Umformarbeit
$\Delta\theta_Z$	Temperaturänderung durch Zunderabspritzung
$\Delta\theta_\mu$	Temperaturänderung durch Reibung
θ_i	Schwellenwert (Bias) des i´ten Neurons
σ	Standardabweichung

Abkürzungen

AGC	Automatic Gaugemeter Control (Dickenregelung)
AKM	Advanced k-Means
DB	Datenbank
DBMS	Datenbankmanagementsystem
DD	Data Dictionary
GRENN	Gaugemeter Residual Error Neural Network
IDL	Interface Definition Language
KNN	Künstliches neuronales Netz
LMS	Least Mean Square (Kleinstes mittleres Quadrat)
LVQ	Lernende Vektorquantisierung
MAPE	Mittlerer absoluter prozentualer Fehler
MEWE	Messwerterfassung
MLP	Multilayer-Perceptron
MRE	Mittlerer relativer Fehler
NN	Neuronales Netz
ODBC	Open Database Connectivity
OOA	Objektorientierte Analyse
OOD	Objektorientiertes Design
PCA	Principal Component Analysis
PDB	Prozessdatenbank
PDS	Prozessdatenbank-Server
PRFSON	Prozessrechner Fertigstraße Online
PRFSSB	Prozessrechner Fertigstraße Standby
PSC	Vorausberechnung
RBF	Radialbasis-Funktionen Netzwerk
RMA	Nachberechnung (Rolling Model Adaption)
RMS	Wurzel des mittleren quadratischen Fehlers
RMSE	Root Mean Square Error
SOM	Self Organizing Maps
SSE	Summe der quadratischen Residuen
SVA	Singular Value Decomposition
UML	Unified Modeling Language
WBB	Warmbreitbandstraße
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformations

1 Einleitung

1.1 Motivation

Das Walzen von Stahl besitzt auch heute noch eine hohe volkswirtschaftliche Bedeutung. Der ThyssenKrupp-Konzern produzierte beispielsweise im Jahr 2001 alleine in seinen vier Warmbreitbandwerken im Ruhrgebiet insgesamt etwa 15 Millionen Tonnen Warmband im Gesamtwert von fast 5 Milliarden Euro. Die Steigerung der Qualität und der Produktivität beim Warmwalzen ist demnach eine nach wie vor wichtige und auch ökonomisch interessante Aufgabe.

Beim Warmwalzen von Breitband ist das Treffen einer vorgegebenen Dicke eines der wichtigsten, wenn nicht das wichtigste Qualitätsmerkmal. Mit Hilfe moderner Regelungstechnik ist es möglich, eine sehr hohe Genauigkeit der Dicke im Mittelsegment des Bandes zu erreichen. Die Qualität am Anfang des Bandes hängt sehr stark von einer guten Prognose durch die Prozessautomatisierung ab. Diese setzt unterschiedlichste Verfahren ein, um die vielfältigen technischen Vorgänge beim Walzen zu beschreiben, dennoch ist es bisher gerade bei Anlagen mit einem sehr breiten Werkstoffspektrum nur eingeschränkt möglich, eine optimale Voreinstellung der Stellglieder zu erzielen.

In der Vergangenheit bestand die Prozessautomatisierung meistens aus einfachen physikalischen, meist linearen Modellen, die um statistische Korrekturen und Adaptionen ergänzt wurden. Moderne Prozessautomatisierungen beschreiben die Vorgänge im Walzspalt mit verschiedenen mathematisch-physikalischen Teilmodellen, die eine deutlich höhere Genauigkeit besitzen. Allerdings erhöht sich der Aufwand bei der Superposition der Modelle und unabhängig von der Art der Prozessautomatisierung muss mit langen Inbetriebsetzungsphasen gerechnet werden, die hohe Kosten verursachen.

Ziel der Prozessautomatisierung ist es, eine optimale Vorausberechung aller Stellgrößen der Anlage zu erzielen. Die Qualität der Prozessautomatisierung zeigt sich dabei einerseits in ihrer Generalisierungsleistung für neue Werkstoffe und andererseits in der Diagnostizierbarkeit von Fehlern. Einfache lineare Modelle lassen kaum Rückschlüsse bei fehlerhafter Vorausberechnung zu, aber auch komplexe Prozessmodelle können Fehler nur schwer diagnostizieren, da diese häufig in einer heterogenen Systemumgebung verschaltet sind. Erschwerend kommt in den meisten Fällen hinzu, dass die für die Analysen benötigten Daten oft nur in unzureichender Form vorliegen.

1.2 Zielsetzung

Die Dickentreffsicherheit beim Warmwalzen beschreibt die erreichte Toleranz des Dickenfehlers aller gewalzten Bänder. Ziel dieser Arbeit ist es, diese Dickentreffsicherheit speziell im Kopfbereich der Bänder zu optimieren. Dieser Bereich unterliegt noch nicht der Regelung, sondern wird nur durch die Voreinstellung der Anlage beeinflusst. Daher werden Methoden beschrieben, mit denen die für die Voreinstellung verantwortliche Prozessautomatisierung einer Walzstraße optimiert werden kann.

Diese Optimierung geschieht durch Schaffung einer homogenen Softwareumgebung. Durch die Definition eines Entwicklungsrahmens für Prozessmodelle und deren Schnittstellen werden Reibungsverluste zwischen Modellen vermieden und die Diagnostizierbarkeit einzelner Fehler stark verbessert. Die Einbeziehung der Datenhaltung führt zudem dazu, dass alle wesentlichen Prozessgrößen verwaltet werden können.

Die Diagnostizierbarkeit des Dickenfehlers und anderer Prozessgrößen erlaubt eine einfachere Inbetriebsetzung und Wartung der Prozessautomatisierung, die darüber hinaus eine Ferninbetriebsetzung und Fernwartung gestattet.

Der Entwicklungsrahmen wird dabei zweistufig aufgebaut: Die abstraktere, erste Stufe stellt einen Entwicklungsrahmen für beliebige Prozessmodelle dar, der sich vor allem durch die Möglichkeit der Beschreibung adaptiver Modelle auszeichnet - dies beinhaltet die Simulation neuronaler Netze; die zweite Stufe beschreibt die Anpassung an spezielle Anforderungen in einem Walzwerk.

Neben dem theoretischen Ansatz dieses Entwicklungsrahmens ist es Ziel der Arbeit, die Ergebnisse anhand einer realen Walzstraße zu validieren. Die für praktische Experimente betrachtete Warmbreitbandstraße wurde während des Untersuchungszeitraums von einer alten auf eine moderne Prozessautomatisierung umgestellt, so dass interessante Vergleiche zwischen beiden Verfahren gezogen werden können.

Die Verbesserung einzelner Prozessmodelle durch Optimierung der physikalischen Modellbildung ist kein unmittelbares Ziel dieser Arbeit. Diese Arbeit geht vielmehr von der Annahme aus, dass die bestehenden, im folgenden Kapitel beschriebenen analytischen Prozessmodelle hinreichend genau sind, um einzelne Bänder „offline“ nachzurechnen. Für den Echtzeiteinsatz müssen für einzelne Modelle aufgrund von Rechenzeitbeschränkungen Näherungslösungen verwendet werden. Die hierbei auftretenden Modellrestfehler werden in dieser Arbeit ebenso wie alle weiteren Fehler nur durch eine Verbesserung von adaptiven und empirischen Teilmodellen kompensiert. Eine wichtige Methode stellen in diesem Zusammenhang neuronale Netze dar.

1.3 Einordnung der Arbeit

Aufgrund der Vielzahl der angesprochenen Fachgebiete wird hier nur ein kurzer Überblick über den Stand der Technik gegeben, der im nachfolgenden Kapitel detailliert wird. Die wichtigsten Grundlagen der verschiedenen Fachgebiete sind im Anhang dargestellt.

Zur Beschreibung der Vorgänge beim Walzen existieren eine Reihe komplexer Modelle, die jedoch teilweise aufgrund hoher Rechenzeiten noch nicht in der Produktion eingesetzt werden können. Die komplexesten Modelle, die in der Praxis eingesetzt werden, besitzen jedoch auch heute schon eine Genauigkeit, die in der Größenordnung der Messungenauigkeit liegt. Ein komplettes homogenes Modell, das alle Vorgänge beschreibt, ist allerdings nicht vorhanden.

In den letzten Jahren wurden vielfältige Experimente mit neuronalen Netzen durchgeführt, die das Ziel hatten, die analytischen Modelle noch weiter zu optimieren oder sogar zu ersetzen. Die Theorie der neuronalen Netze wurde in den letzten 15 Jahren umfassend erforscht, und die softwaretechnische Simulation neuronaler Netze hat im Zuge immer schnellerer Rechner vor allem in den letzten 10 Jahren einen enormen Fortschritt erlebt. Es existieren eine ganze Reihe von Simulatoren für neuronale Netze, von denen die wenigsten jedoch eine direkte Integration in die analytische Modellierung bieten. Auch die Möglichkeit der Online-Adaption, die gerade die Prozessautomatisierung einer Walzstraße erfordert, ist bei vielen marktüblichen Simulatoren nicht vorhanden.

Der Einsatz neuronaler Netze in der Walzwerkstechnik hat sich in den letzten Jahren durchgesetzt, um Prozesse zu beschreiben, die sich analytisch nicht vollständig lösen lassen. Zu diesem Thema existieren eine Vielzahl an Veröffentlichungen (z.B. [BulHoc1996], [Broese1997], [Lindhoff1994], [Martinez1996], [PetPlü1994] und [Pleier1995]). Dabei wurden vor allem folgende Netze entwickelt:

- Berechnung der Walzkraft
- Berechnung der Bandtemperatur
- Prognose der natürlichen Breitung in der Fertigstraße
- Steuerung der Staucheranstellung zur Optimierung der Bandendenform (SSC)

Netze zur Berechnung des Anstellungsnullpunktes existieren zurzeit nicht, da diese sehr walzwerksspezifisch sind und die Einführung eine eingehende Fehleruntersuchung der jeweiligen Anlage erfordert.

1.4 Aufbau der Arbeit

Im folgenden zweiten Kapitel wird der Stand der Technik, auf dem diese Arbeit aufbaut, ausführlich erläutert. Dabei werden vor allem die bekannten Möglichkeiten

zur Modellierung der Vorgänge beim Warmwalzen von Stahl vorgestellt; außerdem werden die wichtigsten neuronalen Netze, die sich zum Einsatz in der Prozessautomatisierung von Walzwerken eignen, erläutert. Das Kapitel endet mit einem kurzen Überblick über moderne Methoden der Softwaretechnik.

Im dritten Kapitel setzen die eigenen Überlegungen mit der Beschreibung der gewählten Methodik ein. Zunächst wird ein softwaretechnischer Entwicklungsrahmen für allgemeine Prozessmodelle beschrieben; anschließend wird die Erweiterung dieses Entwicklungsrahmens für Walzwerke vorgestellt.

Im vierten Kapitel werden die Ergebnisse der praktischen Datenakquisition und Datenaufbereitung dargestellt. Dazu wird vorab eine eigens für diese aufwendigen Experimente entwickelte Methodik beschrieben.

Im fünften Kapitel werden verschiedene Verfahren, die zur Optimierung der Dickentreffsicherheit von warmgewalzten Bändern mit neuronalen Netzen im Rahmen dieser Arbeit entwickelt wurden, vorgestellt. Dabei werden die Optimierung einer Prozessautomatisierung mit konventionellen Prozessmodellen und Adaptionen und die Optimierung einer modernen Variante unterschieden.

Im sechsten Kapitel werden – motiviert durch die im fünften Kapitel beschriebenen Ergebnisse – weitere Einsatzmöglichkeiten von neuronalen Netzen in der Prozessautomatisierung zu Diagnosezwecken diskutiert. Weiterhin werden erweiterte Möglichkeiten des in Kapitel 3 beschriebenen Entwicklungsrahmens in Bezug auf Data Mining, Ferndiagnose und Ferninbetriebsetzung herausgearbeitet.

Im abschließenden siebten Kapitel werden die wichtigsten Ergebnisse dieser Arbeit noch einmal zusammengefasst.

Der Anhang enthält die wichtigsten Grundlagen der Umformtechnik, der Softwaretechnik und der Grundlagen neuronaler Netze, wie sie im Rahmen dieser Arbeit Verwendung finden, und legt eine einheitliche Nomenklatur fest, deren Ergebnis sich auch im Symbolverzeichnis widerspiegelt.

2 Stand der Technik

2.1 Einleitung

Im folgenden Kapitel wird der Stand der Technik erläutert. Da die Arbeit mehrere Fachrichtungen berührt, wird sowohl ein kurzer Abriss über die Modellierung von Walzmodellen, die Simulation neuronaler Netze und die wichtigsten Grundlagen moderner Softwareentwicklung gegeben.

Es sei an dieser Stelle ausdrücklich darauf hingewiesen, dass diese Abschnitte nicht den Anspruch auf Vollständigkeit erheben, sondern nur eine einheitliche Nomenklatur für die anschließenden Kapitel definieren. Für weitergehende Informationen wird an geeigneter Stelle auf entsprechende Literatur verwiesen. Die wichtigsten Grundlagen der einzelnen Fachgebiete sind zudem im Anhang angegeben.

2.2 Modellierung der Vorgänge im Walzspalt

2.2.1 Vorbemerkungen

Der im Stahlwerk erschmolzene Stahl besitzt noch nicht die Form, Abmessungen und Eigenschaften, die vom Verbraucher gefordert werden. Um diese zu erhalten, wird er weiterverarbeitet; zu diesen Arten der Weiterverarbeitung zählt das Walzen.

Das Walzen von Stahl stellt nach wie vor ein Verfahren dar, das grundlegende Bedeutung besitzt. Die hohe Wertschöpfung existierender Warmbreitstraßen verspricht beim Einsatz moderner Methoden ein hohes Innovationspotential mit hoher Wirtschaftlichkeit.

In den nächsten Abschnitten werden die Grundlagen der wichtigsten Prozessmodelle einer Fertigstraße eines Warmbreitbandwerkes beschrieben. Weiterführende Informationen entnimmt man z.B. [LipMah1967], [Schwenzfeier1979], [Lange1988] oder [DahKop1993]. Im Anhang werden außerdem die wichtigsten umformtechnischen Grundbegriffe definiert, außerdem wird dort exemplarisch das Warmbreitbandwerk eines großen europäischen Stahlproduzenten vorgestellt, auf dem die praktischen Versuche in dieser Arbeit durchgeführt wurden.

2.2.2 Prozessautomatisierung der Fertigstraße

Die Prozessautomatisierung eines Walzwerkes teilt sich üblicherweise in mindestens drei Ebenen auf. Die oberste technische Ebene (Level 3), die so genannte Prozessleitebene, steuert dabei die Abfolge der gewalzten Stücke und gibt Sollwerte und Primärdaten der Vorverarbeitung weiter. Die Prozessleitebene besitzt zusätzlich eine

Schnittstelle zu Warenwirtschaftssystemen, die manchmal mit zur Prozessautomatisierung hinzugezählt werden (Level 4). Die Prozessführungsebene (Level 2) bestimmt aus den Parametern, die sie von der Prozessleitebene übergeben bekommt, die Voreinstellungen der Basisautomatisierung (Level 1), die alle Stellglieder (Level 0) direkt ansteuert.

In dieser Arbeit wird nur die Prozessführungsebene der Fertigstraße betrachtet; daher wird im Folgenden meist von der Prozessführungsebene und der dazu korrespondierenden Prozessautomatisierung gesprochen.

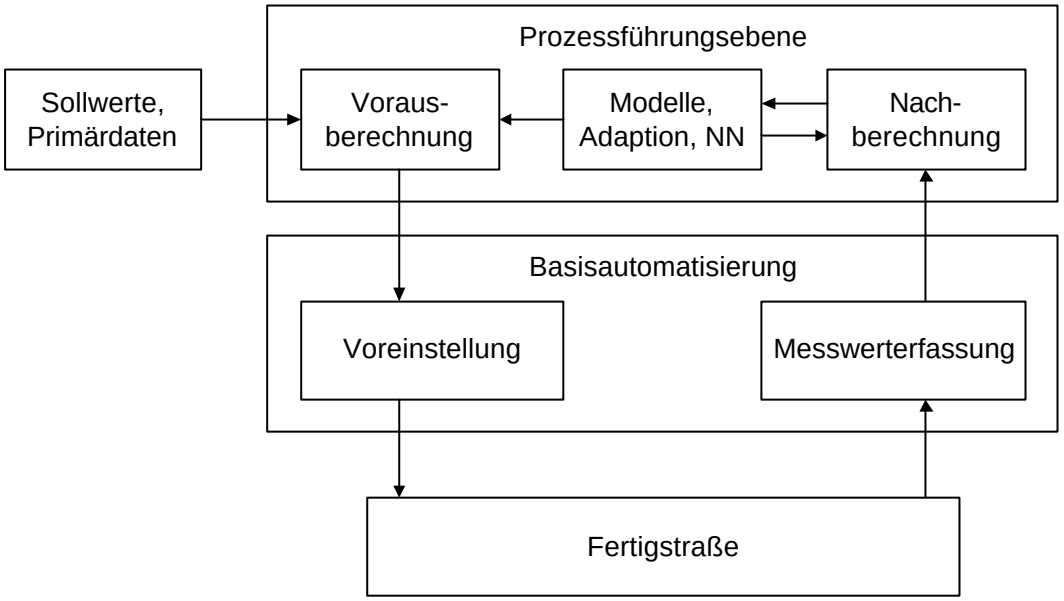


Abbildung 2.1: Prozessautomatisierung einer Fertigstraße

Die Prozessführungsebene setzt sich üblicherweise aus den beiden Komponenten *Stichplan-Vorausberechnung*, oder kurz *Vorausberechnung*, und *Nachberechnung* zusammen. Mit Hilfe verschiedener Prozessmodelle bestimmt die Vorausberechnung eine optimale Voreinstellung für die Basisautomatisierung. Da diese Prozessmodelle, wie sich später noch zeigen wird, meistens adaptiv ausgelegt sind, wird nach jedem gewalzten Band eine Nachberechnung durchgeführt, die alle adaptiven Parameter der Modelle anpasst. Zu diesen adaptiven Parametern gehören auch die Gewichte der verwendeten neuronalen Netze. Dies wird in Abbildung 2.1 verdeutlicht.

2.2.3 Prozessmodelle der Fertigstraße

In der Stichplan-Vorausberechnung wird mit Hilfe verschiedener Prozessmodelle die Geometrie des Walzspaltes in der Austrittsebene bestimmt. Diese verschiedenen Prozessmodelle lassen sich in zwei Gruppen aufteilen.

Als *Basismodelle* werden diejenigen Modelle bezeichnet, die keine direkten Terme für das Walzspaltmodell, sondern Zwischengrößen liefern, die andere Modelle

benötigen. Zu diesen Modellen gehören das Walzkraftmodell und das Bandtemperaturmodell; dieser Gruppe können außerdem das Voreilungsmodell und das Breitungsmodell zugeordnet werden.

Die *Walzspaltmodelle* liefern Ergebnisse, die direkt in das Modell zur Bestimmung der Anstellung eingehen; sie gliedern sich wiederum in zwei Gruppen: Gerüstmodelle und Walzenmodelle. Die *Gerüstmodelle* beschreiben den Einfluss des Walzenständers auf den Walzspalt. Diese Modelle wirken sich nur auf die Walzspalthöhe, nicht jedoch auf das Walzspaltprofil, aus. Das Ständerdehnungsmodell und das Modell zur Bestimmung des Lageraufschwimmens gehören zu dieser Gruppe. Die *Walzenmodelle* andererseits beeinflussen neben der Walzspalthöhe auch das Walzspaltprofil. Sie beschreiben die Änderung der Arbeitswalzendurchmesser. Die drei wichtigsten Walzenmodelle sind das Walzendeformations-, das Walzentemperatur- und das Walzenverschleißmodell.

Algorithmisch lassen sich Prozessmodelle in drei Varianten einteilen. *Analytische Prozessmodelle* lösen ein physikalisches Problem durch explizite Lösung analytischer, physikalischer Modellgleichungen. *Statistische Modelle* hingegen stellen sich einer vorgegebenen Herausforderung alleine durch empirische Betrachtungen. Zur Gruppe der statistischen Modelle gehören auch neuronale Netze. Die Verknüpfung analytischer und statistischer Modelle wird *hybride Modellierung* genannt.

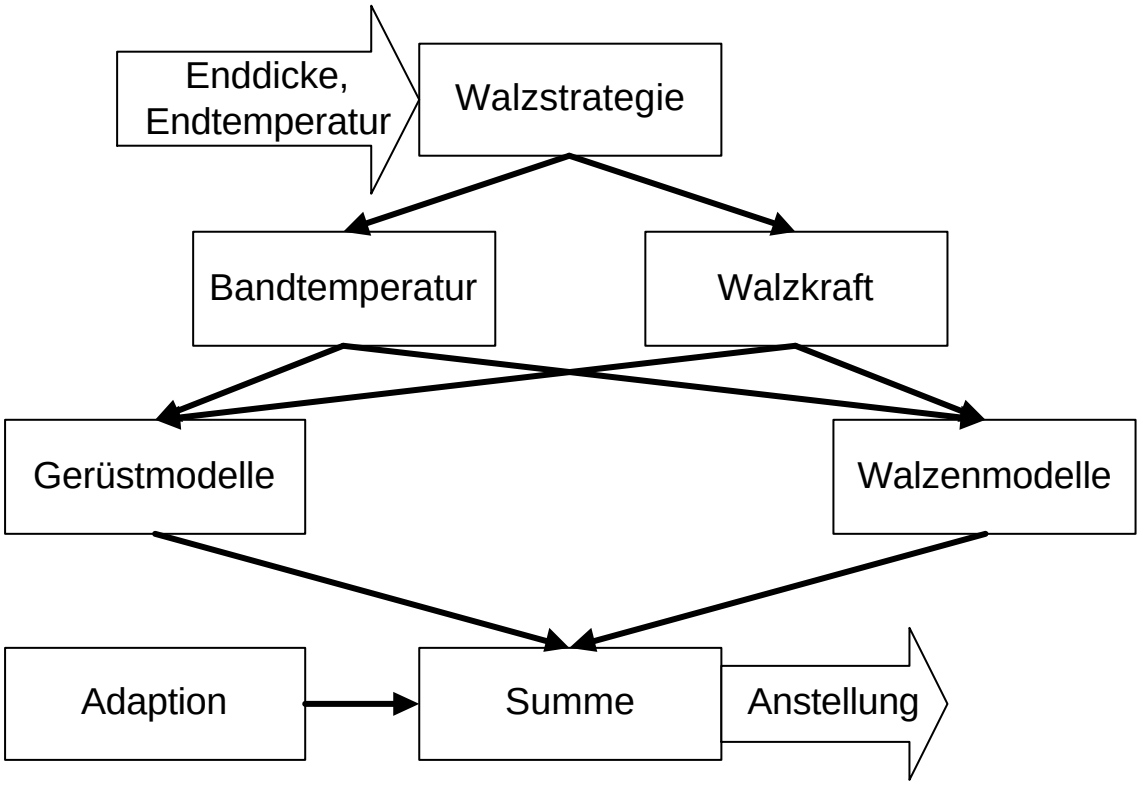


Abbildung 2.2: Beispiel einer Kombination von Teilmodellen

Abbildung 2.2 zeigt exemplarisch das komplexe Zusammenwirken der verschiedenen Prozessmodelle. Um eine gewünschte Endwalzdicke und Endwalztemperatur zu erzielen, werden mit Hilfe verschiedener Teilmodelle die Anstellungen in den Walzgerüsten bestimmt. Nachfolgend werden die wichtigsten Prozessmodelle erläutert; dabei wird jeweils kurz der physikalische Effekt beschrieben und ein Hinweis auf die Modellbildung gegeben. Auf eine konkrete Beschreibung der Lösung wird bewusst verzichtet.

2.2.4 Basismodelle

Mit Hilfe der Stichplan-Vorausberechnung werden die Stichabnahmen, die Walzkräfte und die Walzgeschwindigkeiten in der Fertigstraße derart bestimmt, dass aus der Einlaufdicke und der Einlauftemperatur eine gewünschte Enddicke und Auslauftemperatur erzielt werden.

Zunächst werden die einzelnen Stichabnahmen für die gesamte Straße und anschließend nacheinander für alle Gerüste Einlaufgeschwindigkeiten, Walzkräfte und Auslaufgeschwindigkeiten bestimmt. Da die Bestimmung der Walzkraft von der Einlaufgeschwindigkeit abhängt und die Bestimmung der Auslaufgeschwindigkeit von der Walzkraft, müssen alle Gerüste nacheinander gerechnet werden. Bei Verletzung einer vorgegebenen Gültigkeitsgrenze muss eine so genannte *Umverteilung* durchgeführt und die Stichplanberechnung wiederholt werden. Dies geschieht solange, bis ein gültiger Stichplan vorliegt.

Nach der Berechnung des Stichplans wird über das Voreilungsmodell die Umdrehungszahl der Walzen bestimmt. Das Breitungsmodell bestimmt die zu erwartende Breitung des Bandes in der Fertigstraße.

2.2.4.1 Bestimmung der Stichabnahmen

Die Stichabnahmen werden von der Walzstrategie festgelegt. Die Aufgabe der Walzstrategie besteht in der Aufbereitung der Vorbanddaten und der Zielgrößen, in der Verwaltung der material- und gerüstspezifischen Vererbungskoeffizienten und in der Festlegung der Abnahmevorschrift.

Die Abnahmevorschrift ist entsprechend technischer Randbedingungen im Prozessrechner hinterlegt. Die Walzstrategie ermittelt aufgrund der geforderten Bandabmessungen und der Materialqualität die zugehörige Abnahmeverteilung und übermittelt sie der Stichplanberechnung [Siemens1997].

2.2.4.2 Bestimmung der Walzkraft

Derzeit in der Praxis verwendete Walzkraft-Modelle sind in einem Überblick in [Hinkfoth1989] detailliert beschrieben. Am weitesten verbreitet ist danach das auf Siebel (1924) und Karman (1925) zurückgehende so genannte *Streifenmodell*. Das Material im Walzspalt wird hierbei gedanklich in senkrechte Streifen aufgeteilt, wobei für jeden Streifen einzeln der Spannungszustand bestimmt wird. Eine exakte Lösung ist nur mit numerischen Verfahren möglich. Eine verbesserte Variante arbeitet statt mit senkrechten mit gekrümmten Streifen ([Orowan1943]).

Das in der Praxis häufig verwendete Modell von Sims ([Sims1952]) stellt ähnlich wie das Modell von Lippmann und Mahrenholtz ([LipMah1967]) eine Näherungslösung des Streifenmodells dar. Es weist jedoch laut [Hinkforth1989] gegenüber letzterem eine Reihe von Mängeln auf. Der Vorteil dieser Modelle besteht darin, dass sie theoretisch fundiert sind und ohne freie Parameter auskommen.

Eine Alternative stellt die Walzkraftformel nach Carlton, Edwards und Thomas dar ([CaEdTh1992]). Sie basiert auf einer mittleren ebenen Fließspannung. Wenn man die Formel auf das Warmwalzen überträgt, ergibt sich eine Gleichung mit fünf freien Parametern. Da diese Methode mit der expliziten Hitchcock-Gleichung (A.8) arbeitet, muss die Walzkraft iterativ bestimmt werden.

Mit Hilfe von FEM-Programmen kann der Walzspalt modelliert und die Walzkraft berechnet werden. Diese Methode erzielt die höchste Genauigkeit, ist aber für den Einsatz auf Prozessrechnern zu speicher- und rechenintensiv.

Die zurzeit in der betrachteten Fertigstraße eingesetzte Walzkraftformel geht auf Arbeiten von Cook und McCrum [CooMcC1958] zurück. Sie besitzt folgende Form:

$$F_{W,i} = f_{W,a} \cdot F_{W,g_i} \cdot F_{W,MM,i}(e,...) \qquad (2.1)$$

$F_{W,MM}$ stellt einen Ausdruck dar, der die Abhängigkeit der Walzkraft von relativer Abnahme ε, Breite, Temperatur, Zugspannungen, Einlaufdicke und Arbeitswalzenradius angibt.

Der materialabhängige Anteil der Materialhärte wird durch $f_{W,\alpha}$ modelliert, dabei wird $f_{W,\alpha}$ mit einem globalen neuronalen Netz bestimmt. Dieses Netz besitzt als Eingangsgrößen die komplette chemische Analyse und wird nach der Initialisierung nur in Sonderfällen nachtrainiert und nicht online-adaptiert.

Zur Bestimmung des gerüstabhängigen Korrekturfaktors $f_{W,\gamma}$ steht pro Gerüst ein neuronales Netz zur Verfügung, das neben dem Vektor der Legierungsanteile des zu walzenden Bandes auch einige weitere Werte als Inputs erhält. Dazu zählen die Vorbandbreite, die Eintrittsdicke und die Eintrittstemperatur, der Rückwärts- und

der Vorwärtszug, die relative Stichabnahme, der Walzenradius und die Walzenumfangsgeschwindigkeit. Die $f_{W,\gamma}$-Netze werden im Gegensatz zum $f_{W,\alpha}$–Netz online adaptiert.

2.2.4.3 Berechnung der Walzguttemperatur

Zwischen dem Eintritt in zwei aufeinanderfolgende Gerüste erfährt das Band eine Temperaturerhöhung durch:

- Umformarbeit im Walzspalt
- Reibung zwischen Walzgut und Walzen im Kontaktbogen

Andererseits erfährt es eine Temperaturabnahme durch:

- Wärmeleitungsverluste im Walzspalt an die Arbeitswalze
- Wärmeverluste durch Zunderabspritzung
- Wärmeübertragung infolge von Strahlung

Die Wärmeverluste durch Walzenkühlwasser, Konvektion an die Umgebungsluft und durch Wärmeleitung an die Schlingenheberrollen und den Rollgang können vernachlässigt werden.

Insgesamt ändert sich die Temperatur zwischen zwei Gerüsten:

$$\Delta q = \Delta q_U + \Delta q_m - \Delta q_L - \Delta q_Z - \Delta q_S \qquad (2.2)$$

Die einzelnen Größen lassen sich mit in der Literatur beschriebenen, expliziten Modellgleichungen direkt bestimmen ([Pawelski1969], [Bruns1974], [Harms1975] und [Langer1976]).

Durch sukzessives Bestimmen dieser Größen für jedes Gerüst lässt sich aus der Temperatur vor dem ersten Gerüst die Endwalztemperatur ermitteln. In der Realität wird dieser Mechanismus um ein neuronales Netz ergänzt, das verschiedene Bandparameter erhält und mit dem der zu erwartende Temperaturfehler vorausgesagt werden kann, der dann gleichmäßig auf alle Gerüste verteilt wird.

Konkret handelt es sich bei den Eingangsgrößen um den Anteil der Legierungselemente C, Si, Mn, P und S, die Vorbanddicke, die Vorbandbreite, die Fertigbanddicke, die Vorbandtemperatur, die Fertigbandtemperatur, die Laufzeit des Bandes zwischen Vorstraße und Fertigstraße und die relativen Abnahmen in den 7 Gerüsten.

Da es sich bei der Endwalztemperatur um eine Zielgröße handelt, wird eine Größe benötigt, die frei veränderlich ist und einen großen Einfluss auf die Temperatur hat. Die Bandgeschwindigkeit scheint dieser Anforderung am besten zu genügen, daher

wird im Temperaturmodell die Bandgeschwindigkeit derart bestimmt, dass sich eine vorgegebene Endwalztemperatur einstellt.

2.2.4.4 Berechnung der Voreilung

Die Bestimmung der Voreilung ist, wie in Anhang A.5 näher beschrieben, aufgrund fehlender Messmöglichkeiten zur Berechnung der Bandgeschwindigkeit aus der Winkelgeschwindigkeit der Walzen sehr wichtig. Das in der betrachteten Anlage eingesetzte Voreilungsmodell vernachlässigt Zugspannungen und nutzt eine der vorgestellten Variante ähnliche Lösung. Der Unterschied besteht im Wesentlichen in einer um einige Glieder weiterentwickelten Taylor-Reihe.

2.2.4.5 Breitung in der Fertigstraße

Zur Beschreibung der Breitung in der Fertigstraße existieren vielfältige analytische Ansätze, die auch in einem Überblick in [Mauk1976] beschrieben werden. In der betrachteten Anlage wird jedoch anstelle eines analytischen Modells ein reines, neuronales Netz eingesetzt, dessen Eingangsgrößen fast denen des Temperaturkorrekturnetzes entsprechen.

Konkret handelt es sich bei den Eingangsgrößen des Breitungsnetzes um den Anteil der Legierungselemente C, Si, Mn, Ti, V und Nb des Bandes, den Ausgang des α-Netzes, die Vorbanddicke, die Vorbandbreite, die Fertigbanddicke, die Vorbandtemperatur, die Fertigbandtemperatur, die Austrittsgeschwindigkeit aus dem letzten Gerüst, den Rückwärtszug im letzten Gerüst und die relativen Abnahmen in den 7 Gerüsten.

2.2.5 Gerüstmodelle

Die Basismodelle liefern - neben den Abnahmen in den einzelnen Gerüsten - die Walzkräfte, die Bandtemperaturen und die Walzenumfangsgeschwindigkeiten, mit denen dann die Walzspaltgeometrie bestimmt werden kann. Dazu werden weitere Modelle verwendet, die wiederum miteinander verknüpft sind. In diesem Kapitel werden zunächst nur die Verformungen des Walzgerüstes betrachtet, die zu einem Auffedern des Walzspaltes führen.

2.2.5.1 Elastische Auffederung des Walzenständers

Der Zusammenhang zwischen der Walzkraft und der Auffederung des Walzspaltes wird durch die so genannte *Gerüstkennlinie* beschrieben. Die Steigung dieser Kennlinie kennzeichnet den *Gerüstmodul* (Federsteifigkeit).

Der Gerüstmodul des Walzgerüstes ergibt sich aus den Verformungen der Walzenständer, der Anstellung, der Einbaustücke, der Walzenlager und der Stütz- und Arbeitswalzen. Die Verformung des Ständers ist in [Berger1976] dargestellt. Eine genauere Betrachtung ist [Schwarzer1971] zu entnehmen. Die Stauchung der Teile des mechanischen Anstellsystems (Spindel, Druckmutter, Spindeldrucklager und Einbaustücke) wird nach dem Hookschen Gesetz bestimmt:

$$\Delta l = \frac{F \cdot l}{E \cdot A} \qquad (2.3)$$

Für hydraulische Anstellsysteme lässt sich keine allgemeingültige Berechnungsvorschrift angeben.

In der Praxis werden die Formeln aus der Literatur selten genutzt, stattdessen wird die Gerüstkennlinie direkt aufgenommen. In der betrachteten Fertigstraße wird die die durch die Ständerdehnung verursachte Auffederung des Walzspaltes durch folgende Gleichung beschrieben:

$$\Delta s_G = \begin{cases} c_{G1} + \sqrt{\left|c_{G2} F_W + c_{G3}\right|} & \text{für } F_W < F_{um} \\ c_{G1} + \sqrt{\left|c_{G2} F_{um} + c_{G3}\right|} + c_{G4}(F_W - F_{um}) & \text{für } F_W \geq F_{um} \end{cases} \qquad (2.4)$$

F_{um} kennzeichnet dabei den Umschaltpunkt, ab dem von einem proportionalen Zusammenhang zwischen Δs_G und F_W ausgegangen wird. Die Konstanten c_{G1} bis c_{G4} und F_{um} werden nach jedem Walzenwechsel durch Aufnahme der Gerüstkennlinie neu ermittelt.

2.2.5.2 Lageraufschwimmen

Die Walzenlager schwimmen in Abhängigkeit von Walzkraft und Drehzahl der Walzen mehr oder weniger auf und verändern dadurch den Walzspalt. Diese hydrodynamischen Effekte werden selten mit physikalischen Modellen beschrieben, stattdessen werden empirische Modelle entwickelt, die das Verhalten der Walzenlager der Walzstraße näherungsweise beschreiben. Die durch das Lageraufschwimmen verursachte Auffederung wird in der betrachteten Walzstraße nach folgender Gleichung bestimmt:

$$\Delta s_L = c_{L1} + c_{L2} F_W + \left(c_{L3} + c_{L4} F_W + C_{L5} F_W^{\,2}\right) \ln v_{um} \qquad (2.5)$$

Dabei stellen c_{L1} bis c_{L5} empirisch ermittelte Konstanten dar, die für jedes Walzgerüst variieren. Dieses Modell besitzt die Schwäche, dass weitere Einflüsse, wie z.B. die Temperaturabhängigkeit der Viskosität des Öls im Lager, unberücksichtigt bleiben.

2.2.5.3 Gagemetergleichung

Mit Hilfe der Gagemetergleichung kann die Anstellung aus der Solldicke, den Ergebnissen der Basismodelle, der Gerüstmodelle und den dickenabhängigen Anteilen der nachfolgend beschriebenen Walzenmodelle beschrieben werden. Im einfachsten Fall ergibt sich die Anstellung somit zu:

$$s = s_0(F_W, h_{Soll}) + \Delta s_G + \Delta s_L + \Delta s_W + \Delta s_{Rest} \qquad (2.6)$$

Dabei stellt Δs_W die Summe aller Walzenmodelle in der Mitte des Walzspaltes dar. Der Term Δs_{Rest} wird als *Gagemeteradaption* oder *Nullpunktfehlerkorrektur der Anstellung* bezeichnet. Er fasst alle Modell-, Messgerät- und Stellgliederfehler der Prozessautomatisierung zusammen. Es handelt sich um einen adaptiven Korrekturterm, der üblicherweise direkt aus dem Dickenfehler des zuvor gewalzten Bandes hervorgeht. An dieser Stelle werden bisher keine neuronalen Netze eingesetzt.

2.2.6 Walzenmodelle

2.2.6.1 Walzendeformationsmodell

Das Walzendeformationsmodell beschreibt die Biegung und die Abplattung unter dem Einfluss der Walzkraft über die gesamte Breite der Walze. Es bekommt als Eingabeparameter vom Umformmodell den Linienlast im Walzspalt $p_{1,0}$ und soll diesen durch Optimierung folgender Größen erreichen:

$p_{3,4}$: Linienlast auf die freie Kontur der Stützwalze

F_{a_3}, F_{b_3} : Walzkraft, in den Lagern der Stützwalze aufgebracht

Anzumerken ist, dass in vielen Anlagen immer $p_{3,4}=0$ angenommen wird, da in der Praxis nur in seltenen Fällen das Aufbringen einer Linienlast auf der Stützwalze möglich ist.

Die Walzkräfte in den Lagern der Stützwalze lassen sich unter der Voraussetzung, dass sich die Walzen tatsächlich berühren, aus dem globalen Kräfte- und Momentengleichgewicht ermitteln.

Für das Kräftegleichgewicht in vertikaler Richtung ergibt sich:

$$\int_{x_{a_1}}^{x_{b_1}} (p_{1,0})dx + \int_{x_{a_3}}^{x_{b_3}} (p_{3,4})dx + \sum_{i=1}^{3} (F_{a_i} + F_{b_i}) \equiv 0 \qquad (2.7)$$

Das Momentengleichgewicht um einen beliebigen Punkt ergibt sich:

$$\int_{x_{a_1}}^{x_{b_1}} ((x - x_{a_k})\, p_{1,0})dx + \int_{x_{a_3}}^{x_{b_3}} ((x - x_{a_k})\, p_{1,0})dx + \\ + \sum_{i=1}^{3} ((x_{a_i} - x_{a_k})F_{a_i} + (x_{b_i} - x_{b_k})F_{b_i}) + \sum_{i=1}^{3} (M_{a_i} + M_{b_i}) \equiv 0 \qquad (2.8)$$

mit $x_{a_k} := \min\{ x_{a_i} : i = 1,2,3\}$

Das Walzendeformationsmodell besteht aus drei Komponenten, die für jede Walze einzeln zu lösen sind:

- Randwertproblem für die Biegelinie
- Abplattungsmodell
- Lokales Kraft- und Momentengleichgewicht

Die Bedingung, dass sich die Walzen nicht durchdringen können, lässt sich in einer Kontaktbedingung beschreiben. Da es sich bei Walzen nicht um so genannte „schlanke Körper“ im Sinne der Biegetheorie handelt, muss sowohl die Biegung infolge Biegemoment als auch infolge von Querkraft berücksichtigt werden.

Das Abplattungsmodell lässt sich in Form einer algebraischen Gleichung formulieren, worauf an dieser Stelle aber nicht näher eingegangen wird.

Da sich die entsprechenden Differentialgleichungen nicht direkt lösen lassen, sind numerische Verfahren nötig, wobei die besten Ergebnisse dabei die Finite-Elemente-Methode liefert. Diese ist für den Echtzeiteinsatz allerdings zu rechenintensiv, stattdessen wird ein Ansatz verwendet, der das Differenzenverfahren nutzt. Der in der betrachteten Anlage verwendete Algorithmus ist in [Brüstle1992] näher beschrieben.

2.2.6.2 Thermische Ausdehnung der Walzen

Die Walzen werden während des Walzens durch das Walzgut erwärmt und zusätzlich durchgängig gekühlt, was zu einem zeitlich veränderlichen Wert der Walzentemperatur führt; diese Änderung der Walzentemperatur bedingt eine Änderung des Walzendurchmessers (thermischer Crown). Diese Durchmesseränderung wird mit Hilfe des Walzentemperaturmodells beschrieben. Das Modell hängt sehr eng mit dem im Abschnitt 2.2.4.3 beschriebenen Modell zur Bestimmung der Walzguttemperatur zusammen. Die Durchmesseränderung ist reversibel.

2.2.6.3 Walzenverschleiß

Durch den Umformprozess verschleißen die Walzen. Dieser Vorgang wird mit Hilfe des Walzenverschleißmodells beschrieben. Der Verschleiß stellt eine irreversible

Änderung des Walzendurchmessers dar, der näherungsweise proportional zur gewalzten Länge ist.

2.3 Neuronale Netze in der Prozessautomatisierung

2.3.1 Vorbemerkungen

Neuronale Netze haben sich in den letzten Jahren als Alternative zu traditionellen Adaptionsverfahren im Bereich der Prozessautomatisierung etabliert. In den nachfolgenden Abschnitten werden daher die für diese Arbeit wichtigsten Netztypen, deren softwaretechnische Umsetzung im dritten Kapitel beschrieben wird, dargestellt. Eine kurze Einführung in die Grundlagen neuronaler Netze findet sich im Anhang.

2.3.2 ADALINE

Eine grundsätzliche wissenschaftliche Regel besagt, dass einem einfachen Modell immer der Vorzug vor einem komplexen Modell zu geben ist, wenn sich nicht nachweisen lässt, dass dieses den Zusammenhang besser beschreibt. Im Rahmen der Funktionsapproximation stellen lineare Modelle die einfachsten Modelle dar. Die mathematische Methode ist hier die multiple lineare Regression.

2.3.2.1 Multiple lineare Regression

Das Ziel der statistischen Regressionsrechnung im Allgemeinen ist es, eine Aussage über die Beziehung verschiedener Vorhersagevariablen (auch unabhängige Variable) und einer abhängigen Zielgröße treffen zu können. Bei der einfachen linearen Regression besteht das Problem darin, eine Gerade optimal an eine Reihe von Punkten anzupassen; ähnliches gilt in höherdimensionalen Ergebnisräumen.

$$y^p = b_0 + \sum_{l=1}^{n} b_l x_l^p + e^p \quad mit \quad p = 1,...N \tag{2.9}$$

bzw. vektoriell

$$X = \begin{bmatrix} 1 & x_1^1 & \cdots & x_n^1 \\ 1 & x_1^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^N & \cdots & x_n^N \end{bmatrix}, y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, b = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{pmatrix} und \; e = \begin{pmatrix} e^1 \\ e^2 \\ \vdots \\ e^N \end{pmatrix} \tag{2.10}$$

$$y = X \cdot b + e \tag{2.11}$$

Ziel ist es nun, den Fehler ε zu minimieren; das bedeutet, dass die Norm des Vektors minimal werden soll. Wählt man die Euklidsche Norm, erhält man die schon 1821 von Gauß beschriebene Methode der kleinsten Quadrate [Böhme 1993]:

$$\|e\|^2 = (y - X \cdot b)^T \cdot (y - X \cdot b) \tag{2.12}$$

Eine notwendige Bedingung für das Minimum ist:

$$\frac{\partial}{\partial b}(y - X \cdot b)^T \cdot (y - X \cdot b) = -2 \cdot X^T \cdot y + 2 \cdot X^T \cdot X \cdot b = \underline{0} \tag{2.13}$$

wenn die Matrix X^TX nicht singulär ist, ergibt sich als Lösung für den Parametervektor:

$$b = (X^T \cdot X)^{-1} \cdot X^T \cdot y \tag{2.14}$$

Es sei an dieser Stelle angemerkt, dass auch Lösungen möglich sind, wenn X^TX singulär ist. Auf eine nähere Beschreibung wird hier jedoch verzichtet.

Mit Hilfe der multiplen Regression erhält man die beste lineare Prognose einer unabhängigen Variablen bei gegebenen abhängigen Größen. Da eine Beziehung aber praktisch nie wirklich linear sein wird, treten Fehler von diesem Prognosewert auf. Diese Fehler werden als *Residuen* bezeichnet. Im Anschluss an jede multiple Regression ist daher immer eine *Residualanalyse* durchzuführen. Es ist demnach zu prüfen, ob alle Residuen plausibel sind, oder ob es einige Ausreißer gibt. In diesem Fall muss das Ausgangsmodell bereinigt werden.

Eine starke Korrelation zwischen zwei Größen muss nicht zwangsläufig etwas über einen kausalen Zusammenhang aussagen, wenn eine dritte Variable, von der alle übrigen Variablen abhängen, im Modell vergessen wurde, da sie z.B. nicht messbar ist. Außerdem kann die lineare Regression nichtlineare Zusammenhänge nur unzureichend beschreiben. Einen Ausweg bietet die nichtlineare Regression, die hier jedoch nicht beschrieben werden soll, da sich an dieser Stelle neuronale Nezte mit nichtlinearen Aktivierungsfunktionen anbieten (MLPs).

An dieser Stelle sei darauf hingewiesen, dass sich die Durchführung einer linearen Regression in der Prozessmodellierung an vielen Stellen eigentlich verbietet, da sowohl abhängige als auch unabhängige Größen verrauscht sind und die Zuordnung eines Fehlers und damit die Durchführung einer Residualanalyse eigentlich nicht möglich ist. Eine Alternative stellt die orthogonale Regression dar. Diese soll hier aber nicht näher beschrieben werden, da lineare, neuronale Netze eine einfache Alternative bieten.

2.3.2.2 Lineare neuronale Netze

Das Ergebnis einer multiplen linearen Regression lässt sich in Form eines feedforward-Netzes ohne verdeckte Schicht darstellen, bei dem zwei Neuronenschichten verwendet werden. Die Eingangsneuronen geben den Eingangsvektor an das Ausgabeneuron weiter. Der Gewichtsvektor der Verbindungen ist dabei durch das Ergebnis der Regression β gegeben und der Wert β_0 stellt den Schwellenwert des Ausgangsneurons dar. Die Aktivierungsfunktion des Ausgangsneurons ist die Identität.

Der Vorteil eines solchen neuronalen Netzes liegt in seiner einfachen Topologie und in der direkten Berechenbarkeit der Netzgewichte. Die Tatsache, dass keine Nichtlinearitäten erfasst werden, erweist sich in der Realität nur selten als gravierender Nachteil. Lineare Netze können in jedem Fall als Maßstab für die Güte und die Performance komplexerer Modelle herangezogen werden. Wenn nur wenige Trainingsdaten zur Verfügung stehen, stellen lineare Modelle oft sogar das Optimum dar.

2.3.3 Multi Layer Perceptron (MLP) und Gradientenabstieg

Während lineare Netze mit der Berechnung der Pseudoinversen eine Möglichkeit besitzen, den Gewichtsvektor direkt zu bestimmen, der den Fehler im Sinne der linearen Näherung minimiert, ist dies bei nichtlinearen Aktivierungsfunktionen nicht möglich. Für nichtlineare neuronale feedforward-Netze werden iterative Verfahren eingesetzt. Der Backpropagation-Algorithmus ist eine Methode, die versucht, mit einem Gradientenabstiegsverfahren den Fehler eines neuronalen Netzes zu minimieren.

2.3.3.1 Gradientenverfahren

Gradientenverfahren berechnen den Gradienten einer Zielfunktion, die es entweder zu maximieren oder zu minimieren gilt. Für die Anwendung auf neuronale Netze handelt es sich bei der Zielfunktion um die Fehlerfunktion E(W), die den über alle Trainingsmuster aufsummierten Fehler angibt. Beim Gradientenverfahren wird versucht, durch Änderung der Gewichte in Richtung des negativen Gradienten von E(W) ein Minimum der Fehlerfunktion zu erreichen. Die Änderung des Gewichtsvektors ΔW ist proportional zum steilsten Abstieg auf der Fehlerfläche:

$$\Delta W = -h \nabla E(W) \tag{2.15}$$

oder für jedes einzelne Gewicht

$$\Delta w_{ij} = -h \frac{\partial}{\partial w_{ij}} E(W) \tag{2.16}$$

Dabei stellt η die bekannte Lernrate dar. Als Fehlerfunktion für ein Eingabemuster p findet meistens der quadratische Abstand zwischen gewünschtem y_j und tatsächlichem Wert t_j Verwendung:

$$E^p = \frac{1}{2}\sum_j \left(t_j^p - y_j^p\right)^2 \tag{2.17}$$

Für den Gesamtfehler aller N Eingabemuster ergibt sich somit:

$$E(W) = \sum_{p=1}^{N} E^p \tag{2.18}$$

2.3.3.2 Der Backpropagation-Algorithmus

Die im Folgenden beschriebene Ableitung des Backpropagation-Algorithmus wurde bereits 1974 von Werbos publiziert [Werbos1974], wurde aber erst durch die Veröffentlichungen von Rummelhart und McClelland [RumMcC1986] einer breiteren Öffentlichkeit zugänglich. Die hier dargestellte Variante orientiert sich an den Darstellungen in [Hinton1992], [Zell1994] und [Bishop1995].

Der Eingang eines Neurons j lässt sich mit der bekannten Propagierungsfunktion für ein Eingabemuster p angeben:

$$x_j^p = \sum_i y_i^p \cdot w_{ij} \tag{2.19}$$

Für ein Neuron j mit monotoner, differenzierbarer Aktivierungsfuntion $f_{act,j}$, ergibt sich die Ausgabe zu:

$$y_j^p = f_{act,j}(x_j^p) \tag{2.20}$$

Die Gewichtsänderung ist nach (2.16):

$$\Delta w_{ij} = -h\frac{\partial E(W)}{\partial w_{ij}} = \sum_{p=1}^{N} -h\frac{\partial E^p}{\partial w_{ij}} \tag{2.21}$$

Durch Anwendung der Kettenregel lässt sich die Ableitung schrittweise bestimmen:

$$\frac{\partial E^p}{\partial w_{ij}} = \frac{\partial E^p}{\partial x_j^p}\frac{\partial x_j^p}{\partial w_{ij}} \tag{2.22}$$

Der zweite Faktor ist nach (2.19):

$$\frac{\partial x_j^p}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}\sum_i y_j^p w_{ij} = y_j^p \tag{2.23}$$

Den ersten Faktor definiert man analog zur Delta-Regel als Fehlersignal:

$$d_j^p = -\frac{\partial E^p}{\partial x_j^p} = -\frac{\partial E^p}{\partial y_j^p} \cdot \frac{\partial y_j^p}{\partial x_j^p} \tag{2.24}$$

Der zweite Faktor in dieser Gleichung lässt sich aus (2.20) bestimmen:

$$\frac{\partial y_j^p}{\partial x_j^p} = \frac{\partial}{\partial x_j^p} f_{act}(x_j^p) = f'_{act}(x_j^p) \tag{2.25}$$

Für den ersten Faktor ist eine Fallunterscheidung zu treffen. Wenn j eine Ausgabezelle ist, gilt:

$$-\frac{\partial E_k}{\partial y_j^p} = t_j^p - y_j^p \tag{2.26}$$

Ansonsten müssen die partiellen Ableitungen indirekt über die Kettenregel berechnet werden. Dabei wird ausgenutzt, dass die Ein- und Ausgaben der Nachfolgeschicht bekannt sind:

$$-\frac{\partial E^p}{\partial y_j^p} = -\sum_k \frac{\partial E^p}{\partial x_k^p} \frac{\partial x_k^p}{\partial y_j^p} = \sum_l d_k^p \cdot \frac{\partial}{\partial y_j^p} \sum_i y_i^p w_{ik} = \sum_l d_k^p \cdot w_{jk} \tag{2.27}$$

Damit ergibt sich als Lösung

$$\Delta^p w_{ij} = h\, y_i^p d_j^p \tag{2.28}$$

mit

$$d_k^p = \begin{cases} f'_{act}(x_j^p)(t_j^p - y_j^p) & falls\ j\ ein\ Ausgabeneuron\ ist \\ f'_{act}(x_j^p) \sum_k (d_k^p w_{jk}) & falls\ j\ ein\ verdecktes\ Neuron\ ist \end{cases} \tag{2.29}$$

Wenn als Aktivierungsfunktion die logistische Funktion s_1 gewählt wird, vereinfacht sich das Ergebnis mit

$$f'_{act}(x_j^p) = f_{act}(x_j^p)(1 - f_{act}(x_j^p)) = y_j^p(1 - y_j^p) \tag{2.30}$$

zu

$$d_k^p = \begin{cases} y_k^p(1 - y_k^p)(t_j - y_j) & falls\ j\ ein\ Ausgabeneuron\ ist \\ y_k^p(1 - y_k^p) \sum_k (d_k w_{jk}) & falls\ j\ ein\ verdecktes\ Neuron\ ist \end{cases} \tag{2.31}$$

Die Weitergabe der Ergebnisse der Nachfolgeschichten an die Vorgänger gibt dem Algorithmus seinen Namen: Backpropagation.

2.3.3.3 Probleme des Algorithmus

Der Backpropagation-Algorithmus hat sich beim Training von Multi-Layer-Perceptrons (MLPs) in der Praxis bewährt. Insbesondere gilt dies für Fälle, in denen viele Trainingsdaten zur Verfügung stehen. Aufgrund der großen Anzahl an Rechenoperationen ist es aber kaum praktikabel, Netze mit mehr als 100 Neuronen zu trainieren.

Das Backpropagation-Prinzip ist allerdings aus der Sicht des Biologen unplausibel, da hier, entgegen dem Verhalten der natürlichen Neuronen, beim Lernen entgegengesetzt zur Richtung Eingabe-Verarbeitung-Ausgabe vorgegangen wird.

Eine Reihe weiterer Probleme resultiert aus der Tatsache, dass Backpropagation ein lokales Verfahren ist, das keine Kenntnis über die gesamte Fehlerfunktion besitzt. Im Folgenden sind die wichtigsten Probleme kurz erläutert.

Als *symmetry breaking* bezeichnet man ein Problem, das auftritt, wenn alle Gewichte gleich initialisiert werden. Es ist dann nicht mehr möglich, dass sich in Schichten unterhalb der Ausgabeschicht unterschiedliche Gewichte ausbilden. Dieses Problem lässt sich sehr einfach durch die Initialisierung der Gewichte mit kleinen Zufallszahlen vermeiden.

Wie bei allen Gradientenverfahren kann es vorkommen, dass der Algorithmus nicht das globale, sondern nur ein lokales Minimum findet. Die Wahrscheinlichkeit hierfür steigt mit zunehmender Komplexität des Netzes.

Die Konvergenzgeschwindigkeit von Backpropagation hängt entschieden davon ab, ob die Fehlerfläche flache Plateaus und steile Schluchten besitzt. Auf flachen Plateaus ist der Gradient nahe Null und es findet kaum noch eine Bewegung in Richtung des Optimums statt. Steile Schluchten können dazu führen, dass das Lernverfahren am Minimum vorbeiläuft oder zu Oszillationen neigt.

2.3.3.4 Modifikation von Backpropagation

Um die Probleme des Algorithmus zu lösen, existieren eine Reihe von Ansätzen, wovon im Folgenden die wichtigsten kurz erläutert werden.

Konjugierter Gradientenabstieg (*conjugate gradient descent*) oder auch Backpropagation mit Momentumterm wurde bereits von Rummelhart, Hinton und Williams in [RumMcC1986] beschrieben, Erweiterungen publizierte Bischop 1995 [Bishop1995]. Das Verfahren versucht, durch Berücksichtigung der im letzten Schritt durchgeführ-

ten Gewichtsänderung die Probleme, die Backpropagation auf flachen Plateaus und in engen Schluchten hat, zu umgehen. Die Gewichtsänderung ergibt sich zu:

$$\Delta^p w_{ij}(t+1) = h\, y_i^p d_j^p + a\Delta^p w_{ij}(t) \tag{2.32}$$

Das Verfahren *weight decay* [Werbos1988] berücksichtigt in der Fehlerfunktion die Tatsache, dass es neurobiologisch unplausibel ist, beliebig große Gewichte zuzulassen. Daher werden große Gewichte durch E(W) bestraft:

$$E_{WD} = E_{BP} + \frac{d}{2}\sum_i \sum_j (w_{ij})^2 \tag{2.33}$$

Zur Erzielung einer höheren Konvergenzgeschwindigkeit kann zusätzlich zur ersten Ableitung auch die zweite Ableitung, also die Krümmung der Fehlerfläche, herangezogen werden. Ein solches Verfahren stellt Second-Order-Backpropagation dar.

Eine andere Möglichkeit, die Konvergenz zu erhöhen, ist mit Quickprop gegeben. Quickprop ist ein iteratives Verfahren zweiter Ordnung, das zur Berechnung des nächsten Parameterwertes die letzten beiden Parametervektoren verwendet. Es ergibt sich folgende Lernregel:

$$\Delta w_{ij}(t) = \frac{\frac{\partial E}{\partial w_{ij}}(t)}{\frac{\partial E}{\partial w_{ij}}(t-1) - \frac{\partial E}{\partial w_{ij}}(t)} \cdot \Delta w_{ij}(t-1) \tag{2.34}$$

2.3.4 Radiale-Basisfunktionen-Netze (RBF)

Radiale-Basisfunktionen-Netze (*radial basis functions networks*, *RBF*-Netze) sind feedforward-Netze, die nur eine verdeckte Schicht besitzen. Die Besonderheit der Neuronen in der verdeckten Schicht besteht in ihren radialsymmetrischen Aktivierungsfunktionen, die die Basisfunktionen eines Funktionssystems zur Approximation von mehrdimensionalen Funktionen darstellen.

Da die Aktivierungsfunktionen der Neuronen nur für Werte in der Nähe der Stützstellen große Werte liefern, können Muster, die außerhalb des trainierten Bereichs liegen, nicht wie beim MLP unvorhersagbare Werte liefern.

Die einfache Struktur der RBF-Netze erlaubt eine direkte Berechnung der Gewichte.

Die Darstellung der RBF-Netze in den folgenden Abschnitten orientiert sich an [Zell1994] und damit an [PogGir1989]. Die Theorie der RBF-Netze geht auf die Approximationstheorie von Funktionen mehrerer Veränderlicher zurück. Nachfolgend wird schrittweise zunächst die Interpolation und anschließend die

Approximation einer Funktion mit radialsymmetrischen Basisfunktionen erläutert und auf Besonderheiten bei der Realisierung als RBF-Netz hingewiesen.

2.3.4.1 Interpolation mit Zentrumsfunktionen

Die zu interpolierende Funktion f, die eine Abbildung vom $\mathbb{R}^n$ nach $\mathbb{R}$ darstellt, sei an N Stützstellen gegeben. Jede Stützstelle i besteht aus einem n-dimensionalen Eingangsvektor x_i und dem zugehörigen Funktionswert y_i. Hier wird eine veränderte Notation verwendet (Subskript i statt Superskript p), um die Analogie mit den im nächsten Abschnitt beschriebenen Netzen aufzuzeigen. Als Interpolationsbedingung gilt:

$$f(x_i)=y_i \tag{2.35}$$

Zur Interpolation wird ein Funktionensystem eingesetzt, das aus radialsymmetrischen Basisfunktionen h_i besteht, die von den Stützstellen x_i abhängen:

$$h_i\,(||x-x_i||) \tag{2.36}$$

Meistens wird für die Norm die euklidsche L_2-Norm verwendet. Es ergibt sich folgende Interpolation der Funktion f:

$$f(x)=\sum_{i=1}^{N} c_i\, h_i\,(\|x-x_i\|) \tag{2.37}$$

Abbildung 2.3 zeigt exemplarisch, wie sich eine eindimensionale Beispielfunktion f aus drei Zentrumsfunktionen ergibt.

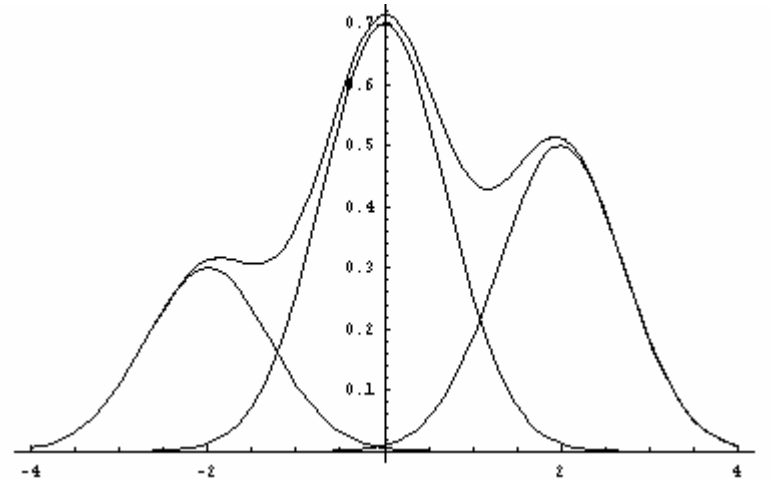


Abbildung 2.3: Interpolation einer Beispielfunktion

Es bleibt als Aufgabe die Bestimmung der Gewichtsfaktoren c_i, die sich direkt ermitteln lassen. Es ergibt sich folgendes Gleichungssystem mit N Gleichungen und N Unbekannten:

$$\sum_{p=1}^{N} c^p h^p(\| x^i - x^p \|) = y^i \quad \text{für } i = 1,\ldots, N \tag{2.38}$$

Für die zumeist gewählte Gaußfunktion als Zentrumsfunktion lässt sich zeigen, dass das Interpolationsproblem direkt lösbar ist [PogGir1989]. Durch Einführung der Matrix H und der Vektoren C und Y lassen sich die Gleichungen auch vektoriell angeben:

$$H\,C = Y \tag{2.39}$$

Dann ergibt sich als Lösung:

$$C = H^{-1}\,Y \tag{2.40}$$

Bei verrauschten Eingangsgrößen hat sich eine leichte Variation des Ergebnisses bewährt (nach Tikhonov und Arsenin):

$$C = (H + l\,I)^{-1}\,Y \tag{2.41}$$

Der Parameter l ist dabei proportional zum Rauschen der Eingangsgrößen.

Im allgemeinen Fall werden neben den Zentrumsfunktionen auch noch m Polynome zur Funktion f addiert, um das Interpolationsproblem zu lösen und die Funktion zu glätten. Diese Polynome bilden die Basis eines Polynomraums. Der maximale Grad der Polynome k-1 hängt dabei von den Zentrumsfunktionen ab. Es ergibt sich die verallgemeinerte Funktion f:

$$f(x) = \sum_{i=1}^{N} c_i h_i(\|x - x_i\|) + \sum_{j=1}^{m} d_j p_j(x) \tag{2.42}$$

Das in diesem Abschnitt beschriebene Verfahren zur Interpolation einer Funktion f lässt sich sehr einfach als RBF-Netz realisieren. In der Praxis bringt die Interpolation jedoch einige gravierende Nachteile mit sich:

- Die Funktion kann stark oszillieren, obwohl die Interpolationsbedingungen erfüllt sind. Polynome hohen Grades, die zur Glättung dienen sollen, können dies eventuell sogar noch verstärken.
- Eine exakte Interpolation ist bei technischen Prozessen, bei denen die Stützstellen aus Messwerten resultieren, die immer verrauscht sind, nicht sinnvoll.
- Die Größe des Gleichungssystems entspricht der Anzahl der Stützstellen, wodurch auch die Netzgröße proportional mit der Anzahl an Trainingsmustern wächst. Es ist aber wünschenswert, dass die Netzgröße problemabhängig und unabhängig von der Zahl der Stützstellen ist.

Die Forderung nach einer möglichst glatten Funktion f und einer problemabhängigen Netzgröße führt auf die verallgemeinerten RBF-Netze. Diese lösen dann allerdings kein Interpolationsproblem mehr, sondern ein Approximationsproblem unter Berücksichtigung von Glattheitsbedingungen an die Funktion f.

2.3.4.2 Approximation mit Zentrumsfunktionen

Die Approximation mit Zentrumsfunktionen verwendet unabhängig von der Zahl N der Trainingsmuster K Zentren (also verdeckte Neuronen), die als Stützstellen der Zentrumsfunktionen dienen. Die Schwierigkeit besteht nun darin, diese Zentren zu ermitteln. Es bieten sich zwei Möglichkeiten: Entweder man verwendet eine Untermenge der Eingangsvektoren oder man schaltet eine weitere Neuronenschicht zur Klassifizierung vor. Wird der zweite Weg beschritten, kommen meistens die im nächsten Abschnitt beschriebenen Verfahren zur Clusteranalyse zum Einsatz. Da die Stützstellen nicht zwangsläufig eine Untermenge der Eingangsvektoren darstellen, werden diese nun mit W_j bezeichnet, da sie durch die Gewichtsvektoren in der Eingangsschicht im Netz gespeichert werden. Ein typisches RBF-Netz ist in Abbildung 2.4 dargestellt.

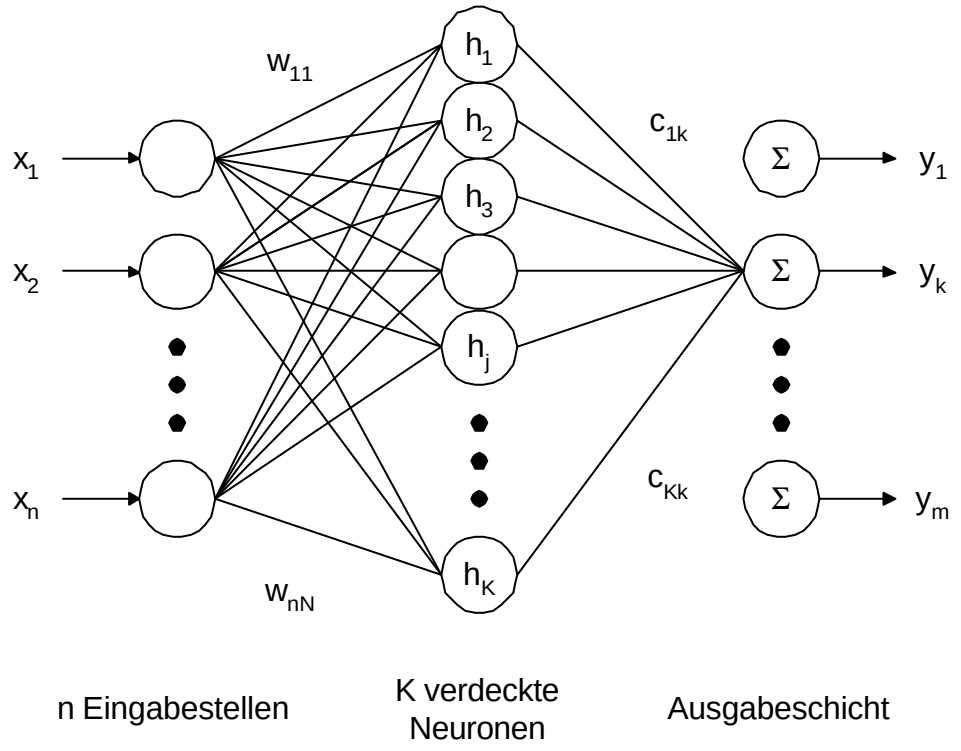


Abbildung 2.4: Aufbau eines RBF-Netzes

Für die zu approximierende Funktion ergibt sich dann:

$$f(x) = \sum_{i=1}^{K} c_i h_i(\| x - W_i \|) \tag{2.43}$$

Schreibt man die Bestimmungsgleichung für die c_i wieder in Matrixform, ergibt sich:

$$H\,C = Y \tag{2.44}$$

Für N>K ist das Problem überbestimmt. Es wird daher versucht, die Summe der Fehler aller Trainingsmuster zu minimieren, woraus sich als Lösung ergibt:

$$C = H^{+}\,Y \tag{2.45}$$

mit der „Moore-Penrose-Pseudoinversen" der Matrix H:

$$H^{+} = (H^{T}H)^{-1}\,H^{T} \tag{2.46}$$

Es lässt sich zeigen, dass die Pseudoinverse existiert, wenn die Ableitung h´ vollständig monoton ist und die Zentren W_j eine Untermenge der Trainingsmuster x_j darstellen.

Ermittelt man die Lösung des Approximationsproblems mit Hilfe der Variationsrechnung, können auch die Glattheitsbedingungen an f berücksichtigt werden. Es ergibt sich als Lösung für den Gewichtsvektor [Zell1994]:

$$C = (H^{T}H + \lambda\, H')^{-1}\,H^{T}Y \tag{2.47}$$

mit

$$H = \begin{bmatrix} h(x_1,W_1) & \cdots & h(x_1,W_K) \\ \vdots & \ddots & \vdots \\ h(x_N,W_1) & \cdots & h(x_N,W_K) \end{bmatrix} \quad H' = \begin{bmatrix} h(W_1,W_1) & \cdots & h(W_1,W_K) \\ \vdots & \ddots & \vdots \\ h(W_K,W_1) & \cdots & h(W_K,W_K) \end{bmatrix} \tag{2.48}$$

Der Parameter λ ist dabei wiederum proportional zum Rauschen der Eingangsgrößen.

Weitere Verallgemeinerungen der RBF-Netze erhält man, wenn sie um einen linearen Anteil erweitert werden. Dies soll hier jedoch ebensowenig beschrieben werden wie der allgemeine Fall der Hyper-Basisfunktionen-Netze (HBF-Netze). Es sei nur noch angemerkt, dass RBF-Netze sich problemlos auf mehrwertige Funktionen erweitern lassen.

2.3.4.3 Iteratives Nachtraininieren

Die direkte Bestimmung der Netzparameter ist einer der Vorteile von RBF-Netzen. Es gibt jedoch einige Probleme: Zum einen ist die Wahl der Zentren nicht optimal, zum anderen treten bei der Berechnung der Pseudoinversen numerische Probleme auf; das bedeutet, dass im Allgemeinen weder die Gewichte der Eingangsschicht, noch die Gewichte der Ausgangsschicht optimal gewählt werden. Daher sollten RBF-Netze

nach der Initialisierung iterativ nachtrainiert werden. Der Trainingsalgorithmus folgt dabei den Ideen von Backpropagation; er soll hier daher nicht näher erläutert werden.

Die in den letzten Abschnitten beschriebenen Netze eignen sich sowohl für Klassifikations- als auch für Approximationsaufgaben. Im nächsten Abschnitt werden Verfahren beschrieben, die eine optimierte Wahl der Zentren erlauben.

2.3.5 Clusterverfahren

Clusterverfahren werden u.a. zur Bestimmung der Zentren von RBF-Netzen verwendet. Nachfolgend sind drei Verfahren beschrieben, die im Rahmen dieser Arbeit verwendet werden.

2.3.5.1 Das k-Means-Verfahren

Das k-Means-Verfahren bildet Cluster-Zentren aus. Das Kriterium zur Verteilung der Zentren sind die minimalen Abstände der Datenpunkte zum jeweils nächsten Cluster-Zentrum. Der Algorithmus endet, wenn die Verbesserung nach einem Adaptionsschritt unter einer bestimmten Schranke ε liegt oder eine vorgegebene maximale Anzahl Iterationen erreicht ist.

2.3.5.2 Erweiterung des k-Means-Verfahrens

Eine wesentliche Eigenschaft des k-Means-Verfahrens und des anschließend beschriebenen Neural-Gas-Verfahrens ist eine Verteilung der Zentren in Abhängigkeit der Dichteverteilung der Daten, die dazu führen kann, dass auf der einen Seite dünn besetzte Raumbereiche nur wenige oder gar keine Zentren erhalten, in denen dann auch der durchschnittliche Abstand der Datenpunkte zu einem Zentrum relativ hoch ist, auf der anderen Seite dicht besetzte Raumbereiche jedoch zahlreiche Zentren erhalten, deren Abstände untereinander unter Umständen recht gering sind.

Um eine Zentrenverteilung zu erzielen, die einerseits die Verteilung der Daten repräsentiert, andererseits eine möglichst optimale Raumabdeckung ermöglicht, lässt sich das k-Means-Verfahren zum Advanced-k-Means-Verfahren (AKM) erweitern, in dem das k-Means-Verfahren iterativ aufgerufen wird. Heuristiken zum Einfügen und Entfernen von Zentren, um eine bessere Raumabdeckung zu erhalten, ergänzen das k-Means-Verfahren. Eine Epoche des AKM setzt sich aus folgenden Schritten zusammen:

1. Zentren einfügen
2. Zentren mit k-Means zentrieren
3. Zentren ausdünnen

2.3.5.3 Neural Gas

Das Neural-Gas-Verfahren stellt eine Verallgemeinerung des k-Means-Verfahren dar. Die iterative Verteilung der Zentren erfolgt nicht durch ein hartes Wettbewerbslernen (winner-takes-all), sondern alle Zentren sind gleichzeitig beteiligt, aber natürlich mit einer Gewichtung, die vom jeweiligen Abstand des Zentrums zum Datenpunkt abhängt.

Das Prinzip des Einfrierens des „Neuronalen Gases“ wird in dreifacher Weise realisiert:

- Die Adaptionsrate nimmt für die Zentren mit der Anzahl der Lernschritte exponentiell ab.
- Die Adaptionsrate nimmt für die Biase und Steigungen der lokal linearen Modelle ab.
- Die Adaption wird zunehmend lokaler vorgenommen; d.h. entferntere Datenpunkte werden bei der Optimierung der Zentrenposition immer weniger berücksichtigt.

2.3.6 Clusteranalyse mit neuronalen Netzen

Neben den hier beschriebenen klassischen Clusterverfahren existieren mittlerweile auch eine ganze Reihe von neuronalen Methoden. Im Anhang sind daher die selbstorganisierenden Karten (SOM – self organizing maps) und verschieden LVQ-Netze (Lernende Vektor-Quantisierung) beschrieben.

Lernende Vektorquantisierung (LVQ) und selbstorganisierende Karten (SOM) sind zwei Lernverfahren, die beide auf Kohonen zurückgehen [Kohonen1982], [Kohonen1990] und miteinander verwandt sind. Bei beiden Netztypen handelt es sich um einschichtige neuronale Netze, die neben einer optionalen Eingabeschicht nur eine Schicht aktiver Neuronen besitzen. Diese Neuronen werden auch als Kohonen-Neuronen bezeichnet.

2.3.7 Hinweise zum Einsatz von neuronalen Netzen

2.3.7.1 Abgrenzung des Nutzens

Neuronale Netze können nur dann akzeptable Ergebnisse liefern, wenn sehr viele vollständige Beispiele vorliegen, mit denen sie trainiert werden können; des weiteren muss der Fehler, den ein Netz macht, bewertet werden können. Da diese Voraussetzungen in der Prozessautomatisierung vorliegen, können neuronale Netze hier vor allem zur Prognose von Prozessgrößen und zum Ersatz bzw. zur Ergänzung rechen-

intensiver Modelle in zeitkritischen Anwendungen verwendet werden. Neuronale Netze reproduzieren das, was sie aus Beispielen gelernt haben.

2.3.7.2 Wahl der Eingangsgrößen

Bei der Wahl der Eingangsgrößen ist darauf zu achten, dass Inputs gewählt werden, bei denen ein Einfluss auf die Ausgangsgröße zu vermuten bzw. bekannt ist.

Des Weiteren ist zu beachten, dass konstante und stark verrauschte Eingänge keine Verbesserung des Ergebnisses erreichen können. Einen wichtigen Hinweis auf eine stark verrauschte Größe liefert die Spannweite. Bewegt sich diese in der Größenordnung des Messfehlers, so liegt keine sinnvolle Eingangsgröße vor. Eine weitere Minimierung der Eingangsgrößen lässt sich erreichen, wenn stark mit anderen Eingängen korrelierte Größen unberücksichtigt bleiben.

Alle Eingangsgrößen des neuronalen Netzes müssen zum Zeitpunkt der Berechnung zur Verfügung stehen. Es reicht demnach nicht, wenn die Daten zum Trainieren des Netzes bereit stehen. Diese Einschränkung schließt in technischen Prozessen oft eine Reihe von Eingangsgrößen aus, da viele Messwerte erst zum Prozessende vorliegen.

Es zeigt sich, dass es vorteilhaft ist, Kategorien als binäre Größen darzustellen. Gibt es mehrere Kategorien, die zu starken Unterschieden in den Ausgangsgrößen führen, ist es manchmal sinnvoll, das neuronale Netz in mehrere Unternetze aufzuteilen. Die Kategorisierung erfolgt über eine Cluster-Analyse.

Theoretisch gibt es für die maximale Anzahl an Eingangsgrößen keine Beschränkung. Es hat sich jedoch gezeigt, dass mit der heute zur Verfügung stehenden Hardware nur Netze mit maximal 100 Eingängen berechnet werden können.

Zudem ist zu beachten, dass die Zahl der Trainingsbeispiele, die für die gleichmäßige Abdeckung des Eingangsraumes erforderlich ist, exponentiell mit der Zahl der Eingänge ansteigt (*curse of dimension* [Bishop1995]). Ob eine gleichmäßige Abdeckung erforderlich ist, hängt allerdings von der jeweiligen Aufgabe ab.

2.3.7.3 Wahl der Trainingsbeispiele

Bei der Wahl der Trainingsbeispiele ist zu beachten, dass zum einen genug und zum anderen möglichst gleichverteilte Muster gewählt werden.

Die notwendige Anzahl an Beispielen lässt sich nicht genau abschätzen, grundsätzlich gilt jedoch, dass viele Beispiele besser sind als wenige. Die Zahl der Beispiele sollte bei einem Multi-Layer-Perceptron mindestens so groß sein, wie das Produkt aus Anzahl der Eingangsgrößen und Anzahl der verdeckten Neuronen. Insbesondere

bei verrauschten Signalen gilt aber, dass die Zahl der Beispiele besser eine Größenordnung größer sein sollte als dieses Produkt. Werden zu wenige Beispiele gewählt, besteht die Gefahr, dass das Netz „auswendig lernt“ und somit für unbekannte Eingangsvektoren nicht mehr hinreichend genau interpolieren- bzw. generalisieren kann.

Um einen gleichmäßig kleinen Fehler über den gesamten Bereich der Eingabevektoren erreichen zu können, müßten die Trainingsbeispiele etwa gleichverteilt sein. Tatsächlich trifft dies jedoch sehr selten zu, meistens sind die Beispiele normalverteilt. Dabei sind eingipflige Verteilungen unproblematisch, da die negativen Effekte in Randbereichen sehr unwahrscheinlich sind; bei zweigipfligen Verteilungen, mit einem zweiten Gipfel bei einem Vielfachen der Standardabweichung, wird ein einfaches neuronales Netz diese Werte ignorieren. Hier verspricht die Einführung eines zweiten Netzes Besserung.

2.3.7.4 Auswahl des Netztyps und des Trainingsverfahrens

Für Approximationsaufgaben empfehlen sich die vorgestellten MLPs oder RBF-Netze, für einfache Lösungen oder zum Prototyping auch ADALINEs. Andere Netztypen, wie z.B. die hier nicht beschriebenen GRNNs, können durch ihr sehr einfaches Trainingsverfahren dazu benutzt werden, brauchbare von unbrauchbaren Eingangsgrößen zu trennen. LVQ und SOM werden alternativ zur klassischen Clusteranalyse eingesetzt, die einem RBF-Netz vorgeschaltet werden kann.

Kommt ein MLP zum Einsatz, ist zunächst zu entscheiden, wieviele verdeckte Schichten verwendet werden. Im Allgemeinen sollte man zunächst testen, ob sich das Problem mit einer verdeckten Schicht lösen lässt und nur wenn dies nicht funktioniert, weitere verdeckte Schichten ergänzen. Für die Gesamtzahl der Neuronen in der verdeckten Schicht ist in [Kolmogorov1957] eine theoretische Obergrenze angegeben. In der Praxis ist der Einsatz von deutlich weniger Neuronen aber oft ausreichend. Auch bezüglich des Lernverfahrens kann man keine direkte Empfehlung für ein spezielles Lernverfahren angegeben. Es hat sich gezeigt, dass durchaus verschiedene Varianten von Backpropagation bei ähnlichen Aufgaben das Optimum darstellen können.

2.4 Methoden der Softwaretechik

2.4.1 Vorbemerkungen

Im nachfolgenden dritten Kapitel wird ein im Rahmen dieser Arbeit entwickelter softwaretechnischer Entwicklungsrahmen dargestellt. In den nächsten Abschnitten werden daher die wichtigsten softwaretechnischen Methoden beschrieben. Eine

etwas ausführlichere Erläuterung mit Verweisen auf die Literatur findet sich im Anhang.

2.4.2 UML – Unified Modelling Language

Die Unified Modelling Language – oder kurz UML – ist der Standard zur Spezifizierung, Visualisierung, Konstruktion und Dokumentation von komplexen Softwaresystemen. UML stellt dabei verschiedene Diagramme bereit, mit denen sich das betrachtete System modellieren lässt.

Der Mittelpunkt der Notation stellt die grafische Visualisierung von Klassen dar. Dieser Teil wird im Anhang erläutert.

2.4.3 Entwicklungsrahmen

Die objektorientierte Analyse stellt mit dem Konzept der Objektorientierung eine Methodik zur Verfügung, die eine konsequente Wiederverwendung von Software ermöglicht. Um dies auch in der Realität zu erreichen, existieren weitergehende Verfahren.

Eine Möglichkeit besteht in der so genannten pattern-orientierten Software-Architektur, die sich zum Ziel gesetzt hat, Software mit Hilfe von Mustern (Pattern) zu entwerfen. Dabei beschreiben Muster erprobte Lösungen für häufig wiederkehrende Probleme in verschiedenen Phasen des Softwareentwicklungsprozesses. Da diese Arbeit nur bekannte Muster nutzt, sei an dieser Stelle auf die weiterführende Literatur verwiesen ([GHJV1995] und [BMRSS1998]).

Frei nach Erich Gamma [GHJV1995] lässt sich ein solcher *Entwicklungsrahmen* – häufig auch als *Framework* bezeichnet - als Menge von kooperierenden Klassen definieren, die einen wiederverwendbaren Entwurf für eine spezielle Art Software vorgeben. Ein Framework vergibt architektonische Richtlinien durch Unterteilung des Entwurfs auf verschiedene abstrakte Klassen und die Definition ihrer Bedeutung und Zusammenarbeit. Durch Anpassung entsteht aus dem Framework eine Software, wenn aus den abstrakten Klassen des Frameworks konkrete Klassen abgeleitet werden.

2.4.4 Generierung aus OOA-Modellen

Eine weitere Möglichkeit der Wiederverwendung bietet sich durch direktes Generieren von impliziten Operationen aus einem OOA-Modell. Dieses Thema wird ausführlich in [Wick1996] und [Niemann2000] beschrieben und in Kapitel 3.13 weiter vertieft.

2.4.5 XML

Im Jahr 1996 begann das World Wide Web Consortium (W3C) damit, eine neue Standardauszeichnungssprache zu entwickeln. Das Ergebnis stellt XML dar.

Ein XML-Dokument besteht aus Elementen, Attributen und Kommentaren. Die Regeln, die die Gültigkeit eines Dokumentes festlegen, können in Form einer DTD oder eines XML-Schemas definiert werden. XML-Dokumente müssen außerdem *wohlgeformt* sein, wobei wohlgeformt informell bedeutet, dass ein Dokument ein oder mehrere Elemente enthalten muss, die von einem Wurzelobjekt eingebettet werden, und dass eingebettete Elemente vollständig eingeschlossen werden.

XSLT ist eine Sprache zur Transformation von XML-Dokumenten in andere textbasierte Dokumente; dies können wiederum XML-Dokumente oder auch formatierte Textdateien sein. Im Rahmen dieser Arbeit wird XSLT zur Transformation von XML-Dokumenten eines Schemas zu XML-Dokumenten eines anderen Schemas verwendet.

XPath ist eine Sprache zum Referenzieren einzelner Teile eines XML-Dokumentes. Im Rahmen dieser Arbeit wird XPath einerseits zur Definition von Suchmustern bei der Transformation von XML-Dokumenten und andererseits als Abfragesprache verwendet.

3 Modellierung von Vorgängen im Walzspalt

3.1 Motivation

Um die komplexen Vorgänge im Walzspalt beschreiben zu können, werden die im vorangegangenen Kapitel beschriebenen Prozessmodelle verwendet. Zum Einsatz dieser Modelle in einer Prozessautomatisierung müssen sie zusätzlich softwaretechnisch modelliert und implementiert werden.

Im Normalfall wird diese softwaretechnische Umsetzung für jedes Modell einzeln durchgeführt. In der Konsequenz bedeutet dies häufig, dass die softwaretechnischen Realisierungen der Prozessmodelle nicht mehr dieselbe Transparenz zulassen wie die analytischen Varianten. Dies liegt in erster Linie an uneinheitlichen Modellschnittstellen und verschiedenen Adaptionsverfahren.

Die durch uneinheitliche Modellschnittstellen verursachten Fehler sind in Relation zum Gesamtfehler zwar eher unbedeutend, sie verhindern jedoch eine saubere Diagnose von Fehlern. Wenn ein Fehler aufgetreten ist, lässt sich nur noch schwer herausfinden, welches Prozessmodell für diesen Fehler verantwortlich ist und wie ein solcher Fehler in Zukunft vermieden werden kann.

Der Einsatz von neuronalen Netzen zur Prognose von Restfehlern und zur Diagnose von Prozessfehlern ist in diesem Zusammenhang ebenfalls problematisch. Es zeigt sich, dass neben nicht-kompatiblen Modellschnittstellen auch unterschiedlichste Adaptionsmechanismen Verwendung finden. Durch diese uneinheitliche Methodik ist es nahezu unmöglich, Restfehler auf einzelne Modelle aufzuteilen bzw. kaum möglich, modellspezifische Anteile der Adaption zu detektieren.

Ein Ansatz, diese Herausforderungen zu lösen, besteht darin, ein softwaretechnisches Rahmenwerk zu verwenden, das die Entwicklung von Prozessmodellen vorgibt. Mit einem solchen *Entwicklungsrahmen* (auch *Framework* genannt) könnten die beschriebenen Probleme gelöst werden. Ein solcher Entwicklungsrahmen wurde daher im Rahmen dieser Arbeit entworfen und implementiert und wird in den folgenden Abschnitten vorgestellt.

Die vorgestellte Methode verfolgt das Ziel, beliebige Prozessmodelle derart zu verallgemeinern, dass sie nur noch über standardisierte Schnittstellen kommunizieren. Da Einzelmodelle und das Gesamtmodell adaptiv ausgelegt werden sollen, werden Mechanismen bereitgestellt, die die Kombination von statistischen Modellen und neuronalen Netzen zu hybriden Modellen sehr stark vereinfachen.

3.2 Frameworks und Klassenbibliotheken in der Prozessautomatisierung

In der Prozessautomatisierung von Walzwerken werden eine Vielzahl verschiedenster Frameworks, Klassen- und Funktionsbibliotheken verwendet, die vorrangig dem Lösen einzelner Probleme dienen, wie sie im letzten Kapitel beschrieben wurden. Die Integration der Datenhaltung wird dabei nur selten berücksichtigt. Frameworks, die eine Kombination von Prozessmodellen und Adaptionsmechnanismen bieten, sind ebenfalls kaum vorhanden.

Stattdessen werden zur Lösung jedes einzelnen Problems verschiedene Frameworks und Bibliotheken eingesetzt, die jedoch meistens disjunkte Basisklassen besitzen, die eine softwaretechnische Umsetzung des Prozessmodells erschweren.

Adaptionsmechanismen, im speziellen neuronale Netze, werden heute an verschiedenen Stellen eingesetzt. Zur Realisierung bieten sich Neuro-Simulatoren, wie z.B. der Stuttgarter Neuronale Netze Simulator (SNNS [Zell1994]), oder kommerziell erhältliche Klassenbibliotheken an. Einen Überblick über Produkte, die in der Prozessautomatisierung eingesetzt werden, gibt [Sanari1998]. Der Nachteil dieser Produkte besteht allerdings darin, dass sie für die Lösung allgemeiner Probleme ausgelegt sind, die nicht a priori den Erfordernissen bei der Adaption von Walzmodellen entsprechen, wie sie in dieser Arbeit beschrieben werden. In den Prozessmodellen der beschriebenen Walzstraße werden daher spezielle neuronale Netze eingesetzt, die mit der Funktionsbibliothek SIANS [Broese1998] entwickelt wurden. Diese Software unterstützt speziell für die Anforderung in der Stahlindustrie optimierte Netztopologien.

Aber auch die C-Software SIANS besitzt die oben beschriebenen Schwächen, denn neben der fehlenden Objektorientierung, die die Erweiterbarkeit einschränkt, stellt ein fehlendes durchgängiges Konzept den größten Nachteil dar. SIANS ist lediglich ein Simulator für neuronale Netze und kann nicht als Basis für die Modellierung von hybriden Prozessmodellen genutzt werden. Daher wurde bereits vor einigen Jahren mit der Entwicklung einer objektorientierten Variante namens SIANS++ begonnen, die diese Schwachpunkte beseitigen soll. Teile des Konzeptes der neuronalen Netze dienen als Grundlage für die Neuro-Pakete im weiter unten beschriebenen Entwicklungsrahmen.

3.3 Konzept des Entwicklungsrahmens

Das in den nächsten Abschnitten beschriebene Rahmenwerk zur Umsetzung von Prozessmodellen im Walzwerk verfolgt im Wesentlichen vier Ziele:

1. Schaffung eines softwaretechnischen Regelwerkes und einer Umgebung, mit der mit relativ einfachen Mitteln beliebig komplexe Prozessmodelle erzeugt werden können.

2. Bereitstellung eines Verfahrens, mit dem sequentielle und parallele Modellaufrufe organisiert und vor allem notwendige Datenkonvertierungen durchgeführt werden können.

3. Definition einer allgemeinen Datenstruktur für Walzwerke und Walzmodelle mit integrierter Datenverwaltung.

4. Realisierung von eindeutigen Schnittstellen zu Diagnose- und Visualisierungssystemen.

Um diese vier Ziele in der softwaretechnischen Umsetzung eindeutig voneinander zu trennen, wird ein 4-Schichtenmodell verwendet, das sich aus einer Daten-Schicht, einer Algorithmik-Schicht (Modell), einer Workflow-Schicht (Prozess) und einer Visualisierungsschicht zusammensetzt. Nachfolgend wird vor allem auf die softwaretechnische Umsetzung eines Modells und eines Prozesses eingegangen. Das Prinzip der Speicherung der Prozessdaten geht auf [Wick1996] zurück und ist daher nur kurz am Ende dieses Kapitels beschrieben. Die Verknüpfung von Modell und Diagnose wird in Kapitel 6.1 erläutert.

Das Framework ist objektorientiert entwickelt und bietet vor allem Vorteile hinsichtlich der Vereinheitlichung und Wartbarkeit. Zudem lassen sich so mit Verknüpfungsobjekten, die nachfolgend auch vereinfacht Prozesse genannt werden, komplexe Vorgänge wie z.B. ein vollständiges Walzspaltmodell modellieren.

Das gesamte Framework setzt sich aus einer Reihe von Einzelmodulen zusammen, die sich in folgende Gruppen einteilen lassen:

- Basis
- Prozessdaten
- Allgemeine Modelle
- Spezielle Abbildungen
- Neuronale Netze
- Training und Konfiguration von Abbildungen
- Walzmodelle
- Prozesse

3.4 Basis

Das Modul Basis enthält grundlegende Klassen, die von jedem Prozessmodell benötigt werden und im Allgemeinen Bestandteil der verwendeten Programmier-

sprache sind. Diese Klassen werden um Konzepte zur Fehlerbehandlung, Persistenz und Verteilung ergänzt. Das Modul teilt sich dabei in folgende Unterpakete auf:

- Das Paket *System* enthält alle Basisbibliotheken der verwendeten Programmiersprache; für C++ sind dies Teile der Standard Library und der Standard Template Library (STL). Für Programmiersprachen, die diese Funktionalität nicht enthalten, muss das Paket entsprechend selbst implementiert werden.
- Das Paket *Base* definiert die Basisklassen, von denen jede weitere Klasse des Frameworks abgeleitet wird.

- Das Paket *Datatypes* enthält alle Datentypen, die nur zum transienten Aufnehmen von Daten zählen. Dies sind elementare Typen, Kollektionen. An elementaren Datentypen werden verschiedene Integertypen und Gleitkommazahlen unterstützt, außerdem stehen Datentypen für Zeichen, Strings und verschiedene Datums- und Zeitformate zur Verfügung. Es werden vier Kollektoren unterschieden: *Varray*, *List*, *Bag* und *Set*. Zur Iteration wird die Klasse *Iterator* verwendet.

- Das Paket *Error* enthält alle Klassen und Aufzählungstypen, die zur Ausnahmebehandlung benötigt werden.

- Das Paket *Streams* erweitert den aus C++ bekannten Mechanismus der Streams zur einfachen Realisierung der Persistenz der Daten und Abbildungen. Alle Datenhaltungsklassen und alle Abbildungen sind von der Klasse *Streamable* abgeleitet. Diese Klasse stellt Mechanismen zum Lesen und Schreiben der Daten bereit. Das Ziel der Daten ist theoretisch beliebig; Ziel kann im einfachsten Fall eine Datei sein, aber auch das verschlüsselte Versenden der Daten zu Diagnosezwecken über das Internet ist denkbar. Die Klasse Streamable dient als Basisklasse für alle nicht-elementaren Datentypen, die auf den Streams lesbar und schreibbar sein sollen. Sie stellt die grundlegenden Mechanismen bereit, so dass bei Neuimplementierung einer Klasse nur noch minimale Arbeit entsteht.

- Das Paket *Database* erweitert das Paket Streams um eine Methodik zur Speicherung aller Daten in einer Datenbank. Die Realisierung folgt dabei dem ODMG-Standard [Cattel1996]. Die Klasse *Database* ist eine Implementierung der ODMG-Klasse *d_database*. Sie implementiert Methoden zum Öffnen und Schließen der Datenbank und realisiert außerdem Methoden zum Auffinden und Umbenennen von Objekten in der Datenbank. Die Klasse *Transaction* stellt eine Implementierung der ODMG-Klasse *d_transaction* dar und realisiert das Transaktionsmanagement mit den wichtigsten Operationen einer Transaktion: begin, commit und abort. Nähere Informationen sind [Wick1996] zu entnehmen.

- Das Paket *Interprocess* enthält zwei Pakete, die die Kommunikation und die Nebenläufigkeit von Prozessen sicherstellen.

3.5 Prozessdaten

Das Modul Prozessdaten definiert Datentypen, die vor allem für Prozesse und Prozessmodelle benötigt werden. Die Klasse *Vector* ist die Basisklasse aller Vektoren und implementiert ein dynamisch veränderbares Feld aus beliebigen elementaren Datentypen aus dem Paket Datatypes in Form eines Zeilenvektors oder Spaltenvektors; dabei stellt der Zeilenvektor *Row* die Basisklasse für alle Muster und Parameter dar. Die Klasse *Matrix* ist die Basisklasse aller Matrizen. Sie realisiert ein zweidimensionales, dynamisch veränderbares Feld aus beliebigen Datentypen und ist somit die Basisklasse für alle Mengen aus Datensätzen und Parametern. Die Klasse *Index* stellt ein dynamisch veränderbares Feld aus vorzeichenlosen Ganzzahlen dar. Instanzen dieser Klasse werden typischerweise dazu benutzt, einzelne Zeilenvektoren einer Matrix zu indizieren.

Grundsätzlich sind zwei Ausprägungen der Vektoren zu unterscheiden: *Pattern* (Muster – Messwerte, Berechnungsergebnisse, Pattern im Sinne von Ein- und Ausgaben einer Abbildung) und *Parameter* (z.B. Netzparameter, Gewichte). Die entsprechenden Datentypen sind für Einzeldaten (Pattern und Parameter) von der Klasse Vector bzw. für Datenmengen (*PatternSet* und *ParameterSet*) von Matrix abgeleitet. Einen Sonderfall stellen virtuelle Parameter dar. Diese werden genutzt, um Sichten auf reale Parameter zu realisieren, wodurch sich z.B. alle Netzparameter in einem realen Parameter speichern lassen. Die Unterscheidung geschieht dann mit virtuellen Parametern. Virtuelle Parameter sind in Form der Klasse *ParameterCursor* implementiert.

Die Ausprägung einer Variablen eines Vektors bzw. einer Matrix – also vor allem ihr Name und ihr Datentyp - lässt sich durch Methoden direkt setzen und lesen. Komfortabler lässt sich die Initialisierung allerdings durchführen, wenn ein XML-Schema übergeben wird. Im hier betrachteten Fall handelt es sich bei Variablen im Allgemeinen um Prozessdaten, die global definiert sind. Auf den Zusammenhang zwischen Prozessdaten und Prozessen wird in Abschnitt 3.13 näher eingegangen.

Es stehen Methoden für den lesenden und schreibenden Zugriff mit Feldnamen zur Verfügung. Außerdem ist der Zugriff über einen Index möglich. Jeder Daten-Container bietet zudem die Möglichkeit des XML-Imports und –Exports. Durch Zuordnung eines XML-Mappings ist die persistente Speicherung in der Prozessdatenbank möglich.

Die interne Kommunikation im Framework geschieht nur in Sonderfällen über XML. Im Allgemeinen werden die Daten-Container direkt verwendet. Dadurch wird ein

unnötiges Parsen der XML-Datenströme und ein Mapping vermieden. Die externe Kommunikation zur Ferndiagnose und Ferninbetriebsetzung nutzt hingegen nur XML, um eine vollständige Entkopplung der Softwareprozesse zu erzielen. Ein Beispiel für ein XML-Schema ist in Abbildung 3.1 dargestellt.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:complexType name="ChemieT">
        <xs:sequence>
            <xs:element name="C" type="xs:float"/>
            <xs:element name="Si" type="xs:float"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="Chemie" type="ChemieT">
        <xs:annotation>
            <xs:documentation>Chemische Analyse des Bandes</xs:documentation>
        </xs:annotation>
    </xs:element>
</xs:schema>
```

Abbildung 3.1: Beispiel für ein XML-Schema

3.6 Allgemeine Modelle

3.6.1 Einleitung

Die Basis jeder Prozessautomatisierung sind die Prozessmodelle, die in sehr unterschiedlichen Ausprägungen existieren und unabhängig von ihrer Ausprägung im Allgemeinen als mathematische Funktion implementiert werden. Das Modell stellt also eine Abbildung dar, die einen Eingangsvektor auf einen Ausgangsvektor abbildet. Der Eingangsvektor setzt sich aus Anlagendaten, Messwerten, Werkstückdaten und Zwischenergebnissen anderer Modelle zusammen, der Ausgangsvektor enthält neben Zwischenergebnissen die Voreinstellungen für Stellglieder.

Neben der Voreinstellung besitzen Prozessmodelle meist noch die Aufgabe, die im Prozess auftretenden Fehler zu diagnostizieren. Um diese Fehler bestimmen zu können, wird zumindest die partielle Ableitung des Modells nach dem Ausgangsvektor benötig. Mit Hilfe dieser Ableitung können dann adaptive Modelle die notwendige Änderung ihrer internen Modellparameter bestimmen.

In einigen Fällen muss aus den Ausgangsgrößen der Eingangsvektor bestimmt werden; dann werden inverse Modelle verwendet.

Ein Prozessmodell besteht demnach immer aus einem Vorwärtsalgorithmus, einer Rückwärtspropagierung und ggf. aus einem inversen Modell. Nachfolgend wird ein Objekt mit diesem Verhalten kurz als *Modell* bezeichnet.

Modelle lassen sich grob in drei Kategorien einteilen:

- Einfache Modelle besitzen nur Methoden für den Vorwärtsaufruf und das Berechnen der partiellen Ableitung. Im Framework besitzen sie die gemeinsame Basisklasse *Model*.
- Umkehrbahre Modelle besitzen zusätzlich zu einfachen Modellen eine Umkehrfunktion. Im Framework besitzen sie die gemeinsame Basisklasse *ReversibleModel*.
- Modelle, die ausgehend von vorangegangenen Modellfehlern interne Parameter ändern können, werden als adaptive Modelle bezeichnet. Im Framework besitzen sie die gemeinsame Basisklasse *AdaptiveModel*.

Abbildung 3.2 zeigt ein Klassendiagramm der Modell-Basisklassen des Frameworks.

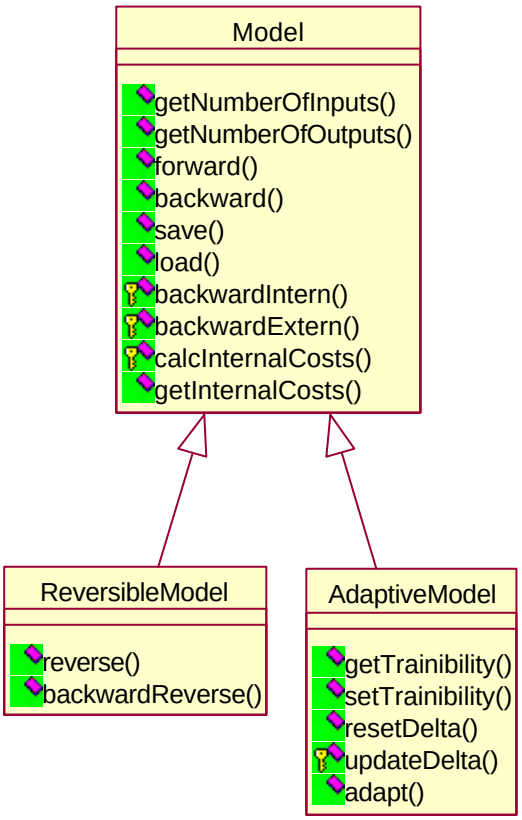


Abbildung 3.2: Klassendiagramm der Modell-Basisklassen

Die zentrale Klasse im Framework ist ein Modell (*Model*). Als grundlegendste Eigenschaften eines Modells sind die Vorwärtspropagation eines Musters und die Rückwärtspropagation eines Gradienten anzusehen. Die eigentliche Rückwärtspropagation wird durch zwei Funktionen realisiert. Hintergrund dabei ist die Überlegung, dass der zu berechnende Gradient im Kontext von Prozessmodellen oft

zum Training eines neuronalen Netzes verwendet werden soll und aus einer mehr oder weniger komplexen Fehlerfunktion berechnet wird. Das Framework unterstützt das Konzept einer komplexen Fehlerfunktion durch das Mittel des internen Fehlers und damit eines internen Gradienten in einem Modell.

Betrachtet man ein Modell, so ergibt sich allgemein bei einem angelegten Input I der Gesamtfehler E als Summe aus dem in der Abbildung entstehenden internen Fehler E_i und dem aus dem Output berechneten externen Fehler E_e:

$$E = E_i + E_e \qquad (3.1)$$

Somit ergibt sich der Gradient des Gesamtfehlers nach den Inputs als:

$$\frac{\partial E}{\partial I} = \frac{\partial E_i}{\partial I} + \frac{\partial E_e}{\partial O} \cdot \frac{\partial O}{\partial I} \qquad (3.2)$$

Für umkehrbare Modelle, also Funktionen im mathematischen Sinne, steht die Klasse *ReversibleModel* zur Verfügung. Der Mechanismus für die Umkehrfunktion ist dabei analog zur Vorwärtspropagation implementiert.

Die Basisklasse aller adaptiven bzw. trainierbaren Modelle stellt die Klasse *AdaptiveModel* dar. Ein neuronales Netz ist somit ein Spezialfall eines trainierbaren Modells. Generell wird davon ausgegangen, dass ein trainierbares Modell eine Reihe von adaptierbaren Parametern besitzt, die mit Hilfe einer Änderungsinformation, einem „Delta“, durch eine Adaptionsmethode modifiziert werden können. Dieses Delta wird durch die virtuelle Methode *calculateDelta*, welche den Wert einer Fehlerfunktion übergeben bekommt, bestimmt. Mittels der virtuellen Methode *resetDelta* wird die Änderungsinformation zurückgesetzt und die eigentliche Parameteradaption durch die virtuelle Methode *adapt* ausgeführt.

Durch die Methoden *calculateDelta* und *adapt* wird eine additive Akkumulation von Parameteränderungen und ein Ausführen der Änderungen nach mehreren Lernschritten, wie es z.B. beim Batchlernen mit MLPs auftritt, ideal unterstützt. Trotzdem lassen sich auch weitaus kompliziertere Verfahren, bei denen z.B. die Änderung nicht aufsummiert werden kann oder die Parameteränderung an sich nicht additiv ist, mit dieser Methodik realisieren.

3.6.2 Die Basisklasse Model

Die Klasse *Model* ist die abstrakte Basisklasse aller Modelle. Sie besitzt im Wesentlichen zwei Methoden: forward und backward.

- forward – beschreibt den Vorwärtsaufruf eines Modells. Der Algorithmus wird ausgeführt.

- backward – beschreibt die partielle Ableitung eines Modells nach dem Ausgang. Diese Methode wird vor allem für die Diagnose und die Adaption benötigt.

Da es sich bei der Klasse *Model* um eine abstrakte Klasse handelt, kann sie nicht direkt instanziert werden.

3.6.3 Die Klasse ReversibleModel

Die Klasse ReversibleModel ist von Model abgeleitet und ist wie ihre Basisklasse abstrakt. Sie erweitert die Funktionalität eines Modells um den Begriff der Umkehrfunktion und besitzt eine zusätzliche Methode:

- *reverse* – beschreibt den inversen Aufruf eines Modells. Es wird also die Umkehrfunktion des Algorithmus aufgerufen.

Da es sich bei der Klasse ReversibleModel um eine abstrakte Klasse handelt, kann sie nicht direkt instanziert werden.

3.6.4 Die Klasse AdaptiveModel

Die Klasse AdaptiveModel ist von Model abgeleitet und ist wie ihre Basisklasse abstrakt und erweitert ein Modell zu einem adaptiven Modell. Die Klasse Adaptive-Modell besitzt im Wesentlichen drei zusätzliche Methoden: *adapt*, *calculateDelta*, *resetDelta*. Die Parameteränderung, die charakteristisch für ein adaptives Modell ist, wird in einem Fehlervektor Delta abgelegt.

- adapt – beschreibt den Vorgang, der zur Adaption eines Modells führt.
- calculateDelta – beschreibt die Berechnung des Fehlervektors Delta.
- resetDelta – setzt den Fehlervektor wieder auf null zurück.

3.7 Spezielle Abbildungen

Das Paket *Abbildungen* enthält eine Reihe einfacher mathematischer Abbildungen, die häufig Bestandteil komplexerer Modelle sind. Dies sind zum einen Normierungen der Eingänge und zum anderen verschiedene Filter.

3.7.1 Identität

Die Identität *Identity* realisiert eine leere Abbildung, d.h. die Ausgabe ist gleich der Eingabe. Ihr Verwendungszweck sollte hauptsächlich im flexiblen Ein- und Ausschalten von Teilabbildungen in einer zusammengesetzten Abbildung zur Laufzeit bestehen. Die Abbildung ist umkehr- und nicht trainierbar, deshalb wird die Klasse von *ReversibleModel* abgeleitet.

3.7.2 Normierung

Verschiedene Einflussgrößen eines Prozessmodells besitzen im Allgemeinen sehr unterschiedliche Größenordnungen. Statistische Algorithmen und vor allem neuronale Netze liefern jedoch dann optimale Ergebnisse, wenn Erwartungswerte und Standardabweichung jedes Eingangs in der gleichen Größenordnung liegen, daher werden die Eingänge vor der Weiterverarbeitung normiert. Das Framework bietet folgende Filter:

- Mininimum-Maximum-Normierung
- Mittelwert-Standardabweichung-Normierung
- Mittelwert-Standardabweichung-Korrellation-Normierung
- Lineare Normierung

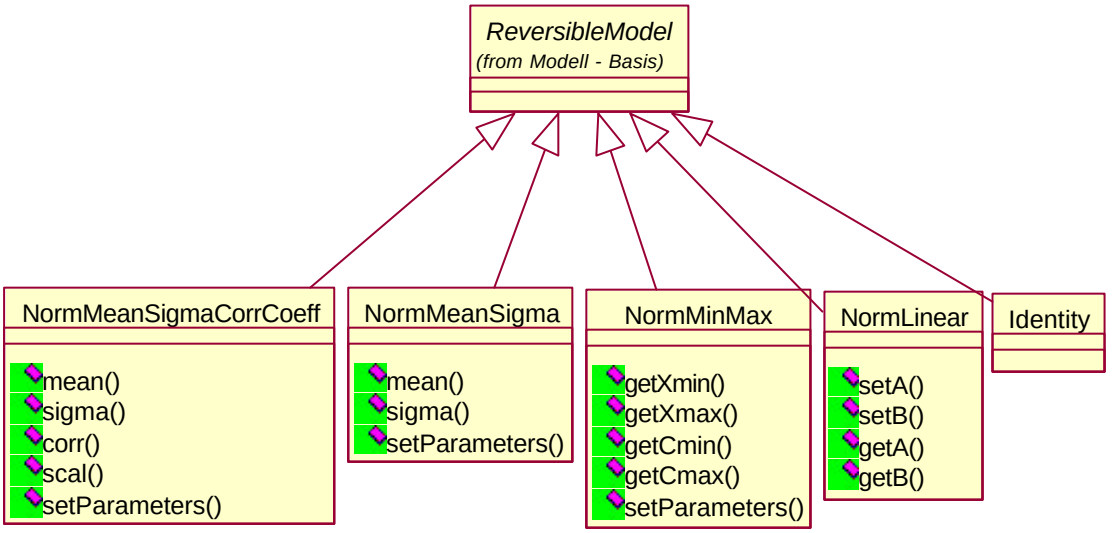


Abbildung 3.3: Normierung

Die Minimum-Maximum-Normierung *NormMinMax* führt elementweise eine lineare Normierung durch, dabei werden die Parameter so gewählt, dass ein gegebener Wertebereich (z.B. zwischen dem Minimum und dem Maximum einer Datenmenge) auf einen definierten Bereich (z.B. –1,0 bis 1,0) abgebildet wird. Es erfolgt jedoch weder eine Begrenzung der Inputs noch eine Begrenzung der Outputs. Liegt also im angeführten Beispiel der Eingangswert außerhalb des Minimum-Maximum-Bereiches, so wird er auch auf einen Wert außerhalb des Bereiches (-1,0;1,0) abgebildet. Die Funktion ist umkehrbar, aber nicht trainierbar, deshalb wird die Klasse von *ReversibleModel* abgeleitet.

Die Mean-Sigma-Normierung *NormMeanSigma* führt eine elemtweise lineare Transformation durch, die eine Datenmenge mit einer Gaußverteilung auf eine mittelwertfreie Datenmenge mit der Standardabweichung 1 abbildet. Die Funktion

ist umkehrbar, aber nicht trainierbar, deshalb wird die Klasse von ReversibleModel abgeleitet.

Die Mean-Sigma-Normierung mit Korrelationskoeffizient *NormMeanSigmaCorrCoeff* führt eine elementweise lineare Transformation durch, welche einer Mean-Sigma-Normierung entspricht, deren Ergebnis noch mit einem konstanten Faktor multipliziert wird. Dieser Faktor entspricht dem Betrag der Korrelation des jeweiligen Inputs mit dem ersten Target-Wert. Die Funktion ist umkehrbar, aber nicht trainierbar, deshalb wird die Klasse von *ReversibleModel* abgeleitet.

Die lineare Normierung *NormLinear* führt eine lineare Transformation der Form

$$y = A\,x + B \qquad (3.3)$$

durch und eignet sich somit für eigene Normierungen. Die Funktion ist umkehrbar, aber nicht trainierbar, daher wird die Klasse von *ReversibleModel* abgeleitet.

3.7.3 Selektoren

Der Inputselektor *InSelect* ermöglicht eine Auswahl der zu verwendenden Eingänge zur Laufzeit. Er gestattet es, aus einer vorgegebenen Menge einzelne Eingänge zu selektieren, ohne ihren Wert zu verändern. Er wird vor allem bei hybriden oder kaskadierenden Modellen benötigt, wobei jedem Element des Ausgangs genau ein Element des Eingangs zugeordnet wird. Einen Spezialfall des Inputselektor stellt die weiter oben beschriebene Identität dar. Die Abbildung ist weder umkehrbar noch trainierbar; somit wird die Klasse von *Model* abgeleitet.

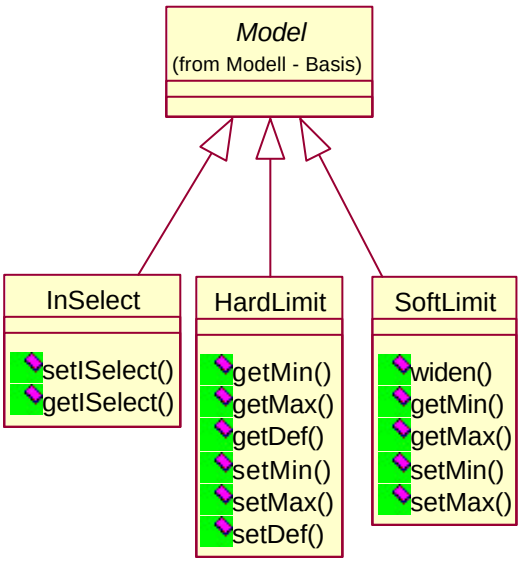


Abbildung 3.4: Filter und Selektoren

Der Inputselektor lässt sich auch als Outputselektor nutzen, um es Prozessmodellen mit mehreren Ausgängen zu ermöglichen, einzelne Outputs zu selektieren. Zur besseren Lesbarkeit existiert jedoch eine identische Klasse *OutSelect*, die von InSelect abgeleitet ist.

3.7.4 Filter

Das Framework nutzt Filter, um unplausible Eingaben zu verhindern. Die Filterparameter können dabei Ergebnis der Vorverarbeitung sein oder sich erst während der Laufzeit ergeben.

Der Begrenzer *HardLimit* bildet analog der Identität seinen Eingang auf seinen Ausgang ab. Zusätzlich wird abgesichert, dass jeder der Ausgänge in einem definierten Bereich liegt, Eingänge außerhalb dieses Bereiches werden angepasst. Die Abbildung ist nicht umkehrbar und nicht trainierbar, deshalb ist *HardLimit* von Model abgeleitet.

Im Gegensatz zum harten Begrenzer, der seine Grenzen vorgegeben bekommt, kann der weiche Begrenzer *SoftLimit* seine Grenzen während der Laufzeit des Modells ändern und den präsentierten Daten anpassen. Dazu muss die virtuelle Methode *widen* bei der Implementierung überschrieben werden. Die Abbildung ist nicht umkehrbar und nicht trainierbar, deshalb ist *SoftLimit* von *Model* abgeleitet.

3.8 Neuronale Netze

Ein Ziel dieser Arbeit ist es, mit statistischen Verfahren das Walzergebnis zu optimieren. Ein viel versprechender Ansatz sind in diesem Zusammenhang neuronale Netze, die in Kapitel 2.3 erläutert sind. Die folgenden Abschnitte zeigen, wie sich diese Netze – in Form von Modellen – darstellen und damit in das Framework integrieren lassen.

3.8.1 Allgemeines

Das Paket *Neuronale Netze* enthält Klassen zur Realisierung von neuronalen Netzen, wobei die in Abschnitt 2.3 beschrieben ADALINEs, MLPs und RBF-Netze unterstützt werden, außerdem sind zur Clusteranalyse auch LVQ und SOM realisiert. Die Implementierung der dort angegebenen Algorithmen orientiert sich dabei an [Broese1998] und [Zell1994]; Anregungen sind bei [Masters1993] und [Rogers1996] entnommen.

Eine eigene Implementierung der neuronalen Netze ist nicht grundsätzlich notwendig; es ist auch die Einbindung von verschiedenen kommerziellen Klassenbibliotheken für neuronale Netze möglich, wenn diese den beschriebenen

Mechanismus der Trennung von Vorwärtsaufruf, partieller Ableitung, Delta-Bestimmung und Gewichtsänderung unterstützen.

Zur Implementierung eines im Modul Neuronale Netze noch nicht realisierten Netztyps muss die Klasse AdaptiveModel abgeleitet werden und die wichtigsten Methoden forward, backward, updateDelta, resetDelta, und adapt müssen überschrieben werden.

Eine weitere Alternative zur Realisierung komplexer neuronaler Netze ist die Verschaltung elementarer Netztypen zu so genannten *hybriden neuronalen Netzen*. In der Praxis hat sich beispielsweise gezeigt, dass eine Kombination aus RBF-Netzen und ADALINEs oder MLPs gerade im Bereich der Prozessautomatisierung von Walzwerken Erfolg versprechend ist.

3.8.2 ADALINE

Die Klasse *ADALINE* realisiert das in Kapitel 2.3.2 beschriebene ADALINE. Sie ist als Spezialfall des im nächsten Abschnitt beschriebenen MLPs implementiert. Diese Abbildung ist trainierbar, jedoch nicht umkehrbar und daher von *AdaptiveModel* abgeleitet.

3.8.3 MLP

Die Klasse *MLP* realisiert das Multi Layer Perceptron (MLP). Dieser Netztyp wurde bereits ausführlich in Kapitel 2.3.3 beschrieben; weitere Informationen zur Simulation finden sich u.a. in [Zell1994]. Diese Abbildung ist trainierbar, jedoch nicht umkehrbar, deshalb wird die Klasse von *AdaptiveModel* abgeleitet.

3.8.4 RBF

Die Klasse *RBF* stellt ein neuronales Netz mit radialen Basisfunktionen (RBF) mit Normierung zu Eins in der verdeckten Schicht und lokalen linearen Experten in der Ausgangsschicht zur Verfügung. Die lokalen Experten werden durch einen globalen Experten ergänzt, der die Generalisierungseigenschaften bei seltenen Daten verbessern soll. Ein weiterer globaler Experte übernimmt die Funktion einer unterlagerten linearen Regression. Die Abbildung ist trainierbar, jedoch nicht umkehrbar, deshalb wird die Klasse von *AdaptiveModel* abgeleitet.

3.8.5 LVQ und SOM

Die Klasse *LVQ* stellt die Basisklasse für alle LVQ-Netze dar. Die verschiedenen in den Anhängen E.1.1 (LVQ1), E.1.2 (LVQ2.1) und E.1.3 (LVQ3) beschriebenen Varianten werden durch entsprechende Ableitungen realisiert. Die SOM, die in Anhang E.2 beschrieben sind, werden durch die Klasse *SOM* implementiert. Diese Abbildungen

sind alle trainierbar, jedoch nicht umkehrbar, deshalb werden sie von *AdaptiveModel* abgeleitet.

3.9 Komplexe Prozessmodelle

Die in den vorigen Kapiteln beschriebenen Prozessmodelle können als elementare Prozessmodelle angesehen werden, die als Bausteine in neuen Prozessmodellen verwendet werden können. Die Realisierung komplexerer Modelle kann dabei auf zwei Wegen geschehen: Entweder kann ein bestehendes Modell abgeleitet werden, und die wichtigsten Methoden können überschrieben werden. In diesem Fall kann aber das Know-How, das bereits im Framework steckt, nur eingeschränkt genutzt werden. Daher kann andererseits ein neues Modell auch aus der Aggregation existierender Modelle implementiert werden.

Mit Hilfe dieser beiden Methoden lassen sich nahezu beliebig komplexe Prozessmodelle entwerfen. Einen Sonderfall stellen in diesem Zusammenhang die so genannten hybriden Prozessmodelle dar, die einen physikalisch-mathematischen Kern mit einer adaptiven Korrektur ergänzen (Abbildung 3.5). Auf verschiedene Varianten von hybriden Prozessmodellen wird später noch näher eingegangen.

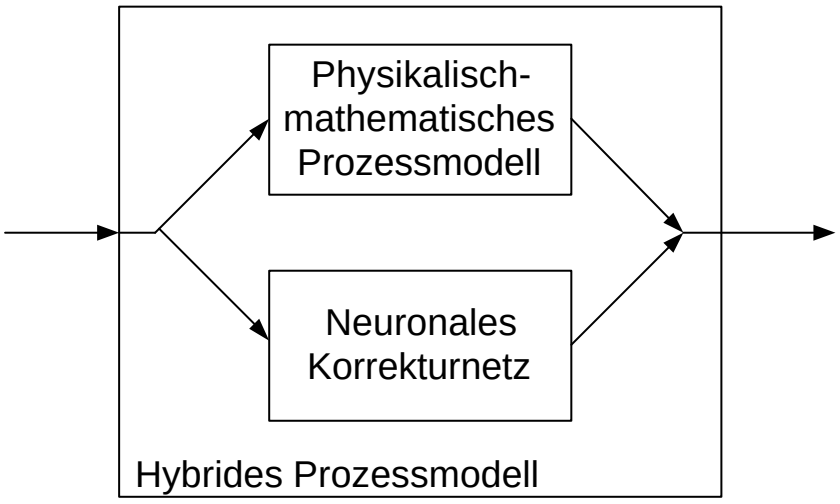


Abbildung 3.5: Beispiel für ein hybrides Prozessmodell

Eine weitere Möglichkeit des Frameworks besteht darin, durch Zusammenschalten vorhandener neuronaler Netze neue Netzstrukturen zu entwerfen. Im praktischen Einsatz in der Stahlindustrie hat sich die Kombination aus MLP und RBF-Netz bewährt; diese ist exemplarisch in Abbildung 3.6 dargestellt.

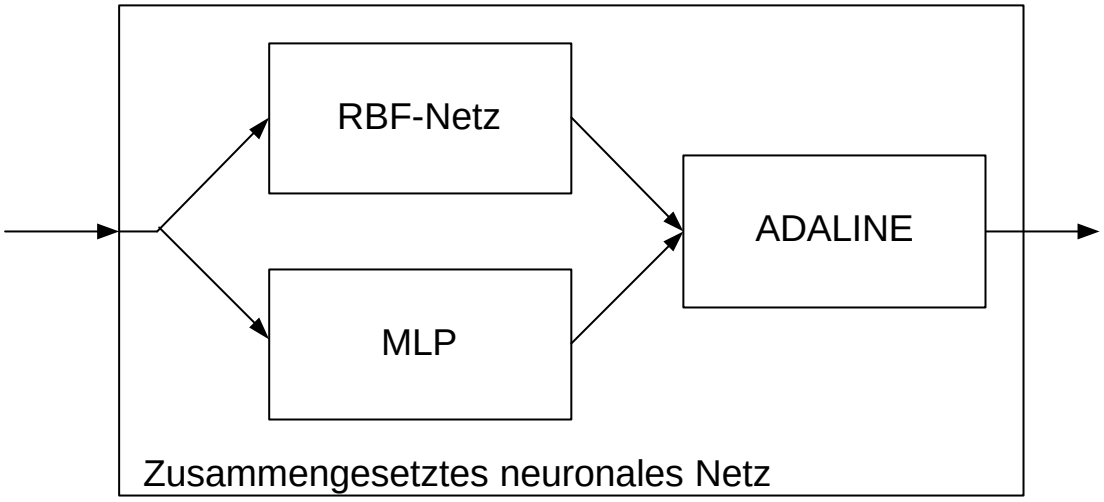


Abbildung 3.6: Beispiel für ein zusammengesetztes neuronales Netz

3.10 Training von adaptiven Abbildungen

Die beim Training einer adaptiven Abbildung, speziell eines neuronalen Netzes, ablaufenden Vorgänge sind einander immer sehr ähnlich, deshalb stellt das Framework die Klasse *Trainer* zur Verfügung, mit deren Hilfe das Training einer Abbildung nach Bereitstellung entsprechender Daten auf sehr einfache Weise ausgeführt werden kann. Mit einer Instanz der Klasse Trainer kann sowohl ein Online- als auch ein Offline-Training durchgeführt werden. Neben dem Training besitzt der Trainer die wichtige Aufgabe, die Ein- und Ausgangsdaten zu organisieren. Die Konfiguration geschieht dabei im Allgemeinen mit XML-Schemata, die in Abschnitt 3.13 beschrieben werden.

3.10.1 Ablauf des Trainings

Das Training eines neuronalen Netzes verläuft immer nach folgendem Schema: Zunächst werden die zur Verfügung stehenden Trainingsdaten in die Gruppen *Training*, *Test* und *Validierung* eingeteilt. Beim Training eines neuronalen Netzes werden zunächst alle Netzparameter initialisiert, wobei die Art der Initialisierung der Netzgewichte vom Netztyp abhängt. Während beim MLP den Netzgewichten nur zufällige Werte zugewiesen werden, müssen bei RBF-Netzen die Zentren in der verdeckten Schicht und die Breiten der Gaußglocken ausgeprägt werden. Für das Vortraining der RBF existiert eine eigene Trainerklasse *RBFTrainer*, die von *Trainer* abgeleitet ist und die die drei Clusterverfahren k-Means, AKM und Neural Gas, die in Kapitel 2 beschrieben sind, implementiert.

Anschließend wird wiederholt ein Eingangsmuster der Trainingsmenge angelegt, die Netzausgabe berechnet, mit einem gewünschten Ausgang verglichen und daraus ein Fehler bestimmt. Anhand dieses Fehlers wird eine Parameteränderung ermittelt und diese sofort oder nach der Präsentation weiterer Muster gemeinsam mit deren

Änderungen ausgeführt. Nach einem vollständigen Durchlauf über alle Werte der Trainingsmenge wird das Netz mit der Testmenge überprüft. Wenn das neue Netz ein besseres Resultat liefert als das bisher beste Netz, wird es gespeichert. Das Training endet nach Unterschreitung eines vorgegebenen Testfehlers oder nach einer Überschreitung der maximalen Epochenzahl. Nach Abschluss des Trainings wird das Netz mit Hilfe der Validierungsmenge nochmals überprüft. Dieser Ablauf ist noch einmal in Abbildung 3.7 dargestellt.

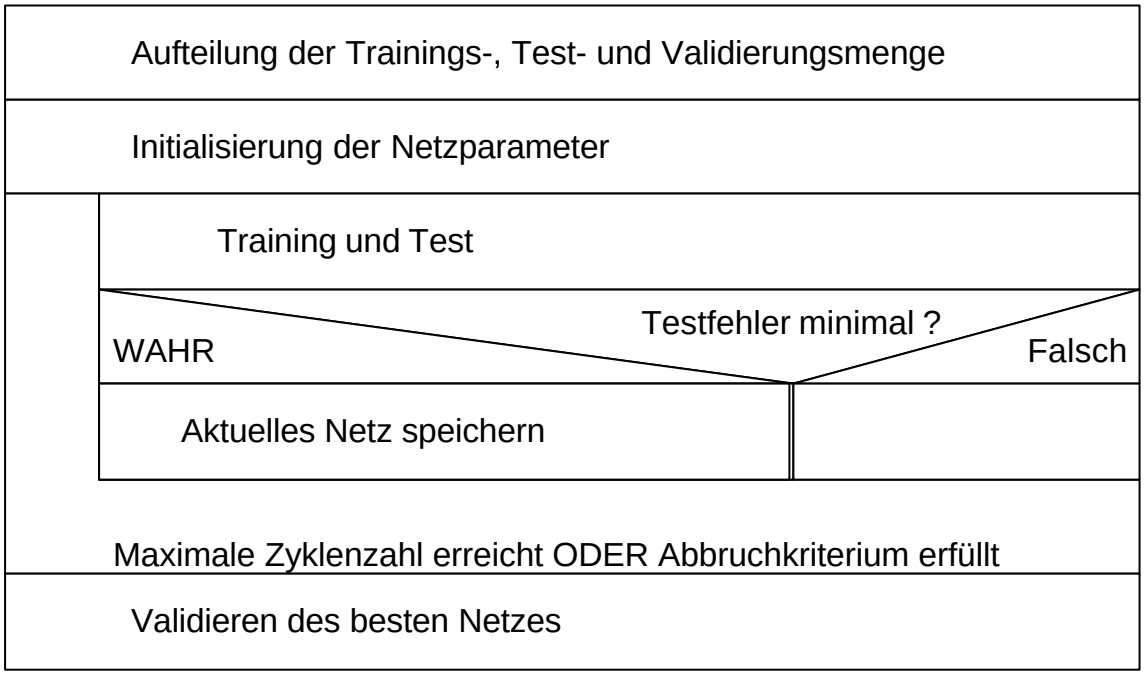


Abbildung 3.7: Ablauf des Netztrainings

3.10.2 Gütemaße für Trainingsfehler

Um den Fehler einer Prognose feststellen zu können, müssen die tatsächlichen Werte y^p mit den neuronal geschätzten Werten $\hat{y}^p$ verglichen werden. Dieser Fehler e^p wird als *Residuum* bezeichnet:

$$e^p = y^p - \hat{y}^p \qquad (3.4)$$

Der Trainings-, Test- und Validierungsfehler kann mit verschiedenen Gütemaßen festgestellt werden. Im Rahmen dieser Arbeit finden folgende Maße Anwendung:

$$MRE = \frac{1}{N}\sum_{p=1}^{N}\frac{\left|e^p\right|}{y^p} \qquad (3.5)$$

$$MAPE = \frac{100}{N}\sum_{p=1}^{N}\left|\frac{e^p}{y^p}\right| \qquad (3.6)$$

$$SSE = \sum_{p=1}^{N} \left(e^p\right)^2 \qquad (3.7)$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{p=1}^{N} \left(e^p\right)^2} \qquad (3.8)$$

Wenn nicht ausdrücklich von einem anderen Gütemaß die Rede ist, wird nachfolgend davon ausgegangen, dass der RMSE als Netzfehler verwendet wird.

3.10.3 Varianten des Trainings

Beim Training von adaptiven Abbildungen sind zwei Varianten zu unterscheiden: das *Single-Step-Training* und das *Multi-Step-Training*. Beim Single-Step-Training wird nach jeder Präsentation eines Trainingsmusters eine Parameteränderung durchgeführt. Der durch die Basisklasse AdaptiveModel geschaffene Mechanismus unterstützt jedoch eine additive Akkumulation von Parameteränderungen und ein Ausführen der Änderungen nach mehreren Lernschritten. Beim Multi-Step-Training werden der adaptiven Abbildung daher pro Adaption mehrere Muster präsentiert. Die Anzahl der Muster wird dabei als *Batch-Länge* bezeichnet. Das Zusammenfassen von verschiedenen Aktionen wird nachfolgend als *Batch-Training* bezeichnet.

3.10.4 Beispiele für das Training adaptiver Abbildungen

In Abbildung 3.8 ist ein typischer Trainingsverlauf für ein MLP dargestellt. Zu Beginn verringert sich sowohl der Fehler des Trainings- als auch des Testfehlers. Mit zunehmender Epochenzahl verändert sich der Testfehler kaum noch, während der Trainingsfehler weiter minimiert werden kann. Nach einer vorgegebenen Anzahl an Epochen wird das Training gestoppt. In Abbildung 3.9 ist ein weiterer typischer Trainingsverlauf eines MLPs dargestellt. Hier ist deutlich das so genannte *Generalisierungsproblem* bzw. *Overfitting* zu beobachten. Das Training des Netzes wurde nicht an der Stelle des minimalen Testfehlers – im Beispiel nach etwa 40 Epochen - beendet, sondern bis zur maximalen Epochenzahl weitergeführt. Daten der Trainingsmenge kann das Netz zwar sehr gut voraussagen, für unbekannte Daten waren die Ergebnisse aber vorher wesentlich besser. In einem solchen Fall muss der Trainer das Netz mit dem besten Testfehler wiederherstellen und als Ergebnis des Trainings zurückliefern.

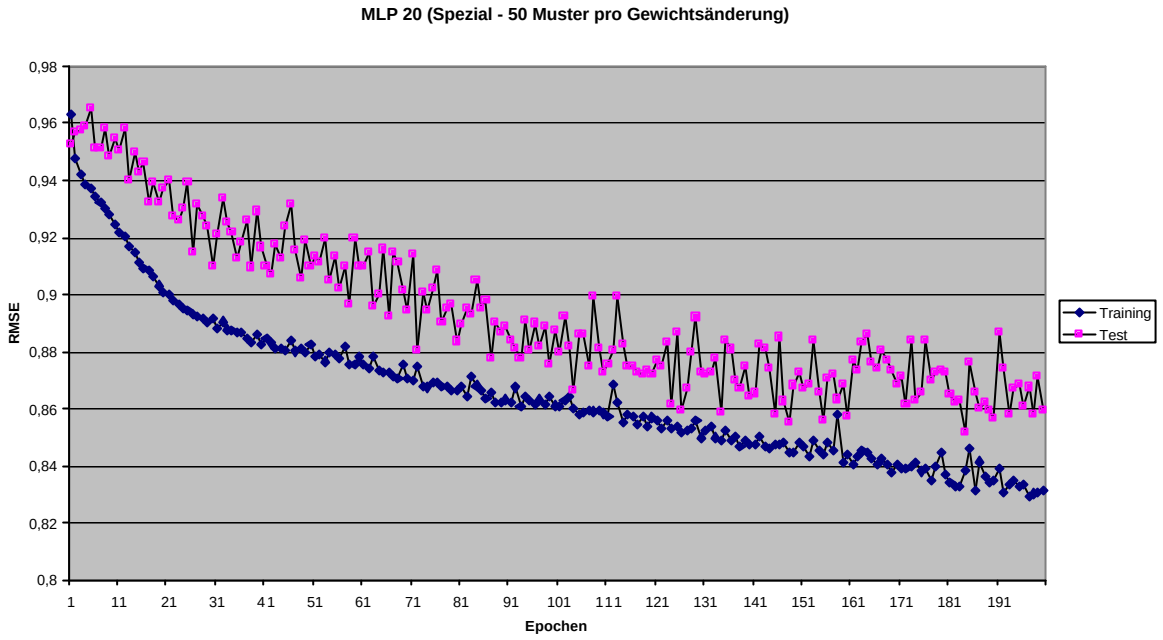


Abbildung 3.8: Abbruch des Netztrainings bei maximaler Epochenzahl

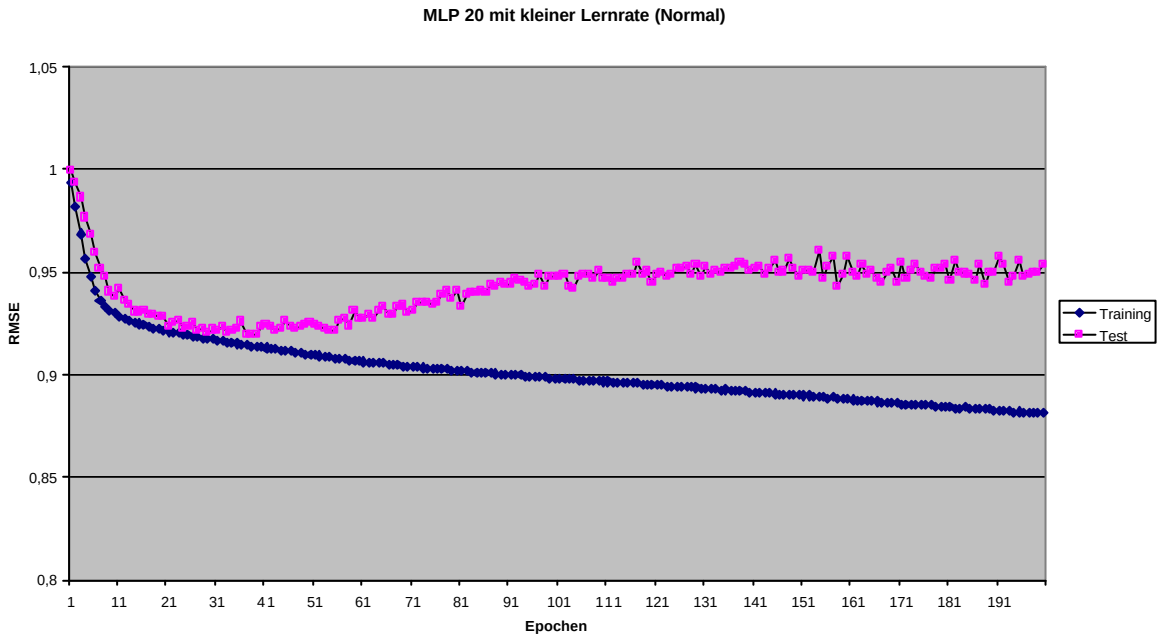


Abbildung 3.9: MLP mit Overfitting

3.11 Konfiguration

Das Modul Konfiguration enthält Konfigurationsklassen, die der strukturierten Verwaltung und Handhabung von Optionen, wie z.B. den Trainingsoptionen für ein MLP oder ein RBF oder auch deren Trainer, dienen. Sie sind von einer Basisklasse *Config* abgeleitet, welche eine Reihe von Methoden zur Verfügung stellt, mit deren

Hilfe sich auch neue Konfigurationsklassen nach einem einfachen Schema komfortabel und schnell implementieren lassen.

Der Import und Export der Konfigurationsinformationen geschieht üblicherweise mittels XML. Hierzu existiert ein eigenes XML-Schema, das alle Optionen, die zurzeit im Framework Verwendung finden, berücksichtigt.

3.12 Walzspaltmodelle

Das Paket *Walzspaltmodelle* enthält alle Modelle, die den Vorgang im Walzspalt beschreiben. Die umformtechnischen Grundlagen für diese Modelle sind ausführlich in den Kapiteln 2.2.4, 2.2.5 und 2.2.6 beschrieben. Alle Modelle sind durch Ableitung einer Klasse von einer bestehenden Modellklasse entstanden, dabei müssen im Allgemeinen die Methoden forward, backward und ggf. auch adapt überschrieben werden. Das Framework implementiert dabei nur grundlegende Modelle nach den analytischen Modellen, die in Kapitel 2 beschrieben sind. Wie bereits erwähnt, ist es gerade die Aufgabe des Frameworks, die Neuimplementierung eines einzelnen Walzspaltmodells zu vereinfachen und keine vollständigen Walzspaltmodelle vorzugeben.

Im Einzelnen besteht das Paket Walzspaltmodelle aus folgenden Klassen:

- Walzkraftmodell
- Bandtemperaturmodell
- Bandbreitungsmodell
- Walzerständermodell
- Walzenlagermodell
- Walzenbiegemodell
- Walzenabplattungsmodell
- Walzentemperaturmodell
- Walzenverschleißmodell

3.13 Definition der Prozessdatenstruktur

Die in den bisherigen Abschnitten beschriebenen Klassen stellen den algorithmischen Kern des Frameworks dar. Wie angedeutet, wird im Framework die Algorithmik von der Visualisierung, der Datenhaltung und dem Work-Flow getrennt, daher wird nachfolgend noch kurz auf Methoden zur Datenhaltung eingegangen.

3.13.1 Vorbemerkungen

Die Forschung zur Vorhersage der Walzgutqualität leidet immer unter einem Mangel an zuverlässigen und übertragbaren Messdaten. Die Herausforderungen und die

Steuerung wachsen schneller als das Wissen über die technischen Daten und deren Zusammenhänge. Laborversuche sind sehr teuer, und die Messergebnisse können nicht immer ohne Korrektur in die industrielle Nutzung übernommen werden. Zur zuverlässigen Vorhersage der Walzgutqualität ist neben der großen Menge gemessener Daten und einer guten Steuerung auch eine umfassende fachliche Erfahrung notwendig. Diese Erscheinung ist durch den Unterschied zwischen der Information und dem Wissen begründet. [CseMen1999]

Eine zielgerichtete Informationssammlung kann durch die Anwendung spezieller Methoden zur Datenakquisition angewandt werden. Das Data Mining ist ein Sammelbegriff für Methoden der Datenakquisition und –aufbereitung. Aus einer großen Datenmenge werden hierbei die Informationsdomäne detektiert und Zusammenhänge aufgezeigt, die für die Verbesserung einer Entscheidungssituation relevant sind.

Die Daten, die entsprechend den Anforderung der ISO 9000ff im Walzbetrieb gesammelt werden, können als Informationsquellen für die außerordentlich komplexen Prozesse genutzt werden. [LePiCs1999] Die in verschiedenen Ebenen der Rechnerhierarchie gespeicherten Datensammlungen sind so zusammenzuführen und einzuordnen (Data Warehouse), dass nachfolgend aus dieser Datenbank verdeckte Zusammenhänge zwischen den einzelnen Daten herausgefunden werden können.

3.13.2 Die Prozessdatenstruktur

Die aufgenommenen Daten lassen sich grob in drei Gruppen einteilen: Sollwerte, die im Allgemeinen vom Prozessleitrechner vorgegeben werden, Messwerte und Stellgrößen, die mit Hilfe der in der Abbildung 3.10 dargestellten Messgeräte ermittelt werden, und berechnete Modellergebnisse der Prozessmodelle.

Es lässt sich festhalten, dass sich ein Widerspruch zwischen dem Datenmangel in der Forschung und der Datenflut in der Produktion ergibt. Die moderne Informationstechnik erlaubt ein Aufnehmen und Speichern nahezu aller Daten, die während des Walzens anfallen. Diese Daten können mit Daten aus dem Labor, wie sie z.B. bei der Ermittlung der Korngröße oder mechanischer Eigenschaften entstehen, ergänzt werden. Zur Analyse ist es nun notwendig, die vorhandenen Daten in ein einheitliches Bezugssystem zu transformieren.

Alle Daten, die im Walzwerk anfallen, werden in einer Prozessdatenstruktur gespeichert, die an die jeweilige Straße angepasst werden muss. Die nachfolgend beschriebene Struktur orientiert sich an den Gegebenheiten des Bochumer Warmbreitbandwerkes.

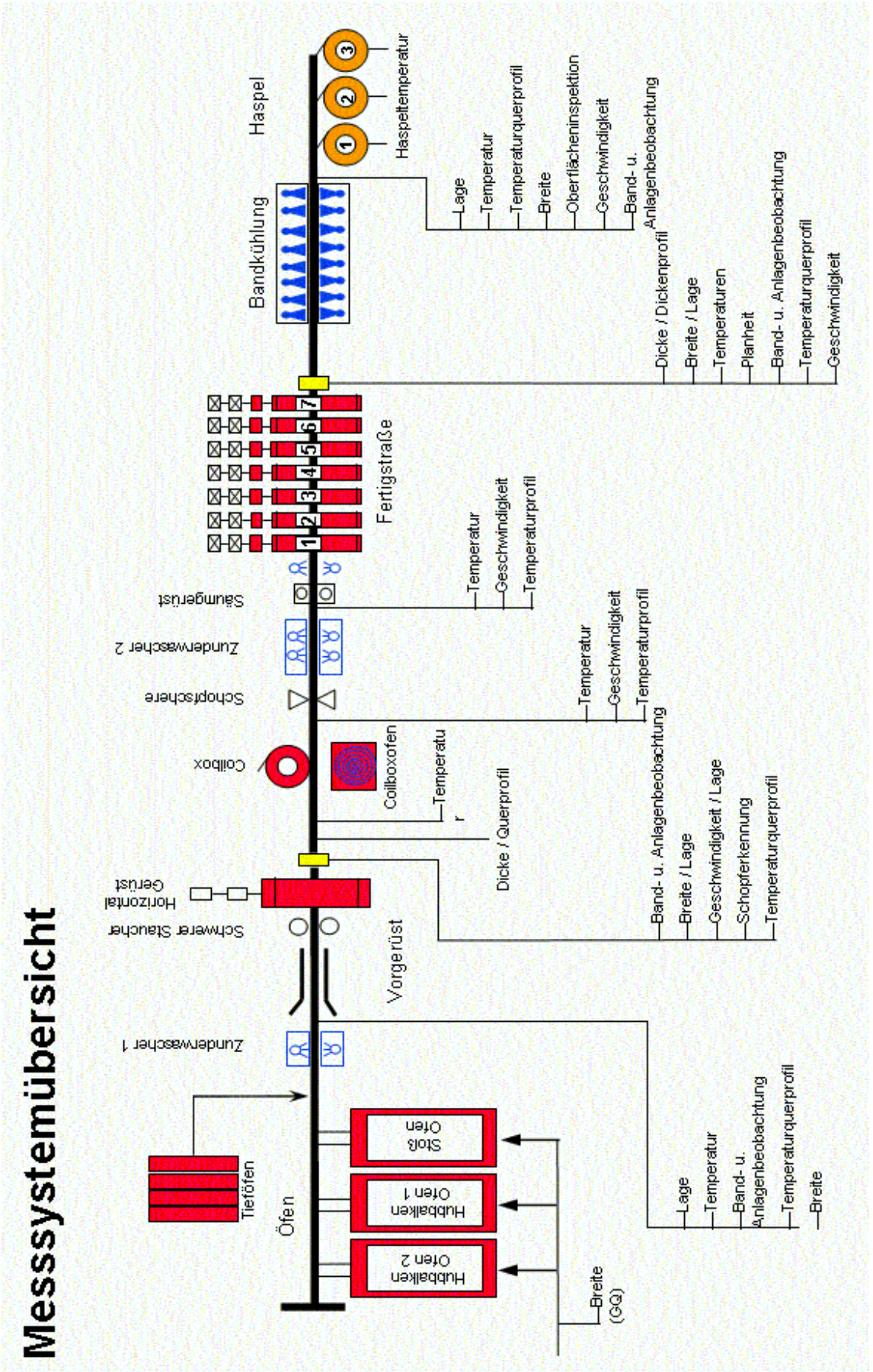


Abbildung 3.10: Messgeräte Walzwerk (aus [KHS 1995])

Die Prozessdatenstruktur setzt sich aus folgenden Komponenten zusammen:

- Anlagenkomponenten: Das Paket *Anlagenkomponenten* enthält alle Komponenten, die in einem Walzwerk zum Einsatz kommen können. Dies sind Messgeräte, Stellglieder und Nebenaggregate. Aus Vereinfachungsgründen sind einige Komponenten zu Teil- und Nebenanlagen zusammengefasst, die einzelnen Komponenten eines Warmbreitbandwerkes werden im Anhang erläutert. Diese Struktur lässt sich direkt in eine Datenstruktur übernehmen. Die Beschreibung aller Komponenten der Anlage ist an dieser Stelle von untergeordneter Bedeutung.

- Anlagenzustand: Das Paket *Anlagenzustand* enthält alle Klassen, die den Zustand der Anlage beschreiben; dies sind in erster Linie Stellgrößen und Messwerte.

- Prozesszustand: Das Paket *Prozesszustand* enthält alle Klassen, die den Zustand eines Prozesses beschreiben; dies sind vor allem Modellparameter und Zwischenergebnisse.

- Walzgut: Das Paket *Walzgut* enthält alle Klassen, die die Eigenschaften des Walzgutes vor, während und nach dem Walzen beschreiben.

Das Framework bietet:

- Wichtige Einrichtungen eines Walzwerkes

- Wichtige Modellausgänge von Prozessmodellen

- Eingangsgrößen der Stichplanberechnung

- Wichtige Messgrößen

Diese Komponenten liegen als Vorlagen im UML-Klassendiagramm vor. Für eine konkrete Straße müssen dann aus diesen Vorlagen alle konkreten Prozessdatenklassen definiert werden. Anschließend müssen alle Assoziationen und Kardinalitäten festgelegt werden.

3.13.3 Generierung des XML-Schemas

Ein in UML definiertes Datenmodell lässt sich eindeutig in ein XML-Schema umwandeln, wenn vorab eindeutige Abbildungsregeln festgelegt werden. In dieser Arbeit werden folgende Regeln verwendet:

1. Jede Klasse im Klassendiagramm wird auf einen *complexType* abgebildet. Für nicht-abstrakte Klassen wird außerdem ein Element dieses Typs angelegt.

2. Jede Vererbung im Klassendiagramm wird im XSD durch das Attribut *base* ausgedrückt.

3. Jeder nicht-elementare Datentyp im Klassendiagramm wird auf einen *complexType* im XSD abgebildet.

4. Jeder elementare Datentyp im Klassendiagramm wird auf einen *simpleType* im XSD abgebildet.

5. Jedes Attribut im Klassendiagramm wird auf ein untergeordnetes Element im XSD abgebildet. Über das Attribut *type* wird ihm ein Typ zugeordnet.

6. Zur eindeutigen Identifikation eines Objektes wird jede Klasse um ein Element ObjectID ergänzt.

7. Aggregationen und Assoziationen werden durch Listen aus ObjectIDs realisiert. Die beiden Attribute *minOccurs* und *maxOccurs* bestimmen dabei die Kardinalität.

Ausgehend von einer in UML vorliegenden vollständigen Prozessdatenstruktur kann dann mit Hilfe eines Generators das Datenbankschema, der Datenbankzugriff und ein XML-Schema erzeugt werden.

3.14 Prozesse

Prozesse verwalten einerseits die komplette Prozessdatenstruktur mit allen Sollwerten, Anlagendaten, Modelldaten und Messwerten. Andererseits verwalten sie alle assoziierten Prozessmodelle mit ihren jeweiligen Trainern und den jeweiligen Datenbeschreibungen. Prozesse besitzen einen Mechanismus zum Mapping der Datenstrukturen eines einzelnen Modells auf die Gesamtstruktur. Außerdem stehen Methoden zur Definition des Workflows, der Initialisierung und der Adaption zur Verfügung.

Im Framework werden Prozesse durch die Klasse Process implementiert, die von AdaptiveModel abgeleitet ist. In Abbildung 3.11 ist ein stark vereinfachtes Klassendiagramm der Klasse Process dargestellt.

3.14.1 Datenversorgung von Prozessmodellen

Die Prozessautomatisierung unterteilt den Prozess, der das Gesamtproblem beschreibt, üblicherweise in Teilprobleme, die durch Prozessmodelle gelöst werden. Diese Prozessmodelle müssen vom Prozess mit Daten versorgt werden und diesem Ergebnisse zurückliefern.

Zur Implementierung eines Prozessmodells bietet das Framework die beschriebenen Basisklassen für Abbildungen, neuronale Netze und einfache Walzmodelle. Außerdem bietet es Basisklassen für die Datenversorgung.

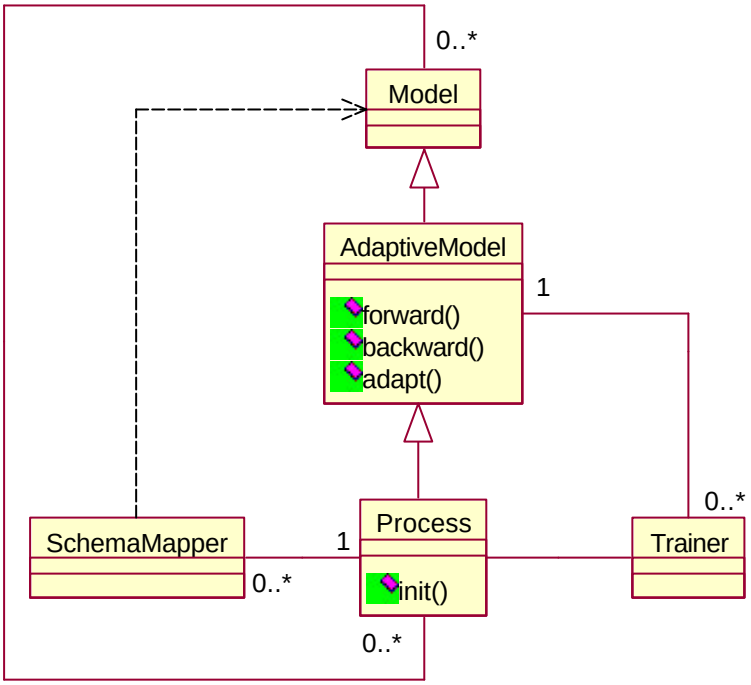


Abbildung 3.11: Die Klasse Process

Zur Realisierung eines konkreten Prozessmodells muss zunächst eine neue Modellklasse von einer beliebigen Modell-Basisklasse abgeleitet werden, dabei kommen die im vorigen Abschnitt beschriebenen Methoden zum Einsatz. Anschließend müssen die Eingangs- und Ausgangsvektoren des Modells definiert werden. Dies geschieht am einfachsten durch Definition einer eigenen Prozessdatenstruktur in UML für das Modell. Aus diesem Klassendiagramm kann, wie beschrieben, direkt ein XML-Schema generiert werden. Mit Hilfe dieses XML-Schemas können die Daten-Container für Ein- und Ausgang anschließend initialisiert werden.

Zum konkreten Einsatz fehlt jedoch noch eine wesentliche Komponente: die Anbindung der Prozessmodelldatenstruktur an die globale Prozessdatenstruktur. Diese Anbindung wird durch die Definition von Mapping-Regeln für die Prozessmodelldatenstruktur erreicht, wobei diese in XSLT geschieht und die im Abschnitt 2.1 definierten Regeln zur Transformation von XML-Schemata genutzt werden. Wenn die Prozessmodelle die Daten der Prozessdatenstruktur verwenden, ist kein Mapping erforderlich.

Da für die interne Kommunikation kein XML genutzt wird, verwaltet der Prozess für jedes Prozessmodell jeweils eine Instanz der Klasse *SchemaMapper*. Diese lässt sich

mit Hilfe der in XSLT formulierten Regeln initialisieren und dient zum Synchronisieren der Daten des Prozessmodells und des Prozesses. Der Datenaustausch geschieht dann über die Daten-Container des Frameworks. Der Prozess selber besitzt zudem Methoden, mit denen ein Mapping der Daten zweier Prozessmodelle ermöglicht wird, wobei die Prozessdatenstruktur immer als Referenz verwendet wird.

Der beschriebene Mapping-Mechanismus ermöglicht es Prozessen, zusätzlich auf Prozessmodelle zuzugreifen, die nicht mit dem Framework entwickelt wurden, die aber dem durch die Model-Klassen definierten Mechanismus entsprechen. Die Anbindung geschieht dabei durch Einbindung des Prozessmodells in eine abgeleitete Model-Klasse und mit Hilfe so genannter *Adapter*-Klassen, die die Datenversorgung auf die Daten-Container des Frameworks umleiten. Dadurch wird ein wesentliches Entwicklungsziel des Frameworks, nämlich die flexible und transparente Verknüpfung von Prozessmodellen,

erreicht.

An dieser Stelle sei angemerkt, dass ein ähnliches Problem unter dem Begriff Enterprise Application Integration (EAI) im Zusammenhang mit dem Informationsaustausch von internen Applikationen und der Kommunikation mit externen Partnern auf Geschäftsprozessebene bekannt ist. Eine mögliche Lösung bietet hier das BizTalk-Framework, das mit den beiden Methoden Mapping und Orchestration einen zur hier beschriebenen Vorgehen ähnlichen Lösungsweg vorschlägt [Vasters2000].

3.14.2 Definition des Prozess-Workflows

Die Klasse *Process* ist von der Klasse *AdaptiveModel* abgeleitet, ein Prozess verhält sich nach außen entsprechend wie ein gewöhnliches Modell. Entsprechend müssen auch hier wiederum die Methoden *forward*, *backward* und *adapt* überschrieben werden.

Eine wesentliche Erweiterung stellt die Methode *init* dar. Diese führt mit Hilfe der Offline-Trainer eine Initialisierung aller adaptiven Modelle durch. Dies bedeutet üblicherweise ein initiales Training der integrierten neuronalen Netze.

3.15 Fertigstraßenprozesse

Das Paket *Fertigstraßenprozesse* enthält alle Prozesse, die in der Prozessautomatisierung der Fertigstraße existieren. Prozesse organisieren den Ablauf und das Zusammenwirken von Modellen, dieses ist beispielhaft in Abbildung 3.12 dargestellt.

Die wichtigsten Fertigstraßenprozesse sind die Vorausberechnung und die Adaption. Die Vorausberechnung führt eine vollständige Stichplanberechnung durch und bestimmt somit die Sollwerte aller Stellglieder; die Adaption rechnet nach erfolgreicher Walzung den Stichplan nach und adaptiert mit Hilfe des Fehlers alle adaptiven Modelle.

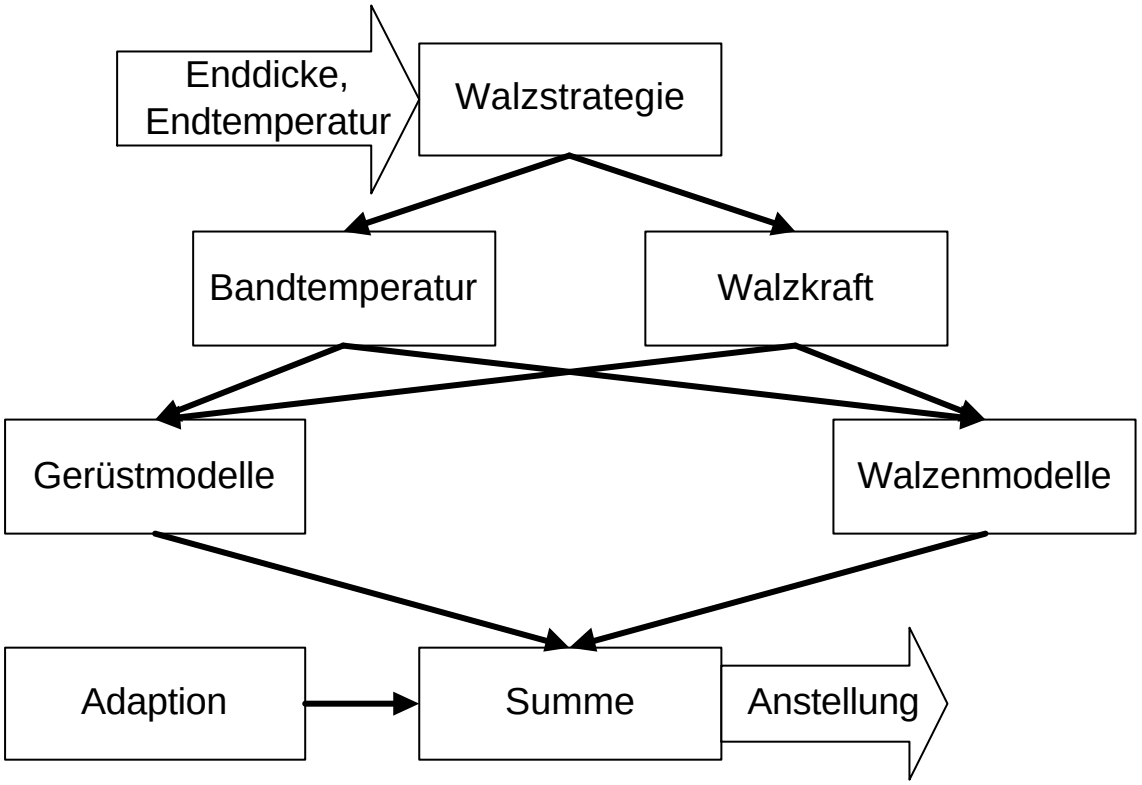


Abbildung 3.12: Workflow der Vorausberechnung

4 Datenerfassung

4.1 Motivation

Die Qualität der Flachwalzprodukte hängt in hohem Maße von Faktoren ab, die in gegenseitiger Abhängigkeit stehen. Wie in Kapitel 2.1 beschrieben, beeinflussen sich einerseits die einzelnen Größen des Umformprozesses, wie Walzkräfte, Drehmomente usw., und andererseits die Qualitätsergebnisse, wie Dicke, Breite, Temperatur und Profil, gegenseitig.

Entscheidend für die Qualität der gewalzten Produkte ist die Fähigkeit, den Walzspalt in den einzelnen Gerüsten genau zu beschreiben. Die Größe des Walzspaltes ist neben der Anstellung im Wesentlichen abhängig von der durch die Walzkraft verursachten Dehnung des Walzgerüstes und der Durchbiegung der Walzen. Die Genauigkeit der Walzkraftkalkulation ist wiederum abhängig von der Anstellposition und den Materialeigenschaften Dicke, Breite, Temperatur und dem Werkstoff.

Ziel dieser Arbeit ist es, unter Beibehaltung der vorhandenen physikalisch-mathematischen Walzmodelle, mit Hilfe von Datenanalysen und Optimierung der Adaptionsmechanismen unter Verwendung der im vorigen Kapitel vorgestellten Methodik, Fehler der einzelnen Modelle zu erkennen und ggf. zu verkleinern. In den folgenden Abschnitten werden die Ergebnisse der praktischen Datenakquisition und Datenaufbereitung dargestellt, die Ergebnisse der Optimierung folgen im nächsten Kapitel. Vorab wird eine eigens für diese aufwendigen Experimente entwickelte Methodik vorgestellt.

4.2 Vorbemerkungen zur Datenaufbereitung

In technischen Prozessen werden, speziell im industriellen Umfeld, große Mengen an Daten gesammelt, die die Regelungen und Steuerung der Prozesse beschreiben. Im speziellen Fall dieser Arbeit werden die Größen der Prozesssteuerung einer Fertigstraße eines Warmbreitbandwalzwerkes untersucht.

4.2.1 Begriffsdefinitionen

Um die Prozessautomatisierung sinnvoll und möglichst optimal zu versorgen, ist ein Verständnis der Daten notwendig. Der erste Schritt, die Erfassung der Daten, wird nachfolgend auch *Datenakquisition* genannt. Das weitere Vorgehen wird als *Datenaufbereitung* bezeichnet, die sich wiederum in zwei Abschnitte unterteilen lässt:

1. *Datenanalyse*, also die Analyse der Daten zur Erlangung eines Verständnisses über die Verteilung der Datenpunkte, z.B. mit statistischen oder graphischen Methoden.

2. *Datenbearbeitung* zur weiteren Untersuchung, z.B. das Entfernen und Korrigieren von fehlerhaften Daten oder Ausreißern.

Ausgehend von der kleinsten Einheit bis zur Gesamtmenge der Daten werden die nachfolgenden Begriffe verwendet:

Als *Datenwert* wird ein einzelner Zahlenwert bezeichnet; entsprechend seiner Funktion wird auch von einem Messwert, Stellwert oder von einem Prozessparameter gesprochen. Ein Beispiel für einen Datenwert ist der Kohlenstoffgehalt einer Stahllegierung für ein bestimmtes Band.

Ein *Datenmuster* besteht aus n zusammengehörenden Datenwerten, entsprechend werden die Daten als n-dimensional bezeichnet. Synonym werden für Datenwert auch die Begriffe *Muster*, *Daten-Pattern*, *Pattern* und *Datenpunkt* verwendet. Ein Beispiel für ein Datenmuster sind alle Legierungsbestandteile einer Stahllegierung.

Ein *Datensatz* ist der Spezialfall eines Datenmusters. Er beschreibt alle Datenpunkte, die zum selben Erfassungszeitraum gehören. Dies können z.B. alle Messungen auf einer Anlage zu einem definierten Zeitpunkt sein.

Als *Datenmatrix*, Mustermenge oder Patternset wird die Darstellung verschiedener Datenmuster in Matrixform bezeichnet.

Zu den wichtigsten Verfahren der Datenaufbereitung zählen Methoden der deskriptiven Statistik und verschiedene Clusterverfahren. Ein weiteres wichtiges Verfahren ist das Auffinden von Ausreißern. Nachfolgend werden die hier verwendeten Methoden kurz erläutert.

4.2.2 Clusterverfahren

Im Allgemeinen sind Messwerte nicht gleichmäßig verteilt, sondern weisen gewisse Strukturen auf. Wenn die einzelnen Datenmuster sich um bestimmte Zentren häufen, die ihrerseits relativ klar voneinander getrennt sind, bezeichnet man diese Zentren als *Cluster* oder *Häufungspunkte*.

Obwohl sich die Lage der Cluster bei niedrigen Dimensionen grafisch sehr einfach veranschaulichen lässt, ist es schwierig, dies mit Algorithmen zu erfassen, und es lassen sich häufig Beispiele angeben, die zu unplausiblen Ergebnissen führen. Entsprechend existieren verschiedene Verfahren zur Bestimmung der Zentren, die abhängig vom Anwendungsfall ausgewählt werden. Zusätzlich wird das Ergebnis

häufig noch von zufällig gewählten Anfangsbedingungen beeinflusst. Die in dieser Arbeit verwendeten Verfahren wurden bereits in Kapitel 2.3.5 beschrieben.

4.2.3 Bestimmung von Ausreißern

Für nahezu jeden Datenwert lassen sich Datenpunkte angeben, die weit entfernt von Bereichen mit einer hohen Konzentration an Datenpunkten anderer Datenmuster liegen. Diese Datenpunkte werden anschaulich als Ausreißer bezeichnet. Nachfolgend werden Ausreißer mathematisch definiert.

Ein n-dimensionales Datenmuster $x = (x_1, x_2, ..., x_n)$ heißt *Ausreißer*, wenn gilt:

$$\sqrt{\sum_{i=1}^{n}\left(\frac{x_i - m_i}{s_i}\right)^2} \geq k \qquad (4.1)$$

wobei μ_i der Erwartungswert und σ_i die Standardabweichung der i-ten Komponente aller Daten ist. Der Parameter k ist frei wählbar (i.a. $2 = k = 4$). Im normierten Fall ist x ein Ausreißer, falls gilt:

$$|x| > k \qquad (4.2)$$

Geometrisch liegen Ausreißer im normierten Fall außerhalb einer n-dimensionalen Kugel mit Radius k; im nicht normierten Fall handelt es sich um einen Ellipsoiden.

Ein Datenmuster x heißt *Ausreißer bezüglich eines Clusters* C mit Zentrum z_C, wenn gilt:

$$\sqrt{\sum_{i=1}^{n}\left(\frac{x_i - z_{C,i}}{s_{C,i}}\right)^2} \geq k \qquad (4.3)$$

wobei $\sigma_{C,i}$ die Streuung des Clusters in der Komponente i ist.

Ein Datenmuster x heißt *Ausreißer bezüglich eines Datensatzes mit Clusterstruktur*, wenn (4.3) für alle Cluster des Datensatzes gilt.

4.3 Verfahren der Datenaufbereitung

An diesem Punkt setzen eigene Überlegungen ein; entsprechend wird nachfolgend die gewählte Methodik zur Datenanalyse und das verwendete Verfahren zur Datenbearbeitung vorgestellt.

4.3.1 Datenanalyse

Aufgabe der Datenanalyse ist, sich ein Bild über den Datenbestand zu machen. Wichtige Verfahren der Datenanalyse sind die deskriptive Statistik und grafische Methoden, die im Folgenden kurz erläutert werden.

Die einfachsten statistischen Größen sind das *Minimum*, das *Maximum*, verschiedene Mittelwerte und Streuungen eines Datensatzes, wobei Minimum und Maximum einer Variablen Aussagen über den Gültigkeitsbereich erlauben. Dies lässt sich durch Mittelwerte und Streuungen unterstützen.

Neben dem *arithmetischen Mittelwert* können das *geometrische Mittel* und das *harmonische Mittel* bestimmt werden. Bei Prozessdaten zeigt sich jedoch, dass Ausreißer Mittelwerte relativ stark beeinflussen. Im Gegensatz dazu ist der *Median* ein robuster Schätzwert. Der Median ist der Datenwert, der genau in der Mitte einer sortierten Menge liegt. Ähnlich robust ist auch der *Trimmed Mean*, der die oberen und unteren x% einer Datenmenge ignoriert.

Streumaße reagieren gegenüber Ausreißern noch empfindlicher als Mittelwerte, wobei die *Spannweite*, also der Abstand von Minimum und Maximum, besonders empfindlich ist. Relativ anfällig sind auch die *Standardabweichung* und das Quadrat der Standardabweichung – die *Varianz*. Weniger empfindlich als die Standardabweichung sind die *mittlere absolute Abweichung*, da hier nicht das Quadrat des Abstandes eingeht, und der *Quartilsabstand*, der den Abstand zwischen dem oberen und dem unteren Viertel der Daten kennzeichnet.

Ziel der Datenanalyse ist vor allem, die wichtigsten Einflussgrößen eines Problems herauszufinden. Meistens sind einzelne Eingangsgrößen nicht mit der Zielgröße korreliert oder stellen abhängige Größen anderer Eingangsgrößen dar, wodurch sich die Dimension eines Problems und damit die entsprechende Komplexität eines Problems deutlich reduzieren lässt. Grundsätzlich werden bei der Untersuchung der Zusammenhänge zwischen Daten Korrelationen zwischen Ein- und Ausgängen - auch *Input-Target-Korrelationen* genannt - und Korrelationen der Eingänge (*Input-Input-Korrelationen*) unterschieden.

Eine weitere Möglichkeit zur Reduzierung der Dimension des Eingangsraums ist die Durchführung einer mutiplen linearen Regression. Mit dieser Methode wird eine n-dimensionale Regressionsgerade durch die Daten bestimmt. Wenn die lineare Regression keine Ergebnisse liefert, muss zu allgemeineren Ansätzen wie polynomialen Fits oder einer Datenmodellierung mit neuronalen Netzen übergegangen werden, wie sie im nächsten Kapitel beschrieben wird.

Mit Hilfe einer Hauptachsentransformation, die im Zusammenhang mit neuronalen Netzen meist als PCA (Principal Component Analysis) oder SVD (Singular Value

Decomposition) bezeichnet wird, kann die Dimension des Eingangsraums ebenfalls reduziert werden. Der Nachteil der PCA liegt darin, dass die gedrehten Komponenten keine anschauliche Interpretation mehr zulassen. Da jedoch Methoden gesucht werden, die gerade eine hohe Transparenz gestatten, wird auf den Einsatz der PCA im Folgenden verzichtet.

Neben den Methoden der deskriptiven Statistik helfen vor allem graphische Darstellungen der Daten in Form von Scatter Plots und Histogrammen, einen guten Überblick über die Verteilung und die Korrelationen der Daten zu erhalten. Ein Scatter Plot stellt die Daten dabei als Streudiagramm in einem X/Y-Koordinatensystem dar. Man erhält mit ihm einen guten Eindruck der Zusammenhänge seiner Variablen. Die besonders anschauliche Kombination von Histogrammen und Scatter Plots verschiedener Eingangsgrößen in Matrixform wird nachfolgend als *Matrixplot* bezeichnet.

4.3.2 Datenbearbeitung ohne Clusteranalyse

Die Datenbearbeitung aller Daten in dieser Arbeit verläuft nach folgendem Schema:

1. Skalieren: Häufig liegen Produktionsdaten in verschiedenen Einheiten und Einheitengrößen vor, daher müssen alle Daten zunächst in SI-Einheiten umgewandelt werden.

2. Normieren: Statistische Methoden, insbesondere neuronale Netze, liefern gerade bei normierten Eingängen optimale Ergebnisse; aus diesem Grund werden alle Eingänge normiert.

3. Sortieren: Prozessdaten werden in Prozessdatenbanken gespeichert. Die Datenakquistion führt diese i.a. unsortiert zusammen, daher müssen für einige statistische Untersuchungen (z.B. Zeitreihen) die Daten sortiert werden.

4. Überprüfung, ob auf eine Clusteranalyse verzichtet werden kann: Die einfachste Möglichkeit bieten hierzu grafische Methoden (z.B. Matrixplots), die sofort veranschaulichen, ob eine Datenmatrix Cluster besitzt oder nicht.

5. Eliminieren von nicht signifikanten Ausreißern: Hierzu kommen die in Abschnitt 4.2.3 beschriebenen Verfahren zum Einsatz.

6. Eliminieren nicht plausibler Daten: Gerade bei Prozessdaten treten häufig Daten auf, die aus verschiedenen Gründen unplausibel sind. Dies können z.B. negative Werte in Messgrößen sein, die nur positive Werte annehmen können. Im Allgemeinen lassen sich für alle Prozessgrößen zudem Gültigkeitsgrenzen angeben, in denen plausible Daten liegen; diese Grenzen können jedoch zusätzlich von Werten anderer Prozessgrößen abhängen.

7. Auffinden von abhängigen Komponenten: Mit Hilfe der in Abschnitt 4.3.1 beschriebenen Methoden - wie multipler linearer Regression und SVD - lassen sich lineare Zusammenhänge zwischen verschiedenen Prozessgrößen auffinden. Dabei werden Abhängigkeiten zwischen Eingängen untereinander (Input-Input-Korrelation) und Eingängen und dem Ausgang (Input-Target-Korrelation) unterschieden.

8. Eliminieren von abhängigen Komponenten: Falls sich im vorigen Schritt lineare oder einfache funktionale Abhängigkeiten ergeben haben, kann eine Eingangskomponente eliminiert werden und bei der Berechnung des Ausgangs durch eine entsprechende Abbildung ersetzt werden. Falls sich kein Zusammenhang zwischen Eingang und Ausgang zeigt, kann eine solche Prozessgröße ohne Informationsverlust eliminiert werden.

Der größte Aufwand entsteht durch das Auffinden abhängiger Komponenten, das Eliminieren der Ausreißer und das Eliminieren nicht plausibler Daten. Nicht alle als Ausreißer erkannten Datensätze dürfen nämlich auch eliminiert werden, wenn diese ein Cluster darstellen, das von der Datenmenge nur durch einen einzelnen Datensatz repräsentiert wird. Auf der anderen Seite müssen Datensätze eliminiert werden, die mathematisch oftmals keine Ausreißer darstellen, die aufgrund ihrer Werte jedoch unplausibel sind. Noch schwieriger ist es, Datensätze mit Inkonsistenzen zwischen einzelnen Komponenten aufzufinden. Dies tritt z.B. auf, wenn Messwert und Sollwert sehr stark abweichen.

4.3.3 Datenbearbeitung mit Clusteranalyse

Die Clusteranalyse vereinfacht die Unterscheidung von plausiblen und unplausiblen Ausreißern, daher sollte eine Clusteranalyse ein Bestandteil der Datenbearbeitung sein, wenn mit geclusterten Daten zu rechnen ist. Beim Walzen tritt dies vor allem auf, wenn der Vektor der Legierungsbestandteile Verwendung findet.

Die Clusteranalyse umfasst zwei Schritte, die als Alternative zu Punkt 4 der zuvor beschriebenen Vorgehensweise angesehen werden kann:

4.1. Bestimmen der Cluster: Die Bestimmung der Cluster geschieht mit den in Kapitel 2.3.5 beschriebenen Methoden, zusätzlich ist auch die optimale Clusterzahl zu bestimmen. Dies geschieht am einfachsten mit grafischen Methoden, die die Verteilung der Cluster mit der realen Datenverteilung vergleichen oder mit einfachen Heuristiken, die den RMS-Abstand der Datenpunkte zum nächsten Cluster in Relation zur Clusteranzahl setzen.

4.2. Normierung der Cluster: Da die meisten Clusterverfahren auf unnormierten Daten basieren, müssen die Cluster abschließend normiert werden.

4.4 Datenakquisition

Während der Entstehung dieser Arbeit wurde die gesamte Prozessautomatisierung der Fertigstraße der betrachteten Warmbreitbandstraße ausgetauscht, so dass die seltene Möglichkeit vorhanden war, Vergleiche zwischen unterschiedlichen Prozessautomatisierungsstrategien, repräsentiert durch das alte und das neue System, zu ziehen. Nachfolgend wird von alter und neuer Prozessautomatisierung gesprochen, womit jeweils das vor bzw. nach der Modernisierung implementierte System auf der betrachteten Anlage angesprochen ist.

Die Daten der alten Prozessautomatisierung wurden durch einen Datenbankabzug kurz vor den ersten Inbetriebnahmetests mit der neuen Prozessautomatisierung erfasst. Die aktuellsten Daten des Altsystems stammen daher aus dem Oktober 1997. Der Übergang vom Altsystem zum Neusystem fand 1998 und 1999 statt. Alle Daten aus dieser Zeit, die in den Betriebsdatenbanken gespeichert wurden, werden nicht für statistische Auswertungen verwendet, da aufgrund der Unsicherheit der Inbetriebnahme von Prozessmodellen keine eindeutige Zuordnung zur alten bzw. neuen Prozessautomatisierung möglich ist.

Die Daten des Neusystems wurden mit dem im letzten Kapitel beschriebenen Framework erfasst und unterliegen somit nicht den Beschränkungen der Betriebsdatenbanken. Dennoch wurden nur Walzpläne erfasst, die durchgängig mit dem neuen Prozessrechner gewalzt wurden. Die Daten wurden von August 1998 bis Januar 2000 erfasst.

4.5 Daten des Altsystems

4.5.1 Erfassung der Daten

Die Daten des Altsystems wurden mit Hilfe eines so genannten technischen Informationssystems gesammelt, das für jedes Band etwa 1000 Prozessgrößen protokolliert. Diese Daten teilt das technische Informationssystem in drei Gruppen auf. Die Daten der ersten Gruppe werden 450 Tage vorgehalten, dies entspricht etwa 170.000 Bändern. Die Daten der zweiten Gruppe werden ein halbes Jahr gespeichert, diese Datenbanken enthalten etwa 75.000 Datensätze. Die Daten der letzten Gruppe werden nur 40 Tage gespeichert; somit liegen von allen Größen, die in diesen Datenbanken gespeichert werden, nur 18.000 Bänder vor.

Die 1000 Prozessgrößen sind entsprechend ihrer Gruppenzugehörigkeit in sehr wichtige, wichtige und weniger wichtige Größen unterteilt. Für die wichtigsten statistischen Auswertungen oder auch für das Training neuronaler Netze stehen damit immer 170.000 Datensätze zur Verfügung, für Auswertungen mit beliebigen Daten sind immer noch fast 20.000 Datensätze vorhanden. Es hat sich gezeigt, dass

auch in dieser kleineren Menge nahezu das komplette Spektrum an gewalzten Güten und Abmessungen der Bänder enthalten ist.

Name Datenbank	Zeitstempel des ersten Datensatzes	Zeitstempel des letzten Datensatzes	Anzahl Datensätze
FS_SD_1	07.07.1996 01:00:27	29.09.1997 06:45:19	171.816
FS_SD_2	07.07.1996 01:00:27	29.09.1997 10:29:35	171.847
FS_SD_3	03.04.1997 08:20:45	29.09.1997 13:41:55	75.556
FS_SD_4	03.04.1997 08:20:45	29.09.1997 15:20:20	75.594
FS_SD_5	21.08.1997 00:00:24	29.09.1997 16:06:39	18.101
FS_SD_6	21.08.1997 00:00:24	29.09.1997 16:32:54	18.103
PLR	14.07.1996 12:52:11	05.10.1997 19:54:49	184.106

Tabelle 4.1: Datensätze Altsystem

Zusätzlich zu den Datenbanken des technischen Informationssystems, das Prozessgrößen, Messwerte und Stellgrößen der Basis- und der Prozessautomatisierung (Level 1 und 2) aufnimmt, existieren weitere Datenbanken für die Daten der Prozessleitebene (Level 3). Die wichtigste Datenbank des Prozessleitrechners speichert die Daten, wie die Prozessgrößen der ersten Gruppe, etwa 450 Tage lang. Entsprechend sind in dieser Datenbank etwa 180.000 Datensätze enthalten. Tabelle 4.1 gibt einen Überblick über die verschiedenen Datensätze.

4.5.2 Rohdatenanalyse

Mit Hilfe der Rohdatenanalyse des Altsystems sollen wichtige von unwichtigen Eingangsgrößen getrennt werden. Außerdem ist es notwendig, sich vor weitergehenden Analysen einen Überblick über einzelne Daten zu verschaffen.

Die Rohdatenanalyse beinhaltet dabei immer die Bestimmung von Mittelwert, Standardabweichung, Minimum und Maximum aller Größen. Zusätzlich ist es vielfach hilfreich, die Statistik um den Median und 25%- und 75%-Quantile zu erweitern. Es zeigt sich nämlich, dass nicht nur bei den Messwerten, sondern bei allen Größen Ausreißer vorhanden sind. Wie weiter oben in diesem Kapitel beschrieben, sind gerade das Minimum und das Maximum sehr anfällig gegenüber Ausreißern.

Einen noch besseren Überblick über die Daten erhält man mit grafischen Methoden. Für die Rohdatenanalyse werden dazu Histogramme in Kombination mit Scatterplots der Eingangsgrößen verwendet. Diese als Matrixplot bezeichneten Darstellungen der Daten besitzen den großen Vorteil, dass sie neben einer Aussage über die Verteilung einer Einzelgröße auch Aussagen über die Korrelation zwischen verschiedenen Eingangsgrößen liefern.

In Abbildung 4.1 ist exemplarisch ein Matrixplot für die wichtigsten Prozessmodellgrößen im letzten Walzgerüst aufgeführt.

Bereits in dieser sehr groben Darstellung von 10 Eingangsgrößen lassen sich eine Reihe von Aussagen über die Prozessgrößen treffen. Das Walzentemperaturmodell (DFB1_7) nimmt in über 90% der Fälle einen identischen Wert an. Dies ist physikalisch nicht plausibel und zeigt direkt Optimierungspotentiale. Die Anstellungsadaption (DFBI_7) zeigt eine gewisse Korrelation zur Walzkraft und auch zum Lagermodell (DFB3_7). Da die Anstellungsadaption andererseits direkt aus dem Dickenfehler des vorhergehenden Bandes bestimmt wird, spricht somit vieles für einen Fehler im Prozessmodell für das Lageraufschwimmen. Allerdings ist anzumerken, dass die Walzkraft zweier nacheinander gewalzter Bänder aufgrund der Walzpläne, die Kommissionen mit ähnlichen Bändern vorsehen, ebenfalls leicht korrelieren.

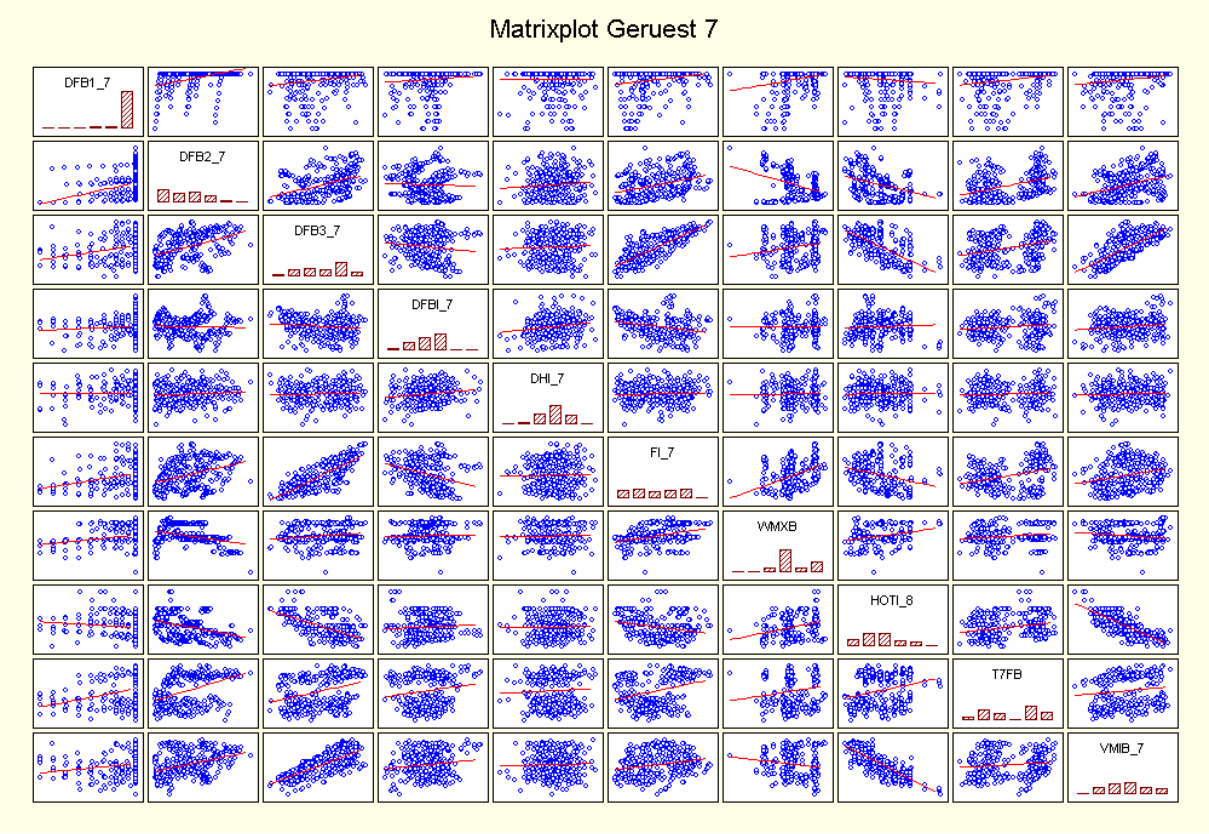


Abbildung 4.1: Matrixplot der Eingangsgrößen im letzten Walzgerüst

Für die weiteren Betrachtungen muss aus der Menge von 1000 Prozessgrößen eine möglichst kleine Teilmenge ausgewählt werden, die jedoch möglichst dieselbe Information enthält. Alle Betrachtungen, die das Altsystem betreffen, werden nur für das letzte Gerüst der Fertigstraße durchgeführt, da sich alle Ergebnisse sehr leicht auf alle vorherigen Gerüste übertragen lassen. Entsprechend lassen sich alle Pro-

zessdaten der ersten sechs Gerüste ohne großen Informationsverlust vernachlässigen.

Die Prozessdatenbank speichert für jedes Band mehrere Messwerte, die sich immer auf definierte Werte des Bandes beziehen. Für die in dieser Arbeit gemachten Untersuchungen werden Messwerte benötigt, die sowohl unabhängig vom Einschwingen der Stellglieder als auch unabhängig von Einflüssen der Basisautomatisierung sind. Aus diesem Grund werden nur gemittelte Messwerte aus einem definierten Segment des Bandes verwendet.

Diese und einige weitere Maßnahmen führen dazu, dass letztlich 25 Größen der Prozessdatenbank für die weitere Untersuchung herangezogen werden. Diese lassen sich folgendermaßen gliedern:

- Messwerte (Bandtemperatur hinter Gerüst 7, Walz- und Biegekraft in Gerüst 7, Bandgeschwindigkeit hinter Gerüst 7, Eintritts- und Austrittsdicke Gerüst 7)
- Chemische Zusammensetzung (C,Si,Mn,P,S,Al,Cr,Cu,Mo,Ti,Ni,V,Nb,N,B,Sn)
- Umformtechnische Kenngrößen (relative Abnahme in F7)
- Gerüstparameter (Arbeitswalzenradius des letzten Gerüstes)

Weiterhin werden die Modellergebnisse des Walzentemperaturmodells, des Verschleissmodells, des Lagermodells, der Ständerdehnung und der Anstellungsadaption für die Auswertung benötigt.

Als Zielgröße wird die Differenz aus Solldicke und gemessener Dicke hinter Gerüst 7 als Dickenfehler Δh definiert. Bei der Solldicke handelt es sich um die in Abschnitt 2.2.5.3 definierte Gaugemeterdicke.

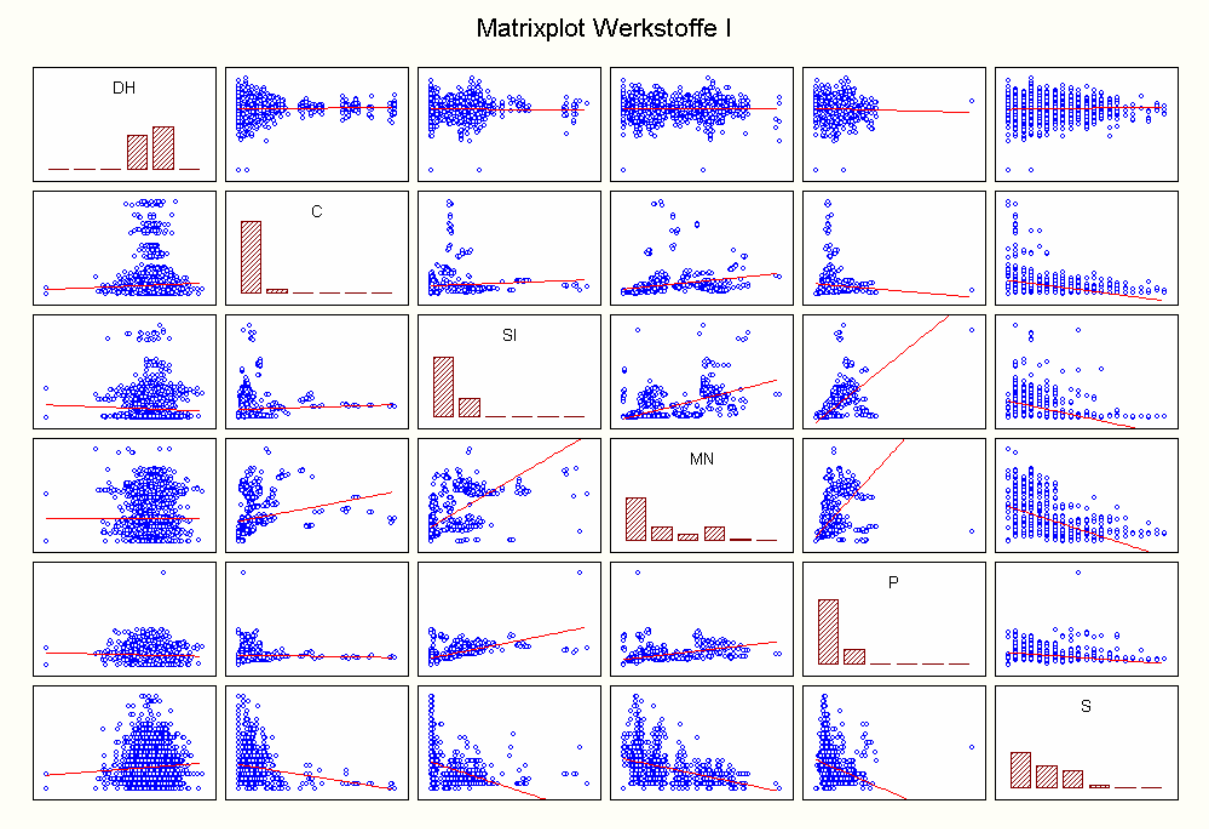
Unter *Dickentreffsicherheit* wird nachfolgend die Güte des Dickenfehlers verstanden. Dies kann einerseits der durch die Standardabweichung ausgedrückte Wert sein, der als *absolute Dickentreffsicherheit* Q bezeichnet wird. Die in der Produktion häufiger verwendete *relative Dickentreffsicherheit* q gibt immer die Relation zwischen den gewalzten Bändern, die sich innerhalb einer gewissen Toleranz des Dickenfehlers befinden, und der Gesamtmenge der Stichprobe an.

$$Q = s(Dh) \tag{4.4}$$

$$q = \frac{|A|}{|\Omega|} = \frac{|A|}{n}, mit\ A = \{\Delta h_i : \Delta h_i \in \Omega,\ \Delta h_i < e\} \tag{4.5}$$

4.5.3 Scatterplots der Zielgröße mit wichtigen Eingangsgrößen

Statistische Untersuchungen zeigen, dass die Dickentreffsicherheit sehr stark vom gewalzten Material abhängt.



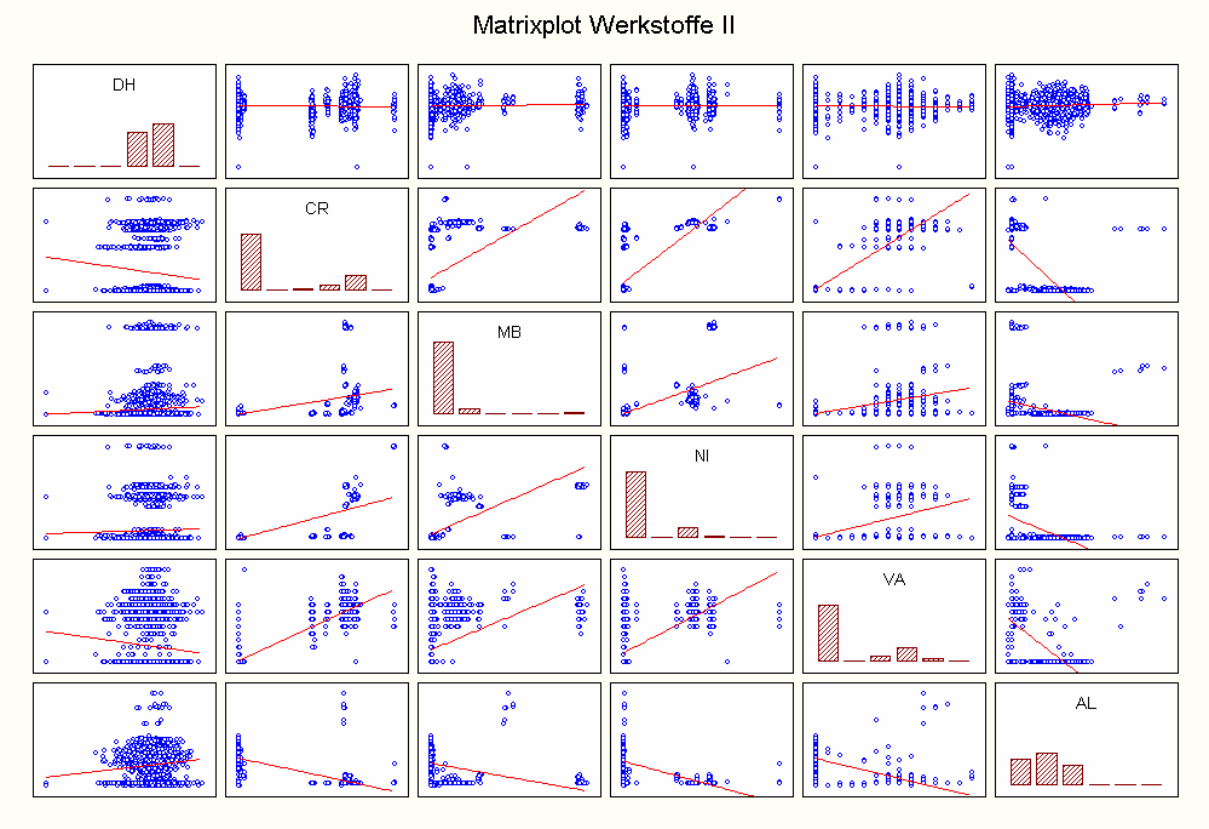


Abbildung 4.2: Abhängigkeit des Dickenfehlers von der chemischen Analyse

Während bei einer großen Anzahl von Normalstählen ohne hohen Legierungsanteil und einer Reihe von häufig gewalzten Edelstählen kaum Qualitätsprobleme auftre-

ten, treten diese gerade bei sehr selten gewalzten Güten auf, daher ist eine Untersuchung der Abhängigkeit zwischen Legierungsanteilen und Dickenfehler Erfolg versprechend.

Die Daten des Altsystems wurden entsprechend mit Hilfe von Matrixplots ausgewertet, das Ergebnis zeigen die beiden Matrixplots aus Abbildung 4.2. Wie zu erkennen ist, ist eine Aussage nur mit Hilfe der Matrixplots nicht möglich. Es lassen sich zwar keine direkten Abhängigkeiten ableiten, die Vielfalt der Cluster zeigt aber auch, dass eine Clusteranalyse, die Wahrscheinlichkeit einen Zusammenhang zu finden deutlich erhöhen würde. Eine solche Analyse wurde daher für die Daten des Neusystems durchgeführt und wird in Abschnitt 4.7 beschrieben.

4.5.4 Datenanalyse mit neuronalen Netzen

Eine Möglichkeit, die Güte der Prozessmodelle zu bestimmen, besteht darin, eine multiple lineare Regression mit dem Dickenfehler als Zielgröße durchzuführen und deren Ergebnis zu bewerten. Alternativ lässt sich diese mit Hilfe eines linearen neuronalen Netzes durchführen. Ein solches Netz wurde an der betrachteten Fertigstraße installiert, im Folgenden wird es als GRENN I (gaugemeter residual error neural network I) bezeichnet. Bei dem verwendeten Netz handelt sich um ein ADALINE, das in den Abschnitten 2.3.2 und 3.8.2 erläutert wurde. Da ein solches lineares Netz sehr unkritisch im Laufzeitverhalten und im Speicherplatzbedarf ist, ist es legitim, zunächst alle Größen als Eingang zu verwenden, die in den Vorbetrachtungen einen Einfluss auf den Dickenfehler zeigten. Das ADALINE wird entsprechend mit insgesamt 25 Eingangsgrößen versorgt; alle eingesetzten neuronalen Netze werden ausführlich in Kapitel 5 beschrieben.

Als Ausgabe liefert das Netz den zu erwartenden Gaugemeterfehler Δh_G^* hinter dem letzten Gerüst. Das Ergebnis wird nicht in den Prozeß zurückgekoppelt, sondern nur für Auswertezwecke protokolliert (Abbildung 4.3).

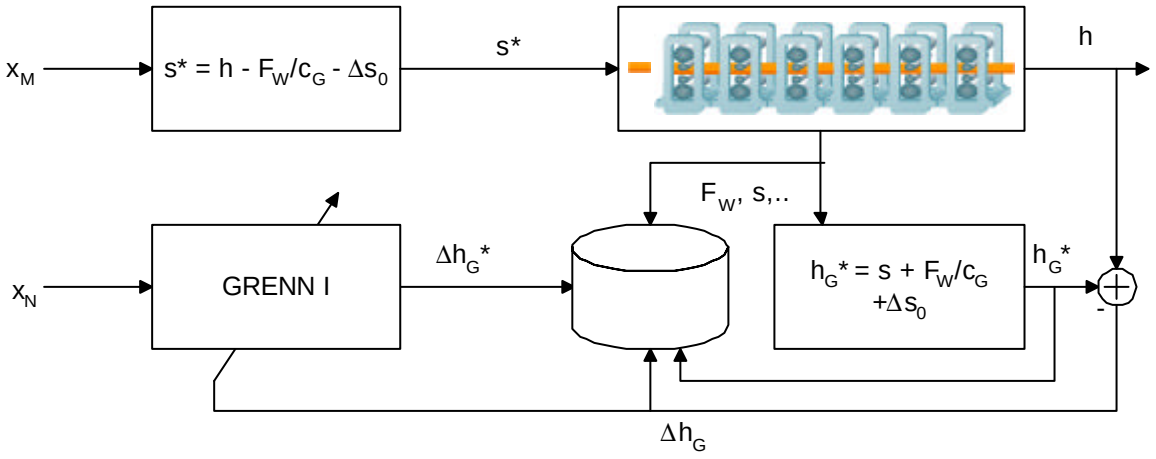


Abbildung 4.3: Schematische Darstellung des GRENN I

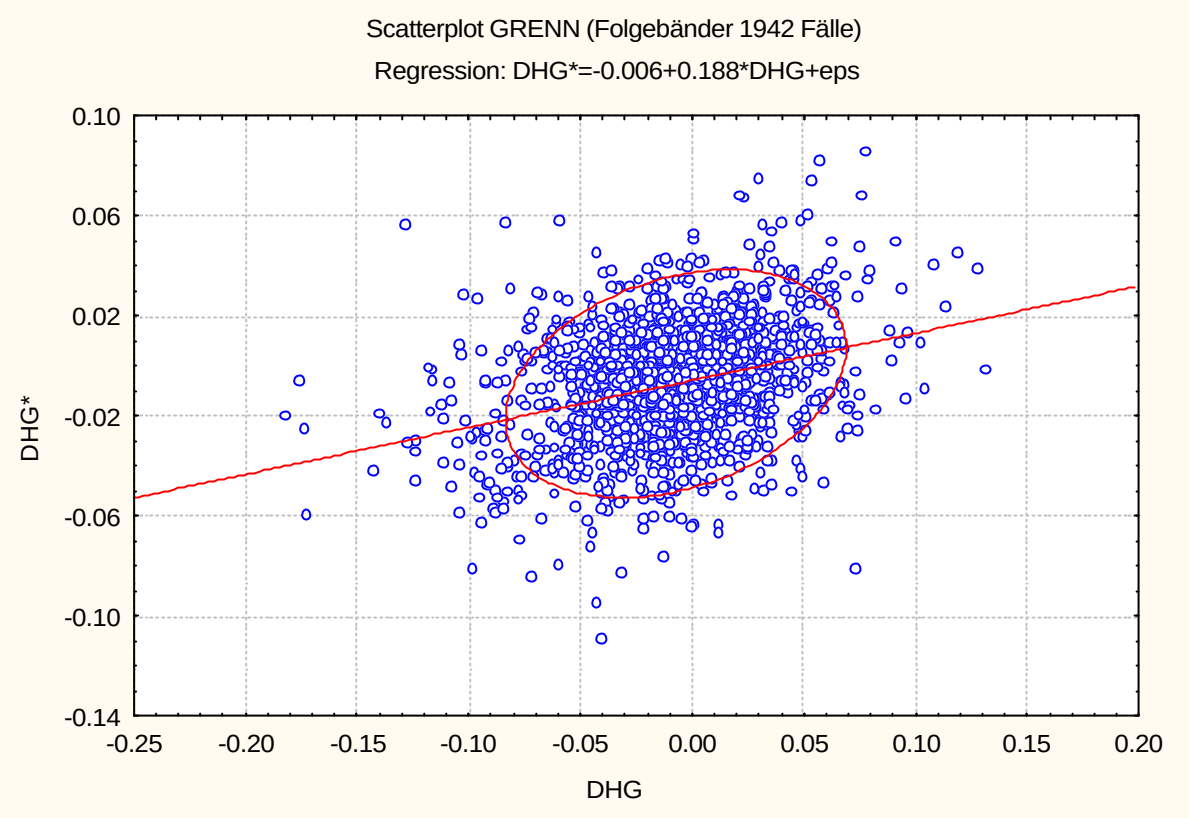


Abbildung 4.4: Vorhersage des Gaugemeterfehlers für Folgebänder

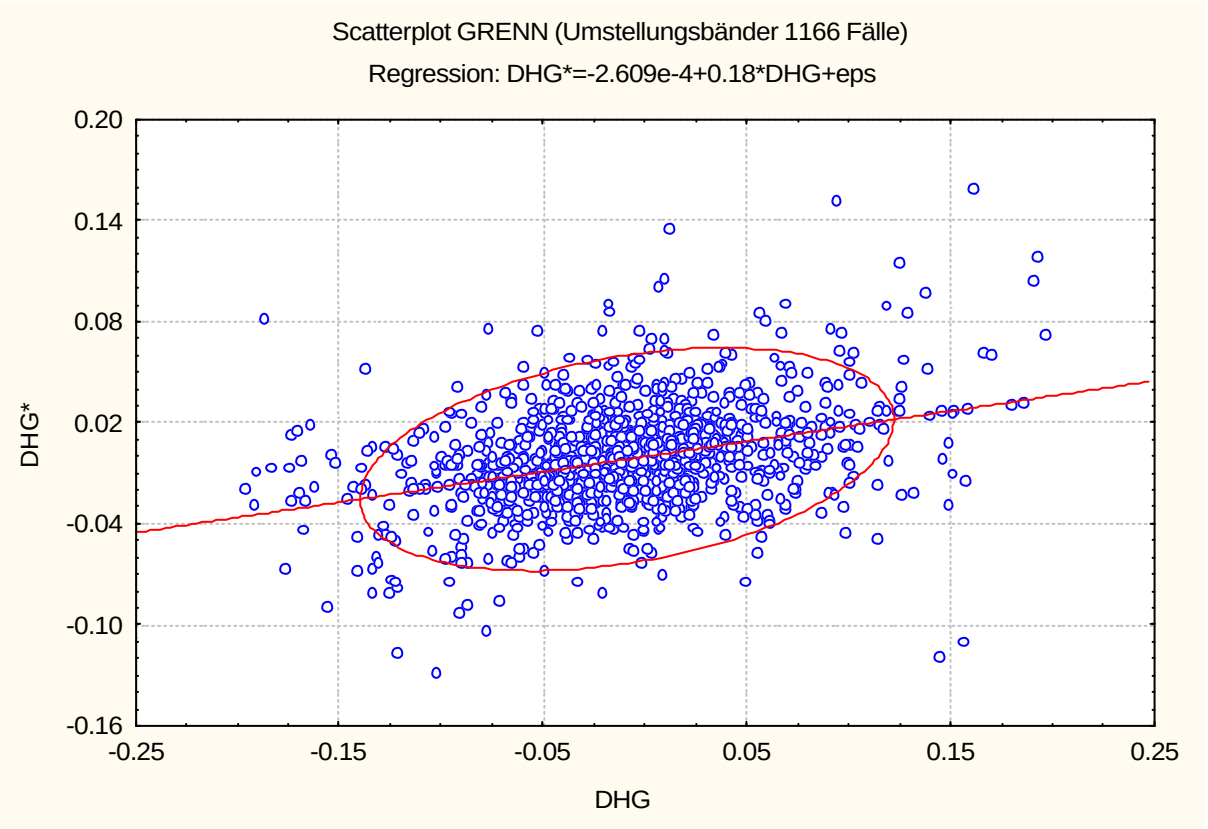


Abbildung 4.5: Vorhersage des Gaugemeterfehlers für Umstellungsbänder

Das GRENN I ist kaum in der Lage, den Gaugemeterfehler vorherzusagen, dennoch lassen sich die Netzausgaben bewerten. Trägt man den geschätzten über den tatsächlichen Fehler auf, so erhält man die in Abbildung 4.4 und Abbildung 4.5 angegebenen Scatterplots. Dabei werden so genannte *Folgebänder*, das sind Bänder, bei denen die chemische Zusammensetzung und die geometrischen Abmessungen im Vergleich zum Vorband nahezu konstant bleiben, und *Umstellungsbänder* unterschieden. Es ist deutlich zu erkennen, dass die Güte der Vorhersage in beiden Fällen etwa gleich ist. Sie lässt sich aus der Steigung der Regressionsgeraden bzw. aus der Steigung der Hauptachse der eingezeichneten Ellipse, die 90% der Daten enthält, ablesen. Dies ist insofern erstaunlich, als der Fehler bei Folgebändern mit 35 µm nur etwa halb so groß ist wie der Fehler bei Umstellungsbändern, der bei etwa 65 µm liegt.

Abgesehen von der Tatsache, dass das Netz nur einen Dickenfehler und keine Anstellungskorrektur prognostiziert, ist das Ergebnis nur so gut, dass bei einem Aufschalten des Netzergebnisses zur Anstellungskorrektur die Verbesserung im Bereich von maximal 1 - 2 % läge. Dies ist angesichts der Einfachheit des Netzes aber auch nicht anders zu erwarten.

Aufgrund der in der Prozessautomatisierung verwendeten Online-Adaption im laufenden Betrieb ist es jedoch erstaunlich, dass sich überhaupt ein Einfluss zeigt, da zu vermuten wäre, dass hier zumindest eine lineare Optimierung erreicht wird - für die Ausgabe des GRENN I würde dies eine Regressionsgerade mit der Steigung Null bedeuten. Tatsächlich ist diese jedoch in beiden Fällen deutlich von Null verschieden. Positiv ist zudem zu vermerken ist, dass wenige Fehler auftreten, bei denen das Netz sehr falsch liegt, die Vorzeichentreue ist besser als zu erwarten, die Ausdehnung der zweiten Hauptachse der eingezeichneten Ellipsen ist jedoch für einen betrieblichen Einsatz noch zu groß.

4.6 Daten des Neusystems

4.6.1 Erfassung der Daten

Die Erfassung der Daten des neuen Prozessrechners konnte nicht mit dem technischen Informationssystem durchgeführt werden, da dieses für die Anlage benötigt wurde, die zunächst weiter mit dem Altsystem betrieben wurde. Stattdessen wurde die Eigenschaft des im letzten Kapitel beschriebenen Frameworks genutzt, Daten für Auswertungen zu sammeln und in Datenbanken abzulegen.

Im Gegensatz zum Altsystem wurde beim Neusystem nicht ein großer Datensatz erzeugt, sondern es wurden für alle Experimente zugeschnittene Daten erfasst. Im Einzelnen handelt es sich um folgende Datensätze:

- August bis Dezember 1998: Daten für Restfehlernetze
- Januar bis April 1999: Daten für Clusteranalyse und Diagnosenetze
- September 1999 bis Januar 2000: Daten für zweite Clusteranalyse und Walzkraftnetze
- Oktober bis Dezember 2000: Referenzdaten des laufenden Betriebes

4.6.2 Datenvolumen

Ein gutes neuronales Netz zeichnet sich durch eine richtige Wahl der Eingangsgrößen aus. Nur Inputs, die sich signifikant auf den Ausgang auswirken, sollten verwendet werden. Eingangsgrößen, die stark mit anderen Eingängen korrelieren oder stark verrauscht sind, sollten dagegen vernachlässigt werden. Eine gute Aussagekraft über die Güte der Eingangsgrößen erhält man, wenn man jede Eingangsgröße gegen jeden anderen Eingang und zusätzlich auch noch gegen den Ausgang aufträgt.

Nach erfolgreicher Suche einer Menge von Eingangsgrößen müssen Ausreißer ausgemacht werden. Dies geschieht am zweckmäßigsten durch Festlegung von Grenzen, die einzelne Eingänge nicht über- bzw. unterschreiten dürfen.

Für die nachfolgend beschriebenen Untersuchungen wurden Daten verwendet, die in den ersten 8 Kalenderwochen 1999 gesammelt wurden. Es handelt sich um 22082 gültige Datensätze. Sie decken bis auf einige sehr seltene Sonderfälle das gesamte Produktspektrum der betrachteten Warmbreitbandstraße ab.

4.6.3 Rohdatenanalyse

Die Rohdatenanalyse der Daten des Neusystems dient der Unterstützung der Ergebnisse der Analyse des Altsystems. Die in Abschnitt 4.5.2 beschriebenen signifikanten Eingangsgrößen werden nun ein zweites Mal untersucht.

Die Untersuchungen mit Matrixplots haben noch keine eindeutigen Zusammenhänge zwischen einzelnen Eingangsgrößen untereinander und den Eingängen und der Zielgröße Dickenfehler hinter dem letzten Walzgerüst gezeigt, daher werden jetzt alle Zusammenhänge mit Hilfe von Scatterplots untersucht.

Wie aus Abbildung 4.6 zu entnehmen ist, existiert eine hohe Korrelation zwischen der Endwalzdicke und der Walzgeschwindigkeit im letzten Gerüst. Diese resultiert aus dem Maximum des Volumenstroms, der sich als Produkt aus Walzgeschwindigkeit, Walzgutbreite und Walzgutdicke ergibt. Aus physikalischen Gründen besteht zudem eine hohe Korrelation zwischen der Endwalzdicke und allen Walzkräften, daher wird der Einfluss der Endwalzdicke nachfolgend vernachlässigt werden und die Endwalzdicke nur noch zur Selektion verwendet.

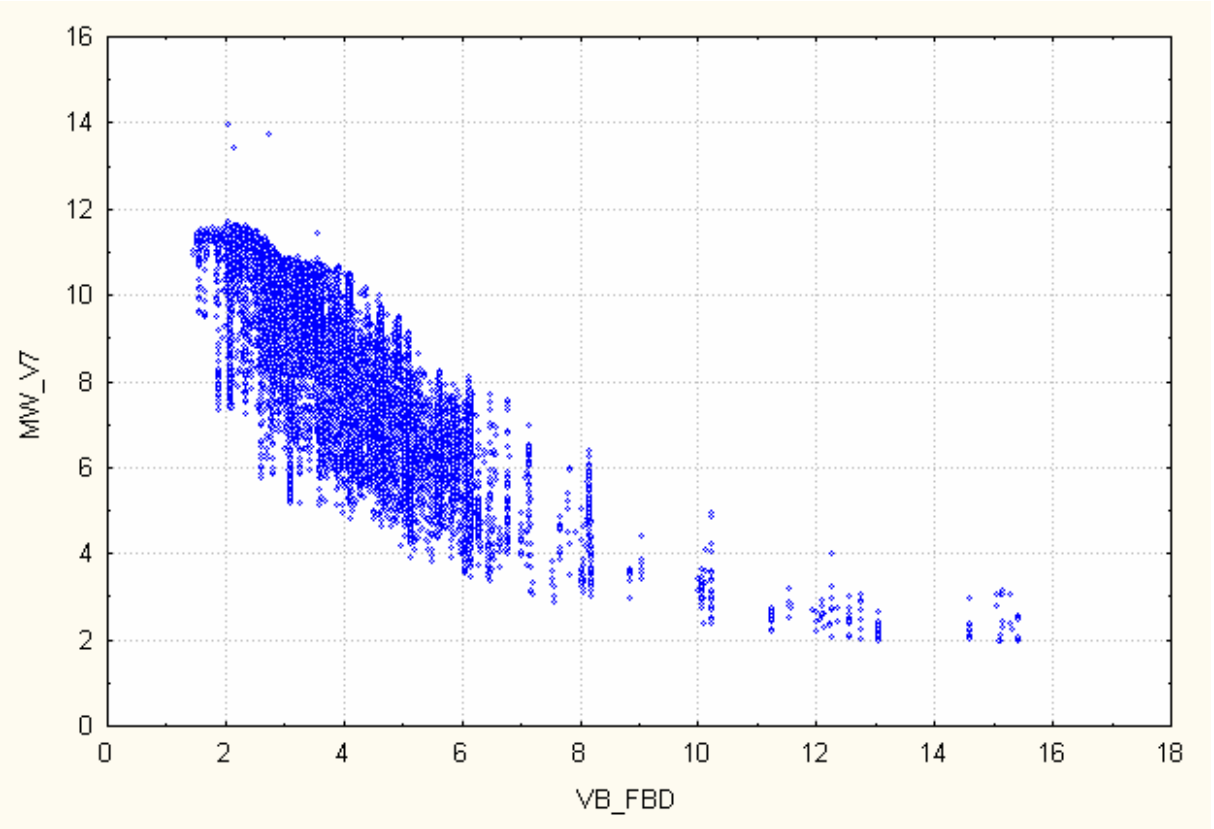


Abbildung 4.6: Korrelation zwischen Fertigbanddicke und Bandgeschwindigkeit

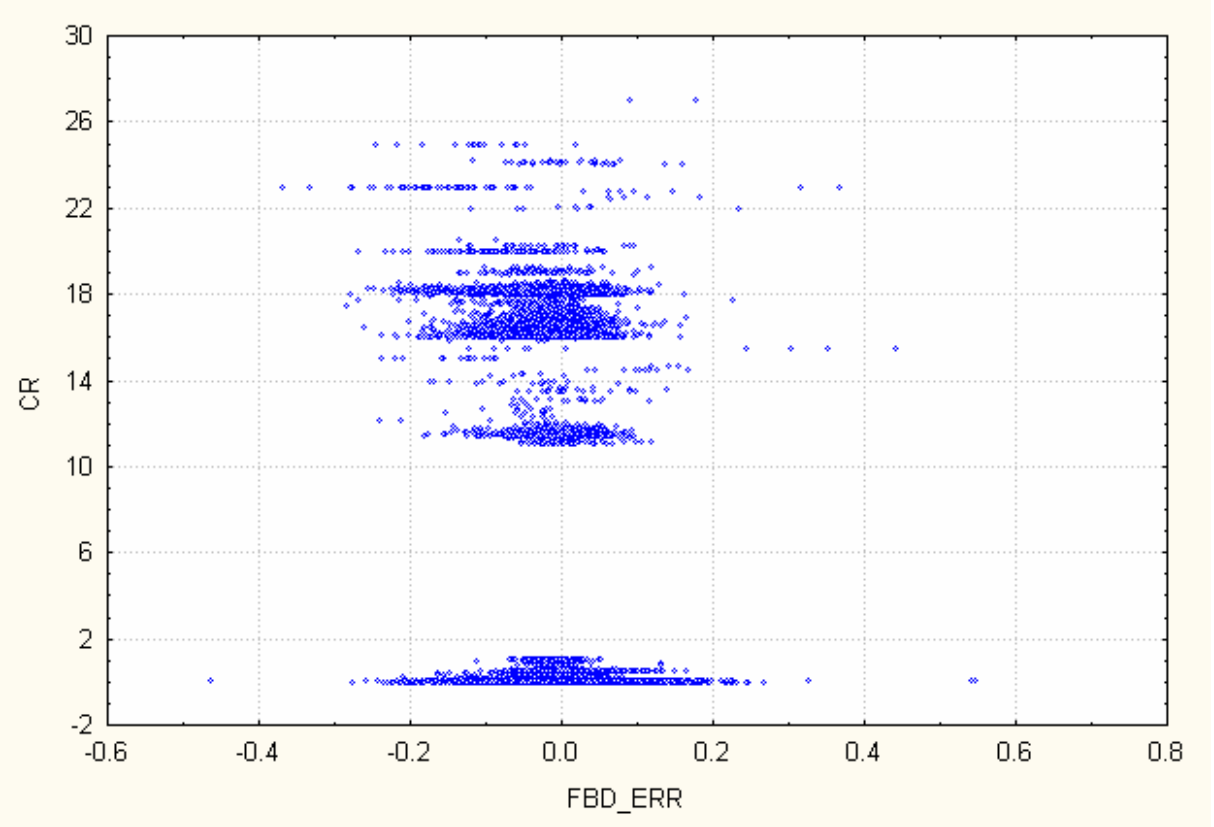


Abbildung 4.7: Korrelation zwischen Dickenfehler und Chromgehalt

Wie schon im vorigen Abschnitt erwähnt, zeichnet sich die betrachtete Warmbreitbandstraße durch ein sehr vielfältiges Produktspektrum aus. Dies führt dazu, dass

die Prozessmodelle nicht alle gewalzten Güten mit der gleichen Qualität bestimmen können. Abbildung 4.7 veranschaulicht am Beispiel eines Scatterplot des Dickenfehlers und des Legierungsanteils des Werkstoffes Chrom, dass sich selbst im zweidimensionalen Fall einige Cluster bilden. Interessanterweise liegen die Clusterzentren nicht, wie zu vermuten wäre, alle bei einem Dickenfehler von Null, sondern teilweise bei positiven und vor allem im Edelstahlbereich bei deutlich negativen Werten. Ähnliche Ergebnisse ergeben sich bei Betrachtung weiterer Legierungen wie Nickel und Mangan; andere Elemente wie z.B. Kupfer oder Schwefel haben demgegenüber fast keinen Einfluss auf den Dickenfehler.

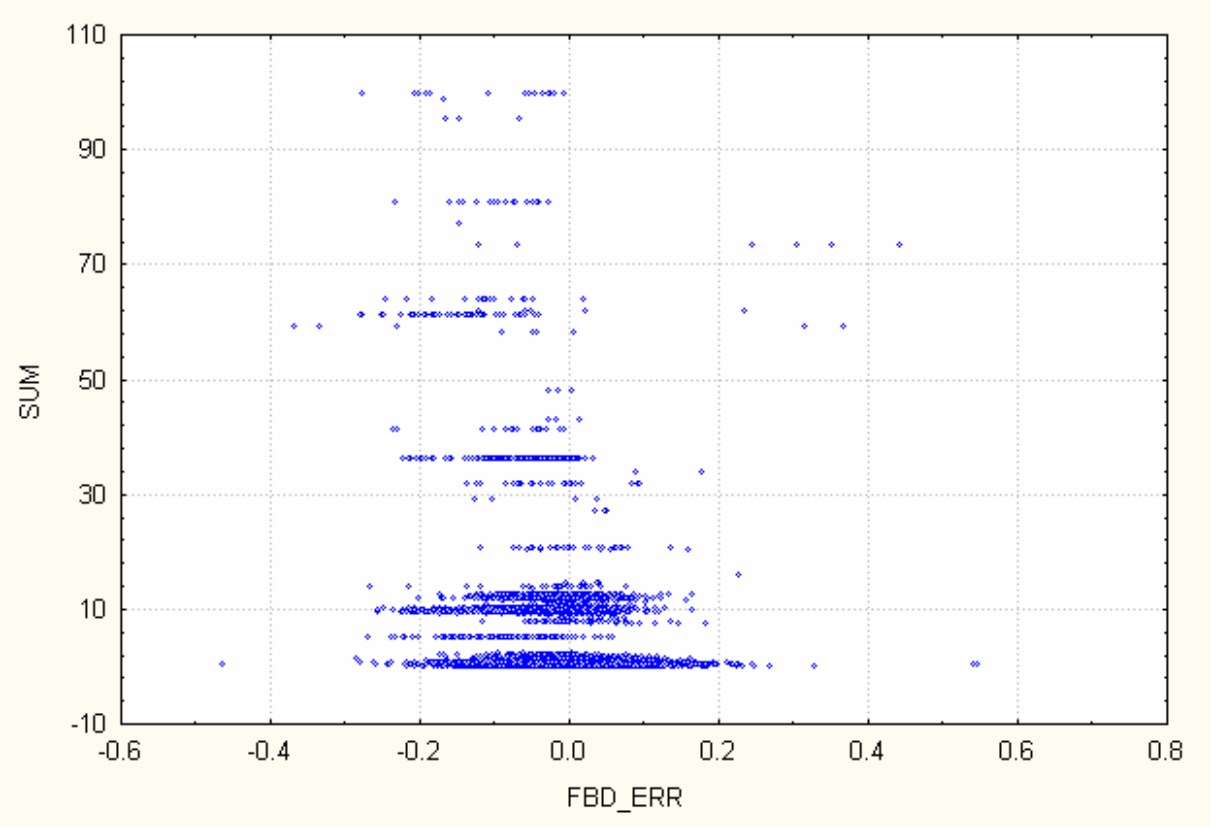


Abbildung 4.8: Korrelation zwischen Dickenfehler und Summe der Legierungselemente

Noch anschaulicher wird die Problematik, die Abbildung 4.8 mit der Darstellung der Summe aller Analysewerte über dem Dickenfehler zeigt. Es sind eindeutig verschiedene Cluster auszumachen, die noch deutlicher als bei der Betrachtung der Chromgehalte im Bereich positiver und negativer Dickenfehler liegen. Bei Betrachtung der absoluten Werte der Legierungsanteile ist festzustellen, dass teilweise Werkstoffe gewalzt werden, deren Summenteil aller Legierungen zwischen 50% und 100% liegt. Da es sich bei diesen Werkstoffen nicht mehr um Stahl handelt, liegen auch fundamentale umformtechnische Kenngrößen, wie z.B. der Umformwiderstand, außerhalb der normalen Gültigkeitsgrenzen der Prozessmodelle. Dieses Dilemma lässt sich nur durch Berücksichtigung der lokalen Unterschiede in den Prozessmo-

dellen lösen. Eine sinnvolle Methode stellen hier die im nächsten Kapitel beschriebenen neuronalen Netze dar.

Da sich einige Analysewerte kaum auf den Restfehler auswirken, während andere einen deutlichen Einfluss haben, wird eine Hilfsgröße *Restsumme* eingeführt, die unwichtige Legierungsbestandteile zusammenfasst. Es handelt sich um die Konzentration der Elemente P, S, Cu, N, B und Sn. Durch diese Maßnahme lässt sich der Eingangsraum um fünf Dimensionen verringern.

In den bisherigen Betrachtungen wurden Anlagenparameter vernachlässigt. Die bessere Erfassung mit Hilfe des Frameworks erlaubt jedoch, auch Anlagendaten mit in die Analyse einzubeziehen. Die Rohdatenanalyse ergibt, dass gerade Walzenparameter wie der Außen- bzw. der Kerndurchmesser der Arbeits- und Stützwalzen, die E-Module der Oberfläche und des Kerns und die spezifischen Wärmekapazitäten der Walzen einen gewissen Einfluss auf die Zielgröße besitzen.

4.6.4 Bestimmung von Ausreißern

Bei der Betrachtung der Scatterplots fällt besonders bei den chemischen Analysewerten eine starke Clusterung auf. Große Unterschiede einzelner Eingangsgrößen stellen speziell bei einem möglichen späteren Training neuronaler Netze ein Problem dar. Da alle Eingänge normiert werden, wirken sich Ausreißer enorm aus und verfälschen das Ergebnis.

Angesichts des in der betrachteten Warmbreitbandstraße gewalzten Produktspektrums ist es sinnvoll, die Daten in drei Klassen zu teilen:

- Normalstahl
- Edelstahl
- Ausreißer (Sondergüten)

Eine grobe Einteilung kann man dem Scatterplot der beiden Legierungsanteile Cr und Ni entnehmen (Abbildung 4.9). Nahezu alle Datenpunkte im Cluster rund um den Koordinatenursprung gehören zur Gruppe der Normalstähle, die Gruppe der Edelstähle fasst die beiden Cluster der austenitischen (Chromgehalt 15 bis 20 %, Nickelgehalt zwischen 6% und 14%) und ferritischen Edelstähle (Chromgehalt zwischen 5% und 22% bei vernachlässigbarem Nickelgehalt) zusammen. Alle weiteren Datenpunkte außerhalb dieser Cluster stellen Ausreißer im Sinne der Statistik bzw. Sondergüten im Kontext der Umformtechnik dar.

Mit Hilfe weiterer Scatterplots der chemischen Analyse lassen sich die Grenzen finden, die jeden Datenpunkt eindeutig einer Klasse zuordnen. Um Normalstahl handelt es sich, wenn jedes Element unterhalb der $\max_{Norm}$-Grenze liegt. Ausreißer sind dadurch gekennzeichnet, dass für ein oder auch mehrere Elemente dessen

$max_{Spezial}$-Grenze überschritten wird. Alle anderen Datensätze gehören zur Gruppe der Edelstähle. Das Ergebnis der Gruppierung ist in Tabelle 4.2 zusammengefasst. An dieser Stelle sei ausdrücklich darauf hingewiesen, dass es sich nicht, obwohl die Begriffe Normalstahl und Edelstahl für die Klassen verwendet werden, um eine Klassifizierung handelt, die Güten mit unterschiedlichen umformtechnischen Eigenschaften unterscheidet. Die Klassifizierung stellt ein Ergebnis der statistischen Analyse der Prozessdaten dar.

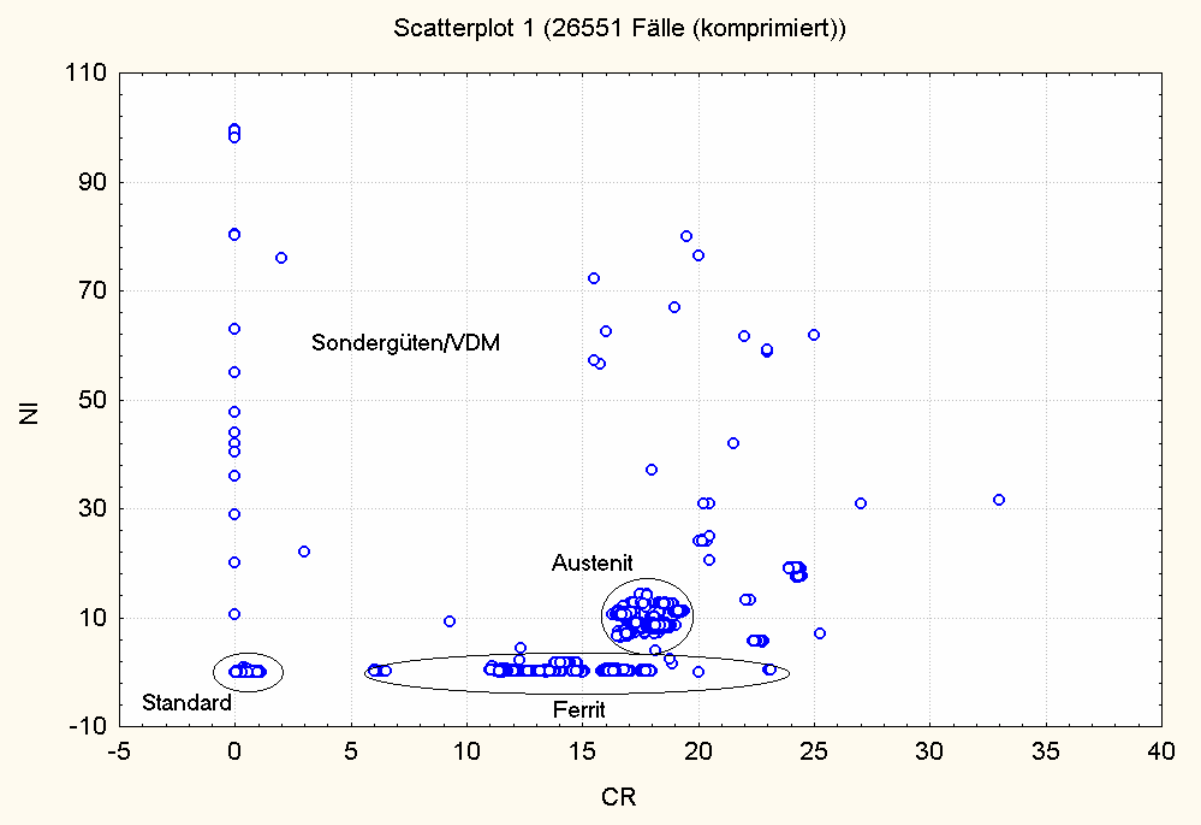


Abbildung 4.9: Aufteilung der Stahlsorten in vier Gruppen

Element	Minimum	max_{Norm}	$max_{Spezial}$
C	0 %	2 %	5 %
Si	0 %	2 %	5 %
Mn	0 %	3 %	5 %
Al	0 %	2 %	5 %
Cr	0 %	2 %	22 %
Mo	0 %	1,8 %	4 %
Ti	0 %	1 %	2 %
Ni	0 %	2 %	40 %
V	0 %	0,15 %	1 %
Nb	0 %	0,25 %	2 %
Restsumme	0 %	5 %	25 %

Tabelle 4.2: Gültigkeitsgrenzen der chemischen Analyse

Kürzel	Bezeichnung	Minimum	Maximum
VBB	Vorbandbreite	650 mm	1700 mm
VB_VBD	Vorausb. Vorbanddicke	29 mm	50 mm
VB_FBD	Vorausb. Fertigbanddicke	1,5 mm	16 mm
MW_EBT	Temperatur Auslauf FS	750 °C	1125 °C
NB_EPS7	nachb. Dickenabnahme F7	0	0,35
MW_F7	Walzkraft F7	0 kN	18000 kN
MW_V7	Walzenumfangsgeschw. F7	1,5 m/s	14,5 m/s
MW_FB7	Biegekraft F7	0 kN	750 kN
AWR7	Arbeitswalzenradius F7	320 mm	340 mm

Tabelle 4.3: Gültigkeitsgrenzen sonstiger Parameter

Die für die chemische Zusammensetzung gemachten Aussagen lassen sich nicht auf die Abmessungen des Bandes und andere Parameter übertragen. Hier können nur gültige von ungültigen Werten unterschieden werden. Tabelle 4.3 zeigt das Ergebnis der Bestimmung dieser Grenzen.

4.6.5 Sensitivitätsanalyse

Eine weitere Möglichkeit, die wichtigsten Einflussgrößen des mit dem Anstellungsmodell berechneten Gaugemeterfehlers zu ermitteln, besteht in einer Sensitivitätsanalyse. Dazu wird die Änderung der Gaugemeterdicke bei einer Variation einzelner Eingangsgrößen des Modells betrachtet. Die Sensitivität lässt sich angeben:

$$S_j^{\,p}(x_j) = \frac{\Delta y_j}{\Delta x_j} = \frac{y_j^{\,p} - y_j^{\,0}}{x_j^{\,p} - x_j^{\,0}} = \frac{h_{G,j}^{\,p} - h_{G,j}^{\,0}}{x_j^{\,p} - x_j^{\,0}} = \frac{h_{G,j}^{\,0}\left(\frac{h_{G,j}^{\,p}}{h_{G,j}^{\,0}} - 1\right)}{p \cdot x_j^{\,0}} \qquad (4.6)$$

bzw. normiert:

$$S_j^{\,p,N} = \frac{1}{p}\left(\frac{h_{G,j}^{\,p}}{h_{G,j}^{\,0}} - 1\right) \qquad (4.7)$$

mit

y^r, y^0 : Ausgangsgröße mit bzw. ohne Änderung des Eingangs
$h_G{}^r$, $h_G{}^0$: Gaugemeterdicke mit bzw. ohne Änderung des Eingangs
x^r, x^0 : Eingangsgröße der Nachberechnung mit bzw. ohne Änderung
r : relative Änderung der Eingangsgröße
p : Nummer des Datensatzes

Zur Analyse der Sensitivität werden von dieser normierten Größe die Mittelwerte und Standardabweichungen über alle Datensätze gebildet. Es ergeben sich für jede Eingangsgröße und jede Änderung dieser Größe jeweils zwei Maßzahlen. Wenn der Mittelwert für alle Änderungen einer Eingangsgröße Null oder beinahe Null ist, so besitzt diese keinen nennenswerten Einfluss auf die Ausgangsgröße. Ansonsten gibt der Mittelwert den Verstärkungsfaktor an, mit dem sich die Änderung des Eingangs am Ausgang bemerkbar macht. Wenn sich zwischen den einzelnen Mittelwerten für unterschiedliche Änderungen einer Eingangsgröße große Unterschiede ergeben, deutet dies auf eine starke Nichtlinearität zwischen den beiden Größen hin. Das gleiche gilt für große Werte der Standardabweichungen.

Die Sensitivitätsanalyse wurde unter Verwendung des im dritten Kapitel beschriebenen Frameworks unter besonderer Nutzung der verbesserten Prozessprotokollierung durchgeführt. Neben dem eigentlichen Prozessrechner (PRFSON1) steht an der betrachteten Fertigstraße ein weiterer Standby-Prozessrechner (PRFSSB2) zur Verfügung, der für Simulationen genutzt werden kann. Um die oben definierten Sensitivitäten bestimmen zu können, müssen vom Online-Prozessrechner gewalzte Bänder auf dem Standby-Rechner komplett nachgerechnet werden können, dazu benötigt dieser sämtliche *shared memories* und alle anfallenden Messwerte. Auf dem Online-Rechner wurde daher eine Datenbank installiert, in die sämtliche für die Nachberechnung, nachfolgend als RMA (rolling model adaption) bezeichnet, interessierenden Daten geschrieben werden. Die Größe dieser Datenbank ist begrenzt, so dass immer nur die letzten zehn Bänder gespeichert bleiben.

Auf dem Standby-Rechner wurde zusätzlich ein Modul ergänzt, das diese Daten ausliest, die shared memories initialisiert und die Messwerterfassung (MEWE) simuliert (MEWE-SIM). Durch Änderung der Messwerte kann das Verhalten der Prozessmodelle auf Störungen der Eingangsgrößen anhand der Änderung der Ausgangsgrößen der RMA bewertet werden. Die Art der Variation der Eingangsgrößen kann in einer Konfigurationsdatei angegeben werden.

Die Ergebnisse der Simulationsrechnung werden von der RMA automatisch in einem Archiv abgelegt. Um diese Resultate mit Hilfe konventioneller Statistik-Software auf dem PC auswerten zu können, wurde ein weiteres Modul (MEWE-SIM-Protokoll) entwickelt, das die Daten aus dem Archiv ausliest und in einer ODBC-Datenquelle ablegt (Abbildung 4.10).

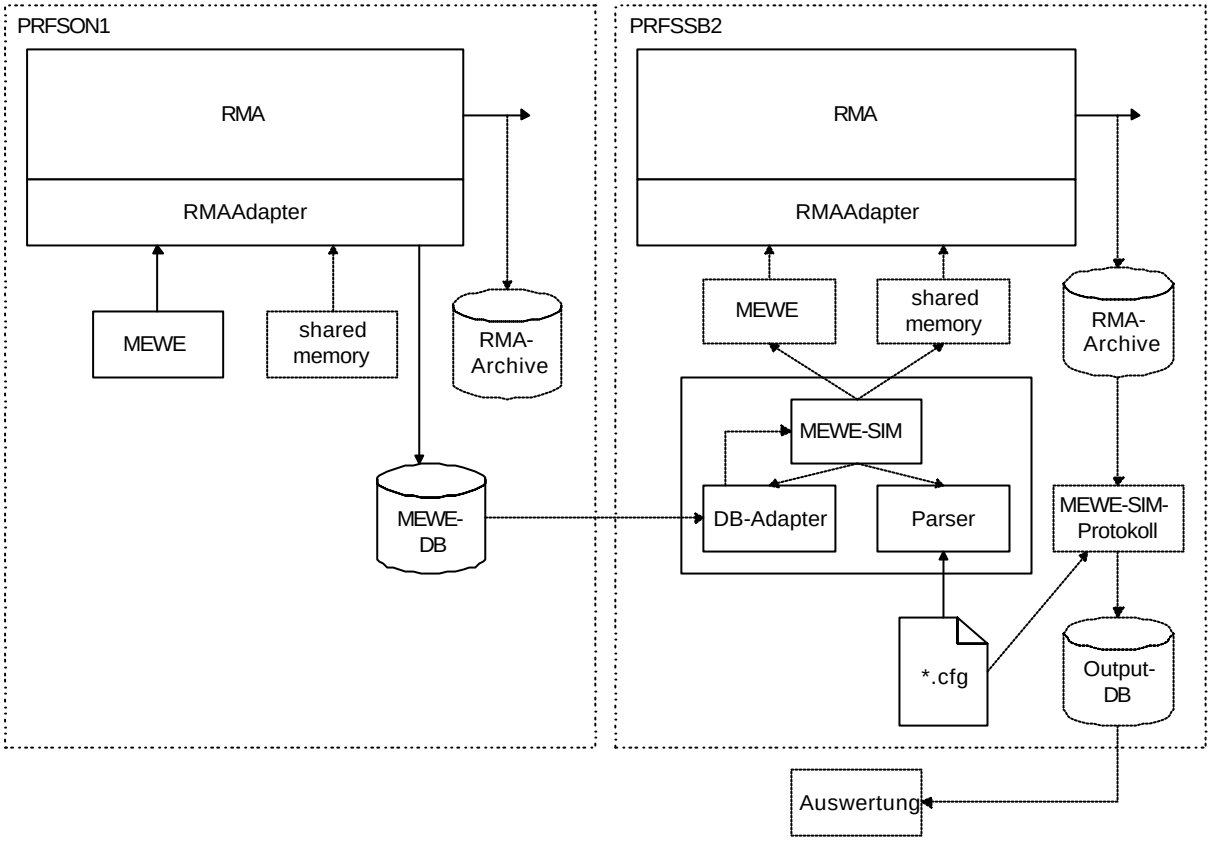


Abbildung 4.10: Systemstruktur für die Sensitivitätsanalyse

Die Sensitivitätsanalyse bestätigt die Ergebnisse der deskriptiven Statistik, dass die wichtigsten Einflussgrößen auf den Gaugemeterfehler die Walzkraft, die Walzenumfangsgeschwindigkeit und die Anstellung sind. Da immer nur eine Eingangsgröße geändert wird, lässt sich nur bei der Walzenumfangsgeschwindigkeit nachweisen, dass auch das vorangegangene Gerüst einen Einfluss auf den Fehler hat.

Insgesamt zeigt sich, dass der Prozessrechner sehr robust reagiert und auf Änderungen in einzelnen Eingangsgrößen sehr schnell auf unplausible Werte schließt. In diesen Fällen ändert sich das Ergebnis des berechneten Gaugemeterfehlers überhaupt nicht.

4.7 Clusteranalyse mit den Daten des Neusystems

Die Analyse der Eingangsgrößen mit Hilfe von Scatterplots ermöglicht die Auswahl geeigneter Inputs und die Definition von Gültigkeitsgrenzen. Da die meisten neuronalen Netze eine Zusammenschaltung verschiedener lokaler Modelle darstellen, benötigen sie zusätzlich Informationen über die Topologie des Eingangsraums. Ein geeignetes Mittel stellt in diesem Zusammenhang eine Clusteranalyse dar. Die nachfolgende Clusteranalyse vergleicht Ergebnisse, die mit unterschiedlichen Verfahren erzielt werden. Ziel jedes Algorithmus ist es, eine vorgegebene Anzahl

Zentren zu ermitteln, die den Eingangsraum möglichst gut repräsentieren. Die verschieden Verfahren der Clusteranalyse sind in Abschnitt 2.3.5 beschrieben.

4.7.1 Ergebnisse Cluster-Verfahren

In einer ersten Versuchsreihe wird das k-Means-Verfahren und der AKM-Algorithmus getestet. Mit beiden Verfahren sollen die Cluster des gesamten Eingangsraums gefunden werden. Es werden wiederum die 22082 Datensätze verwendet, die auch schon bei der statistischen Analyse zum Einsatz gekommen sind. In jeweils elf Versuchen werden mit den Verfahren zwischen 5 und 100 Zentren gesucht.

Ergebnis der Experimente ist, dass lediglich die Eingänge der chemischen Analyse stark geclustert sind und die übrigen Eingangsgrößen kaum Klassenbildung zeigen. Bereits 5 bis 10 Zentren reichen hier aus, um den Eingangsraum darzustellen. Im Gegensatz dazu ergibt sich bei Verwendung des k-Means-Verfahrens, dass mindestens 80 Zentren gewählt werden müssen, um die Verteilung der Legierungsanteile darzustellen. Das AKM-Verfahren, bei dem etwa 60 Zentren genügen, ist insgesamt etwas besser.

Durch die große Spannweite einzelner Legierungsanteile – der Mn-Anteil, der üblicherweise bei unter 0,1% liegt, kann bei Sondergüten über 70% betragen – ergeben sich in allen Fällen numerische Probleme. Die Zentren, die Datenpunkte repräsentieren sollen, die relativ weit vom größten Häufungspunkt entfernt liegen, liegen zu nah an diesem. Mit Hilfe der Ergebnisse aus dem vorherigen Abschnitt werden die Daten daher in zwei Gruppen aufgeteilt. Die Gruppe der Normalstähle umfasst 14015 Datensätze; die Gruppen der Edelstähle und Sondergüten werden zusammengefasst und bestehen aus insgesamt 7797 Fällen.

Die derart durchgeführte zweite Cluster-Analyse, deren Ergebnis in den Abbildung 4.11 bis Abbildung 4.14 dargestellt ist, zeigt keine numerischen Probleme mehr, und die Verteilung der Zentren entspricht der Verteilung der Daten. Das AKM-Verfahren benötigt jeweils etwa 50 Zentren, um den Eingangsraum repräsentativ abzudecken. Ähnlich gut ist das Neural-Gas-Verfahren. Das k-Means-Verfahren benötigt wiederum etwa 20 Zentren mehr.

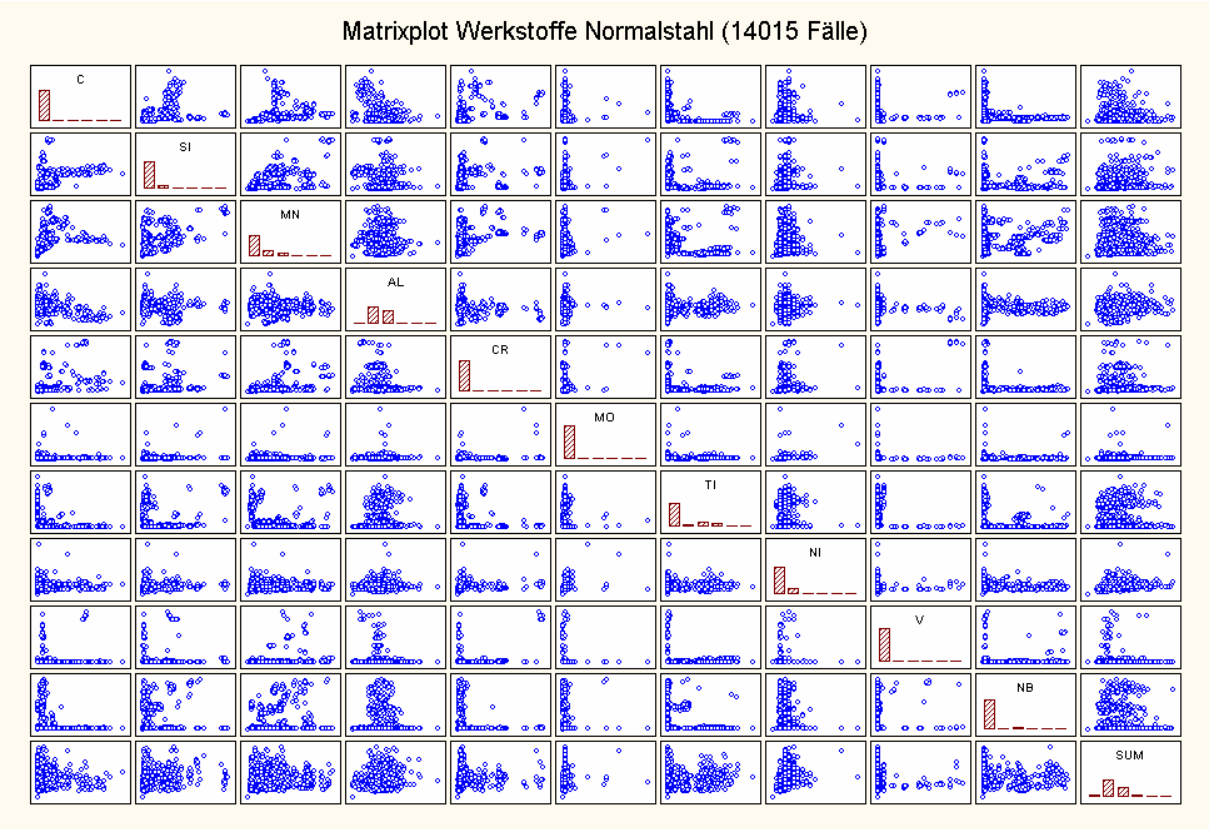


Abbildung 4.11: Matrixplot Werkstoffe Normalstahl

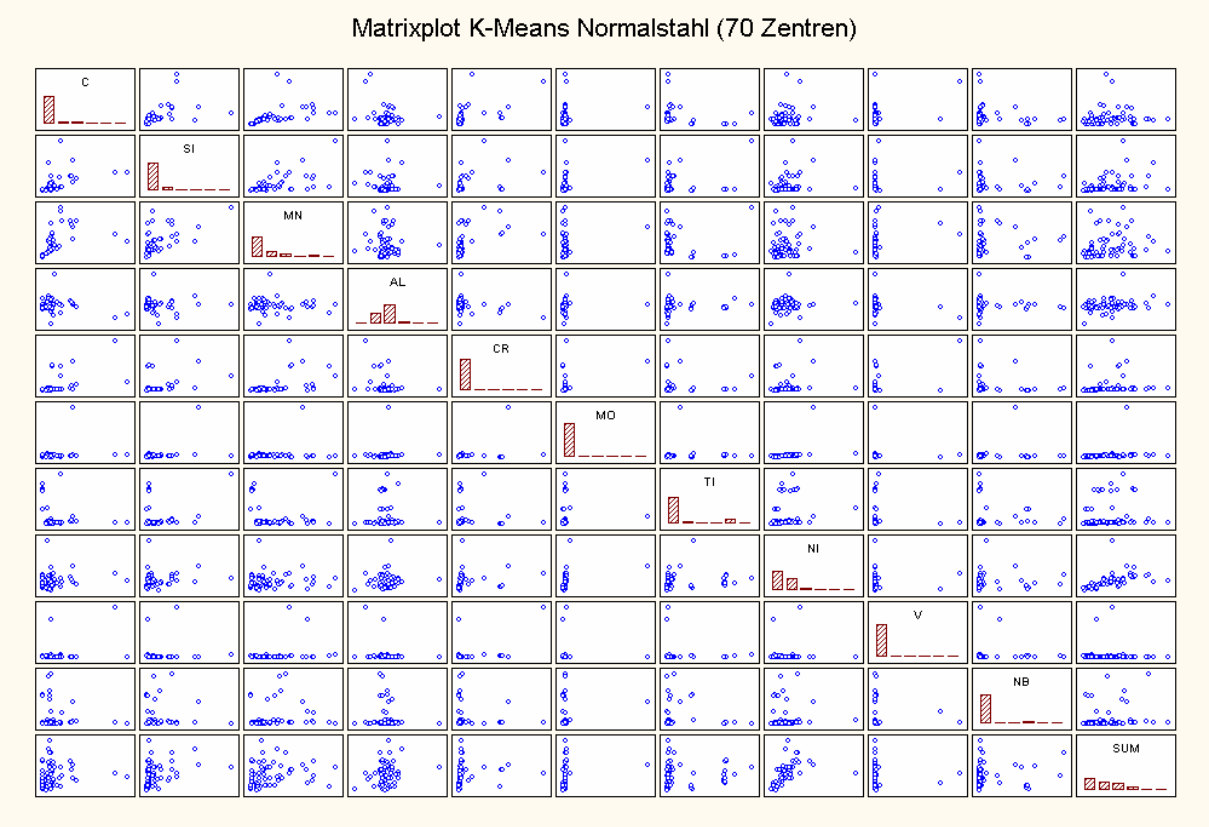


Abbildung 4.12: Matrixplot K-Means-Zentren Werkstoffe Normalstahl

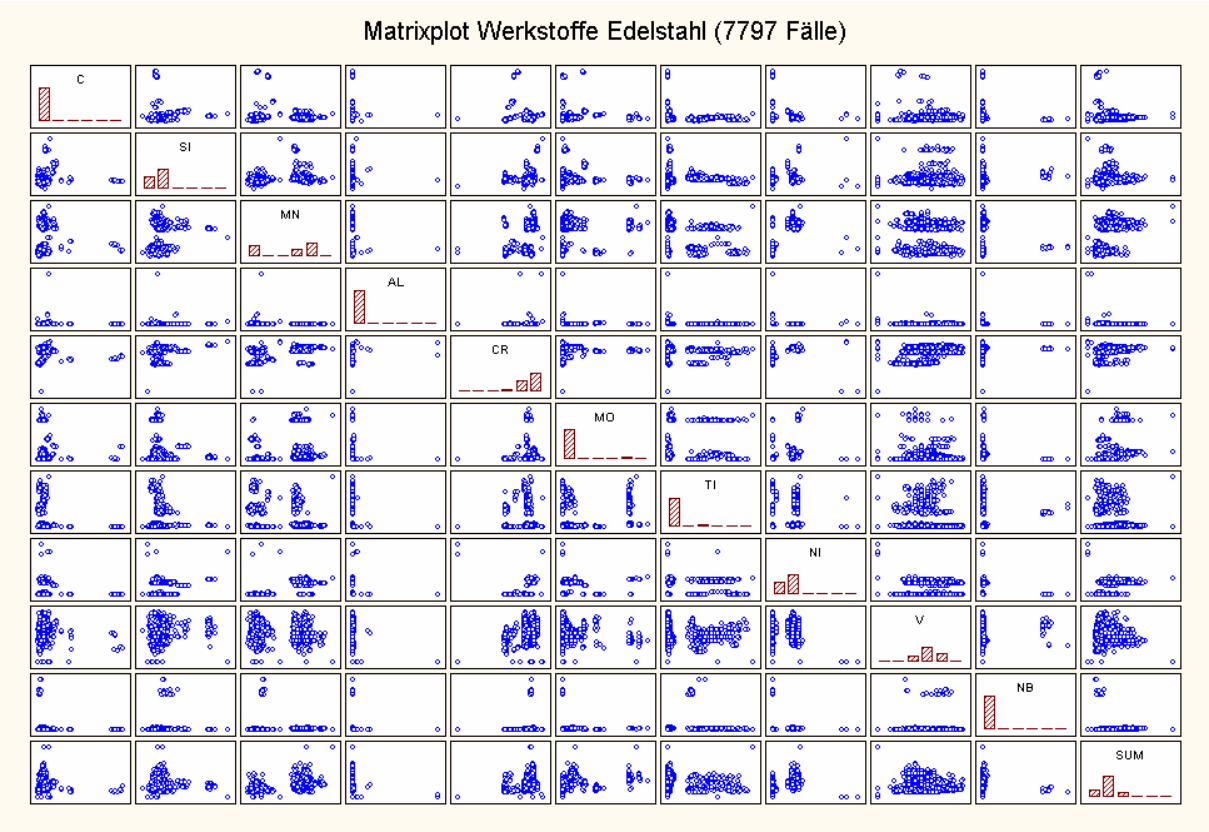


Abbildung 4.13: Matrixplot Werkstoffe Edelstahl

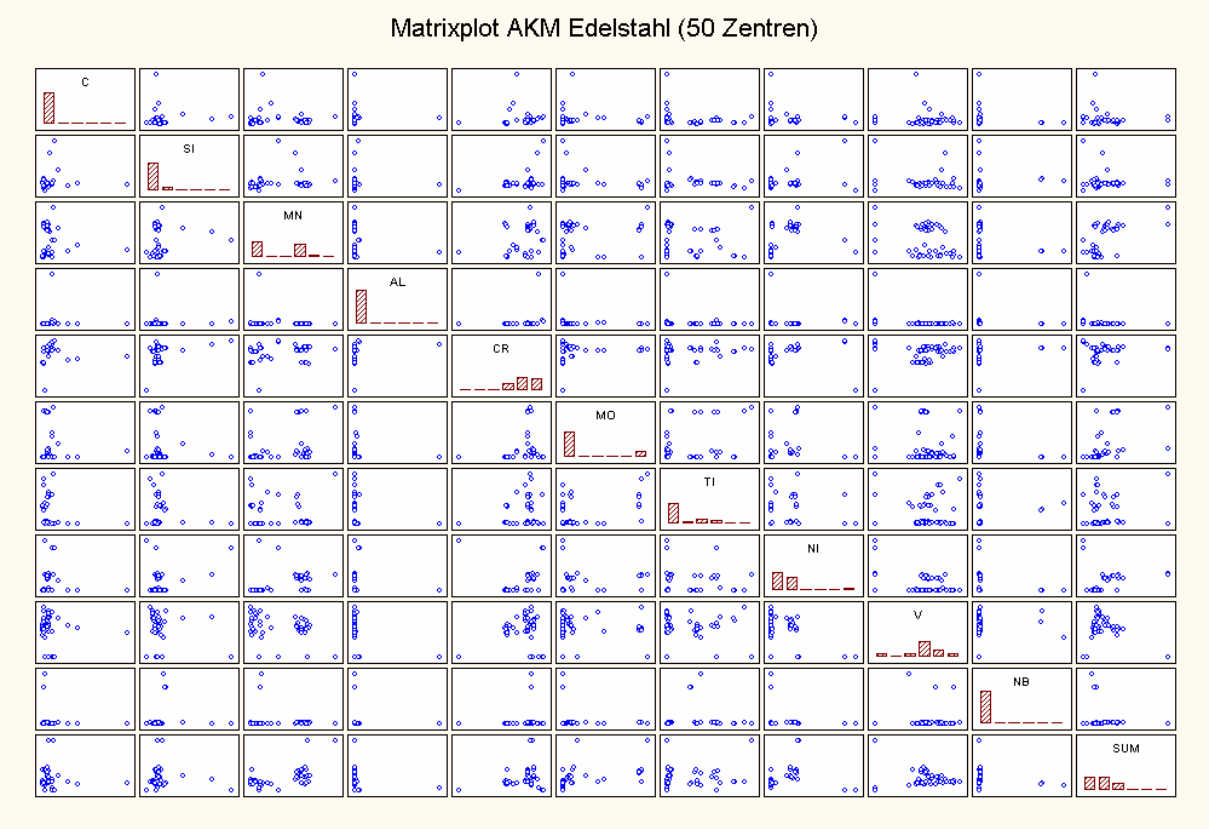


Abbildung 4.14: Matrixplot AKM-Zentren Werkstoffe Edelstahl

4.7.2 Folgerungen

Die Clusteranalyse hat gezeigt, dass das Werkstoffspektrum der betrachteten Anlage zu numerischen Problemen im Algorithmus führen kann. Diese Probleme sind auch bei weiteren statistischen Methoden, wie neuronalen Netzen, zu erwarten. Wenn der Datenbestand jedoch in zwei oder drei Gruppen unterteilt wird, lassen sich diese Probleme vermeiden. In diesem Fall würde bei der Verwendung eines RBF-Netzes ein Vortraining von 50 bis 60 Zentren mit AKM oder Neural Gas ausreichen, wenn zwei getrennte Netze realisiert werden.

5 Optimierung mit neuronalen Netzen

5.1 Motivation

Die im vorigen Kapitel beschriebene Analyse der Daten hat Verbesserungspotentiale der eingesetzten Prozessmodelle aufgezeigt. Dies gilt in besonderem Maße für die Anlage vor der Modernisierung. Die Umstellung einer Anlage auf eine neue Prozessautomatisierung wirft immer auch die Frage auf, ob eine solche Neuimplementierung nicht auch durch die Optimierung der bestehenden Prozessmodelle hätte erzielt werden können. Unter diesem Gesichtspunkt werden nachfolgend zwei Optimierungen beschrieben: die Optimierung des Altsystems und die Optimierung des Neusystems.

Die Optimierung der Prozessmodelle der alten Prozessautomatisierung wird in verschiedensten Versuchen durchgeführt, bei denen die Zielsetzung jeweils darin besteht, mit statistischen Methoden oder neuronalen Netzen existierende Modelle zu verbessern. Die Ergebnisse der neuen Prozessautomatisierung – ohne zusätzliche Optimierung – werden als zusätzliche Referenz mit hinzugezogen, um die Qualität der erzielten Verbesserung mit anderen Alternativen vergleichen zu können.

Neuronale Netze als Alternative für adaptive Komponenten von analytischen Modellen sind eine Möglichkeit der Optimierung. Eine andere Variante sieht reine neuronale Netze ohne physikalischen Modellkern vor. Mit Hilfe dieser Modelle lässt sich sehr gut abschätzen, wie gut eine Optimierung mit hybriden Modellen werden kann und welche Optimierungspotentiale ein existierendes Modell bietet.

Die Optimierung der neuen Prozessmodelle, mit den bereits bei der alten Prozessautomatisierung angewandten Methoden, soll die unterschiedliche Reaktion auf additive Korrekturnetze dokumentieren.

5.2 Vorüberlegung zur Optimierung mit neuronalen Netzen

Wie bereits im vorangegangenen Abschnitt beschrieben, bieten sich zur Lösung von Approximationsproblemen, die bei Prozessmodellen auftreten, zwei Netztypen an: Multi Layer Perceptrons (MLPs) und verschiedene Formen der so genannten Radial-Basis-Funktionen-Netze (RBF).

Um verschiedene Varianten der Netze vergleichen zu können, wurden mehrere Versuchsreichen durchgeführt, die zusammen mit den erzielten Ergebnissen in den folgenden Abschnitten beschrieben werden.

Alle Experimente wurden zunächst nur mit MLPs durchgeführt. Eine Normierung des Eingangs wird bei allen Netzen vorgenommen, da sich aufgrund der stark unterschiedlichen Größenordnungen der Eingangsgrößen ansonsten numerische Probleme ergeben, die zu wenig homogenen Netzen führen. Die Normierung beeinflusst indirekt die Gleichmäßigkeit der Netzgewichte, die dann eine eingeschränkte Interpretierbarkeit ermöglicht. Allen Experimenten gemeinsam ist zudem, dass nur eine verdeckte Schicht genutzt wird. Es kommen jeweils die 22082 Datensätze zum Einsatz, die in den ersten 8 Kalenderwochen des Jahres 1999 gesammelt wurden. Als Eingangsgrößen werden die 20 in Tabelle 4.2 und Tabelle 4.3 definierten Parameter verwendet. Als mittlerer Netzfehler wird der RMSE (3.8) verwendet; für alle Netze wird auf die vom Netz vorgegebene Lernrate zurückgegriffen, die sich nach Heuristiken berechnet, die in [Jansen1995] näher beschrieben sind.

5.2.1 Normierung des Ausgangs

Das erste Experiment vergleicht ein MLP20 - d.h. ein MLP mit 20 Neuronen in der verdeckten Schicht - mit Normierung des Ausgangs mit einem ohne Normierung des Ausgangs.

Es zeigt sich, dass auch eine Normierung des Ausgangs sinnvoll ist, da der mittlere Fehler außerhalb des Netzes korrigiert werden kann und sich die Netzausgabe besser interpretieren lässt. Der RMSE (root mean square error) entspricht in diesem Fall der relativen Verbesserung bzw. Verschlechterung mit Hilfe des Netzes.

5.2.2 Variation der Lernrate

Das zweite Experiment vergleicht ein MLP20, das mit einer typischen Lernrate trainiert wird, zum einen mit einem Netz mit einer um einen Faktor 10 größeren Lernrate und zum anderen mit einem Netz, bei dem die Lernrate um einen Faktor 10 geringer gewählt wird. Hierbei ist anzumerken, dass die (Default-)Lernrate vom Framework vorgegeben wird und nur von der Anzahl der Neuronen abhängt. In allen Fällen werden den Netzen pro Adaption 20 Muster präsentiert. Außerdem werden jeweils unterschiedliche Netze für Normal- und Edelstahl trainiert.

Die große Lernrate führt sowohl beim Normalstahl- als auch beim Sondergütennetz zu einem sehr schlechten Trainingsergebnis im Vergleich zum normalen MLP20. Zusätzlich zeigen sich sehr starke Oszillationen in der Kurve des RMSE der Testdaten.

Eine kleinere Lernrate führt in erster Linie zu einer glätteren Kurve des RMSE. Dadurch lässt sich der Punkt, an dem ein „Overlearning“ eintritt, also der Punkt, an dem der Testfehler wieder ansteigt, besser bestimmen. Allerdings fällt besonders

beim Edelstahlnetz auf, dass das Netz deutlich langsamer lernt; somit sind die Trainingsresultate nach 200 Epochen schlechter als beim normalen MLP20.

5.2.3 Variation der Netzgröße

Um den Einfluss der Netzgröße auf das Trainingsergebnis zu erhalten, werden im dritten Experiment verschiedene MLPs trainiert, bei denen die Anzahl der Neuronen in der verdeckten Schicht variiert wird; dabei wird die Lernrate jeweils der Änderung der Neuronenzahl entsprechend der Ergebnisse der letzten Experimente angepasst. In allen Fällen werden den Netzen pro Adaption wiederum 20 Muster präsentiert.

Die Normalstahl-Netze beginnen erst ab 15 Neuronen in der verdeckten Schicht zu lernen. Für kleinere Netze ist nur Rauschen zu erkennen. Die Vergrößerung des Netzes über 20 Neuronen hinaus führt zwar zu einer Verbesserung des Trainingsfehlers, der für die Generalisierungsleistung aussagekräftigere Testfehler bleibt jedoch konstant.

Ein ähnliches Ergebnis ergibt sich auch für die Edelstahl-Netze, bei denen zwar schon bei 10 Neuronen in der verdeckten Schicht ein Lernerfolg feststellbar ist, insgesamt stellt aber auch hier ein MLP20 das optimale Netz dar, da die Generalisierungsleistung für größere Netze nicht mehr steigt.

5.2.4 Variation der Batch-Länge

In einer letzten Experimentenreihe werden verschiedene MLPs mit 20 Neuronen mit konstanter Lernrate bei Variation der Batch-Länge trainiert. Die Batch-Länge bezeichnet dabei die Anzahl der Muster, die dem Netz pro Adaption präsentiert werden.

Sehr kurze Batch-Längen führen zu einem starken Rauschen des Testfehlers. Das Verhalten ähnelt dem Training mit großen Lernraten. Ein sehr ausgeprägtes Batch-Training erzeugt eine glatte Kurve des RMS-Fehlers, bewirkt aber auch ein langsameres Training und entspricht somit dem Training mit sehr kleinen Lernraten.

Insgesamt lässt sich festhalten, dass die optimale Batch-Länge für beide Netze etwa 100 beträgt und somit etwas oberhalb der für alle anderen Experimente gewählten 20 Muster pro Adaption liegt. Diese Aussage gilt allerdings nur für normale Lernraten, da die optimale Batch-Länge von der gewählten Lernrate abhängt.

5.2.5 Folgerung

Das in allen Experimenten als Referenznetz gewählt MLP20 stellt bereits eine recht gute Lösung dar, dessen Ergebnisse sich vor allem dann noch verbessern lassen, wenn die Batch-Länge erhöht wird.

Der größte Nachteil, den MLPs besitzen, ist die mangelhafte Online-Adaptierbarkeit. Da es sich um nichtlineare Modelle handelt, ist eine Adaption der Parameter nicht in jedem Fall stabil. In diesem Fall bietet sich die Verwendung der RBF-Netze an. Überlegung zu RBF-Netzen finden sich in Abschnitt 4.7.1.

5.3 Optimierung der alten Prozessautomatisierung

Die Optimierung der alten Prozessautomatisierung mit neuronalen Netzen stellt eine einfache Möglichkeit der Verbesserung dar, die keinen Eingriff in grundlegende Prozessmodelle ermöglicht. Eine solche Optimierung bietet im Wesentlichen drei Vorteile: Zum einen kann das produktive Prozessmodell mit einem „Black Box"-Netz verglichen werden und liefert damit direkte Hinweise über die Güte des aktuellen Modells, weiterhin kann mit neuronalen Restfehlernetzen eine Prozessautomatisierung verbessert werden, die Optimierungspotentiale in den verwendeten Modellen dokumentiert, und nicht zuletzt entsteht eine Referenz für die parallel durchgeführt Optimierung mit neuen Prozessmodellen.

5.3.1 Versuchdurchführung

Alle neuronalen Netze nutzen das in Kapitel 3 beschriebene Framework. Wie weiter oben ausgeführt, dienen die hier vorgestellten Experimente nur als Referenz für die Optimierung der neuen Prozessautomatisierung, daher wird in allen Versuchen eine einheitliche Versuchsanordnung verwendet. Auf eine spezielle Optimierung der neuronalen Netze in Hinblick auf optimale Netztopologie, optimales Lernverfahren oder optimalen Eingangsvektor wird daher verzichtet; es werden stattdessen die Ergebnisse der im Rahmen der Vorüberlegungen durchgeführten Experimente berücksichtigt.

Alle Optimierungen nutzen Multi Layer Perceptrons (MLPs). Da es sich bei allen Experimenten um Offline-Betrachtungen handelt, werden keine alternativen Netztypen, wie z.B. RBF-Netze, betrachtet, weil diese nur Vorteile im Online-Verhalten zeigen. Es werden jeweils hstanzen mit 5, 10, 20, 40 und 60 Neuronen in der verdeckten Schicht verwendet, dabei werden alle Netze genau 200 Epochen mit 20.000 Datensätzen trainiert, zur Verifizierung stehen 2000 weitere Datensätze bereit. Da mit den Daten des Altsystems keine Clusteranalyse durchgeführt wurde, wird auf eine Unterteilung der Daten in Normal- und Edelstahl verzichtet, und entsprechend werden alle Netze mit allen Daten trainiert. Es sei an dieser Stelle noch einmal darauf hingewiesen, dass die Stichprobengröße mit 22.000 Datensätzen nur zufällig annähernd so groß ist wie in den vorherigen Versuchen und dass es sich um Daten des Altsystems handelt, die im August und September des Jahres 1997 erfasst wurden.

In allen Experimenten werden als Netzeingänge die unterschiedlichen Legierungsanteile des Walzgutes, die Walzgutbreite, die Walzgutdicke, die Walzgeschwindigkeit, der Arbeitswalzenradius und die relative Abnahme verwendet. Nachfolgend sind die Netze für folgende Zielgrößen beschrieben:

- Anstellung
- Anstellungsfehler
- Walzkraft
- Walzkraftfehler

Alle Versuche beziehen sich auf das letzte Gerüst in der Fertigstraße der betrachteten Warmbreitbandstraße.

5.3.2 Anstellungsnetze

Das Ziel der Anstellungsnetze besteht darin, die Anstellung eines Walzgerüstes aus vorgegebenen Eingangsgrößen, wie z.B. der Solldicke, zu bestimmen, um diese entsprechend voreinzustellen. Das neuronale Netz soll also die komplette Prozessautomatisierung ersetzen. Die Struktur der Anstellungsnetze entspricht dabei dem im nächsten Abschnitt vorgestellten Anstellungsfehlernetz GRENN I. Als Ergebnis zeigt sich, dass die Anstellungsnetze den Walzprozess fast genauso gut beschreiben können wie die eingesetzten Prozessmodelle. Die durchschnittliche Zunahme des Dickenfehlers liegt je nach Netztyp zwischen zwei und fünf Prozent. Mit dem gewählten Ansatz lässt sich jedoch zunächst kein Netz finden, dass den Dickenfehler minimiert, daher wird im nächsten Schritt dazu übergegangen, nicht die Anstellung sondern nur den Fehler der Anstellung vorherzusagen.

5.3.3 Anstellungsfehlernetze

Anstellungsfehlernetze - nachfolgend auch als GRENN (gaugemeter residual error neural network) bezeichnet - stellen eine Ergänzung herkömmlicher Modelle dar, mit denen Anstellungsfehler vorhergesagt werden. Die Kombination aus Anstellungsfehlernetz und Prozessautomatisierung ergibt ein hybrides Modell für die Anstellung. In Abbildung 5.1 ist die Struktur des Offline-Netzes GRENN I dargestellt, das bereits in Abschnitt 4.5.4 zur Datenanalyse beschrieben wurde. Im Gegensatz zum Offline-Netz GRENN I, das das Ergebnis nur protokolliert, wird beim in Abbildung 5.2 dargestellten Online-Netz GRENN II der Netzoutput in den Prozess zurückgekoppelt.

Wie aus Tabelle 5.1 zu entnehmen ist, kann bereits ein sehr einfaches MLP mit nur 5 Neuronen in der verdeckten Schicht für eine deutliche Verbesserung des Anstellungsfehlers sorgen. Die Erhöhung der Anzahl an freien Parametern sorgt nur noch dafür, dass der mittlere Anstellungsfehler nahezu verschwindet.

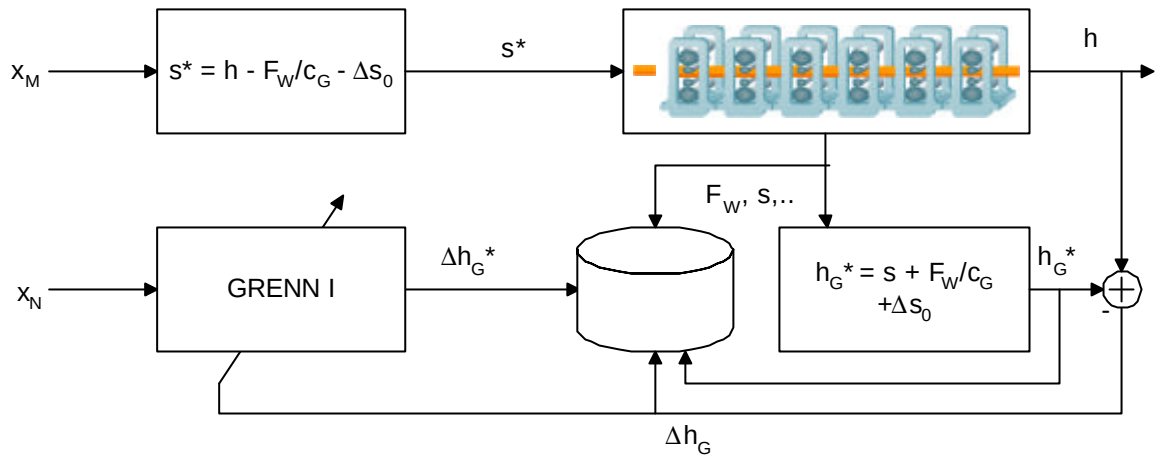


Abbildung 5.1: Struktur des Offline-Anstellungskorrekturnetzes GRENN I

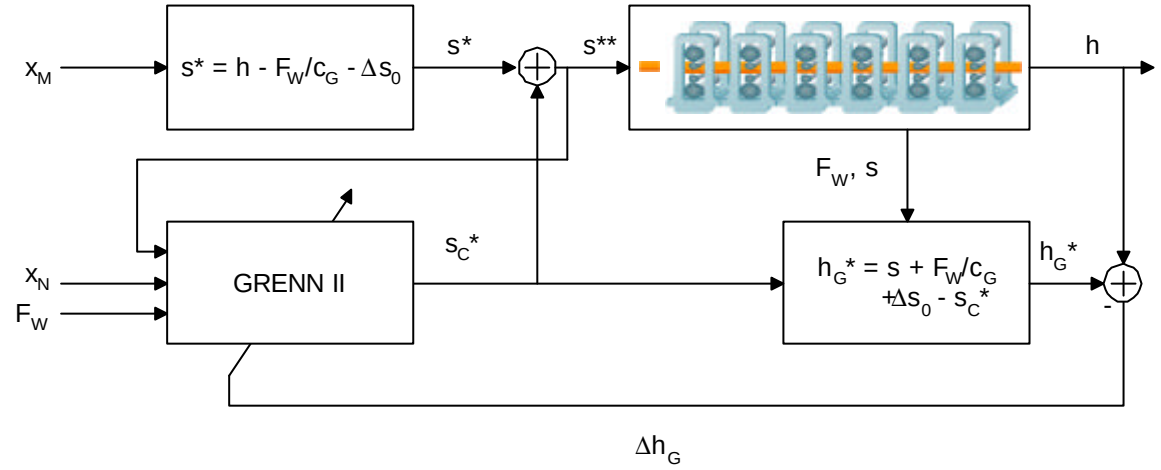


Abbildung 5.2: Struktur des Anstellungskorrekturnetzes GRENN II

	Mittelwert (mm)	Standardabweichung (mm)
Anstellungsfehler	-0,004924	0,048658
Ergebnis MLP5	-0,006759	0,018128
Ergebnis MLP60	-0,005613	0,01870
Netzfehler MLP5	0,001835	0,044954
Netzfehler MLP60	0,000684	0,044654

Tabelle 5.1: Ergebnisse des Anstellungsfehlernetzes

Festzuhalten bleibt, dass mit Hilfe eines Anstellungsfehlernetzes der mittlere Anstellungsfehler, den die normale Anstellungsadaption eigentlich schon unterdrücken sollte, nahezu vollständig eliminiert werden kann, zudem lässt sich der Dickenfehler um etwa 10 % verbessern.

5.3.4 Walzkraftnetze

Walzkraftnetze versuchen, die Walzkraft, die bei der Bestimmung der Anstellung nur eine Zwischengröße darstellt, die sich aber direkt messen lässt, mit einem neurona-

len Netz zu bestimmen. Das größte Problem von Walzkraftnetzen – wie bei allen neuronalen Netzen - ist die fehlende physikalische Interpretierbarkeit des Ergebnisses.

Wie aufgrund der Ergebnisse der Anstellungsnetze zu erwarten, erzielen auch die Walzkraftnetze nicht die Güte der bisher eingesetzten analytischen Modelle. Zwar sorgen schon sehr einfache Netze für eine Optimierung des mittleren Fehlers und geben somit Hinweise auf Modellfehler, letztlich sind sie aber nicht in der Lage, die Standardabweichung des Walzkraftfehlers zu minimieren. Es zeigt sich demnach, dass es schwierig ist, mit einfachen neuronalen Netzen das Ergebnis von physikalisch-mathematischen Prozessmodellen zu erreichen. Motiviert durch die bei den Anstellungsfehlernetzen erzielten Resultate versprechen auch hier Walzkraftfehlernetze bessere Ergebnisse.

	Mittelwert (KN)	Standardabweichung (KN)
Walzkraft Ist (Target)	7462,55	3049,45
Walzkraft Vorausberechnet	7285,88	2921,90
Walzkraftfehler	176,67	818,34
Ergebnis MLP5	7478,47	2975,44
Ergebnis MLP60	7477,86	2954,20
Netzfehler MLP5	-15,92	958,02
Netzfehler MLP60	-15,30	929,77

Tabelle 5.2: Ergebnisse des Walzkraftnetzes

5.3.5 Walzkraftfehlernetze

Walzkraftfehlernetze modellieren den Fehler des mathematischen Walzkraftmodells der Prozessautomatisierung. Die Kombination aus neuronalem Netz und mathematischem Modell ergibt dann ein hybrides Modell für die Walzkraft.

	Mittelwert (KN)	Standardabweichung (KN)
Walzkraftfehler	-176,67	818,34
Ergebnis MLP5	-227,38	245,80
Ergebnis MLP60	-176,92	345,75
Netzfehler MLP5	50,71	781,38
Netzfehler MLP60	0,25	743,17

Tabelle 5.3: Ergebnisse des Walzkraftfehlernetzes

Die in Tabelle 5.3 dargestellten Ergebnisse des Trainings der Walzkraftfehlernetze übertreffen die Erwartung bei weitem. Der mittlere Walzkraftfehler kann wie schon von den zuvor beschriebenen Walzkraftnetzen sehr genau bestimmt werden. Der große Vorteil der Walzkraftfehlernetze liegt darin, dass sie außerdem die Standard-

abweichung des Walzkraftfehlers um bis zu 10 % minimieren können. Die Tatsache, dass neuronale Netze mit vielen Neuronen hier noch deutliche Verbesserungen bringen, spricht dafür, dass das eingesetzte physikalisch-mathematische Prozessmodell Nichtlinearitäten nicht genügend berücksichtigt. Beim Walzkraftmodell werden diese vor allem durch den Vektor der Stahllegierungsanteile verursacht.

5.3.6 Zusammenfassung

Das Training der neuronalen Netze des alten Prozessrechners hat zwei wichtige Erkenntnisse geliefert: Einerseits ist die Beschreibung eines Modells mit physikalischen Modellen immer einem rein statistischen Verfahren vorzuziehen, andererseits kann aber die Optimierung eines bestehenden physikalischen Modells mit einem neuronalen Restfehlernetz das Ergebnis in vielen Fällen verbessern.

5.4 Optimierung der neuen Prozessautomatisierung

Die mögliche Optimierung der alten Prozessautomatisierung mit neuronalen Netzen motiviert zur Wiederholung dieser Experimente für das Neusystem, um zu überprüfen, ob sich die dort gefundenen Erkenntnisse bestätigen lassen. Dazu werden wiederum zunächst Anstellungsnetze und Walzkraftnetze und anschließend Fehlernetze trainiert. Die Struktur der neuronalen Netze entspricht den Netzen des Altsystems. Es werden wiederum die in den ersten 8 Kalenderwochen des Jahres 1999 erfassten Daten für die Experimente verwendet.

Die Anstellungsnetze und die Walzkraftnetze erzielen absolut annähernd die gleichen Ergebnisse wie die entsprechenden Netze, die mit den Daten des Altsystems trainiert wurden. Da die neuen Prozessmodelle jedoch etwa 15% besser sind als die alten, ist die Differenz zwischen reiner neuronaler Modellierung und analytischer Modellierung in diesem Fall noch deutlich größer. Dies unterstreicht das schon zuvor gefundene Ergebnis, dass die Modellierung mit reinen neuronalen Netzen nur bei völliger Unkenntnis des Prozesses sinnvoll ist.

Die Anstellungsfehlernetze und die Walzkraftfehlernetze können das Ergebnis der neuen Prozessautomatisierung nicht nennenswert verbessern. Die Menge von fehlerhaften Prognosen ist so hoch, dass ein praktischer Einsatz mit den getesteten Netzen nicht sinnvoll erscheint. Die Netze müssten noch weiter optimiert werden, um einen Einsatz zu rechtfertigen. Eine viel versprechende Alternative besteht aber darin, die bereits in der Prozessautomatisierung integrierten Restfehlernetze zu optimieren.

5.5 Optimierung der bestehenden neuronalen Netze

5.5.1 Motivation

Die Datenanalyse hat gezeigt, dass durch eine optimierte Clusteranalyse noch einige Potentiale in bereits eingesetzten neuronalen Netzen stecken. Nachfolgend wird daher eine solche Optimierung exemplarisch für die neuronalen Netze des Walzkraftmodells beschrieben.

5.5.2 Optimierung der Walzkraftnetze

Die in Kapitel 4.7 beschriebene Clusteranalyse der Daten des Neusystems zeigt, dass es sich um ein stark geclustertes Werkstoffspektrum handelt, das sich grob in zwei Gruppen einteilen lässt. Idealerweise sollten die vorhandenen neuronalen Netze diese Gruppen auch unterschiedlich behandeln. Ein solcher Schritt ist in den bisher implementierten neuronalen Netzen jedoch nicht vorgesehen.

Daher werden Möglichkeiten gesucht, die Ergebnisse der Clusteranalyse zu berücksichtigen. Es lassen sich drei Varianten der Optimierung der Walzkraftnetze unterscheiden:

1. Analyse der Werkstoffe mit anschließender Ergänzung nicht erkannter Cluster
2. Änderung des Legierungsvektors und Neutraining des Netzes
3. Aufteilung auf zwei neuronale Netze

Im ersten Fall werden Bänder eines bestimmten Werkstoffes, der sich in vorhergehenden Walzungen als problematisch erwiesen hat, vom Prozessmodell erkannt und entsprechend behandelt. Es erklärt sich von selbst, dass diese Methode nur für einige wenige Werkstoffe durchgeführt werden kann, da ansonsten die Anzahl der Cluster derart steigen würde, dass auch die RBF-Netze Stabilitätsprobleme bekommen.

Im zweiten Fall wird, um dieses Problem zu vermeiden, der Vektor der Stahllegierungen geändert. Wie in Kapitel 4.7 beschrieben, besteht die Herausforderung darin, dass die Algorithmen der Clusterverfahren Schwierigkeiten mit Daten besitzen, die entsprechend der statistischen Definition eigentlich als Ausreißer gelten. Dies gilt vor allem für den Vektor der Stahllegierungen. Einige Versuche zeigen nun, dass bereits die Vorverarbeitung mit einer nichtlinearen Funktion zu beachtlichen Verbesserungen führt. Durch eigene Experimente hat sich vor allem folgende Funktion als praktikabel herausgestellt:

$$y_i = \ln (1 + x_i) \qquad (5.1)$$

Die dritte Alternative besteht im Training zweier Netze. Wie die Clusteranalyse gezeigt hat, treten die numerischen Probleme nicht auf, wenn Normalstahl und Edelstahl getrennt behandelt werden. Der Nachteil dieser Variante liegt darin, dass ein solches Netz nur noch sehr schwierig online trainierbar ist und somit kaum auf Änderung der Anlage reagieren kann.

5.5.3 Ergebnisse der Optimierung

Ausgehend von den zuvor beschriebenen Möglichkeiten wurden die Walzkraftnetze in drei Stufen optimiert:

1. Optimierung der Lage existierender Zentren
2. Hinzufügen von zusätzlichen Zentren
3. Änderung der Eingangsnormierung

Im ersten Schritt wurden redundante Zentren durch Zentren in unterbesetzten Gebieten ersetzt. Dieser Optimierungsschritt ist in Abbildung 5.3 an der deutlichen Verbesserung der Dickentreffsicherheit – das ist der prozentuale Anteil aller Bänder, der in einem vorgegebenen Toleranzbereich des Dickenfehlers liegt - von Normalstahl im Zeitraum September bis November 1999 zu erkennen.

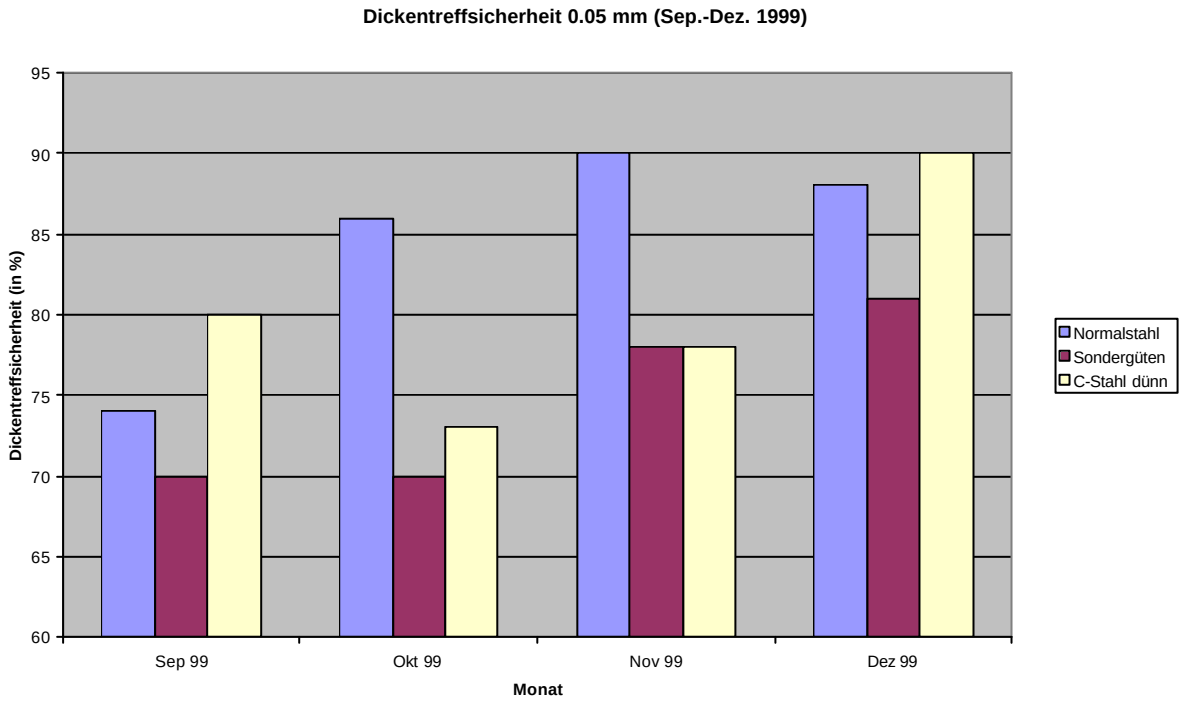


Abbildung 5.3: Dickentreffsicherheit

Im zweiten Schritt wurden zusätzliche Zentren hinzugefügt, um die Modellierung von Sondergüten zu optimieren. Dieser Optimierungsschritt ist in Abbildung 5.3 an

der deutlichen Verbesserung der Dickentreffsicherheit der Sondergüten im Zeitraum Oktober bis Dezember 1999 zu erkennen.

Da trotz dieser Optimierungen für einige spezielle Stahlsorten immer noch Probleme auftraten, wurde Anfang Dezember eine Normierung des Eingangsvektors nach (5.1) installiert. Die deutliche Verbesserung der Dickentreffsicherheit ist in Abbildung 5.3 am Beispiel der am meisten betroffenen so genannten dünnen C-Stähle dargestellt.

5.6 Folgerung

Die an der betrachteten Warmbreitbandstraße durchgeführten Experimente führen zu einer Reihe interessanter Ergebnisse, die sich sicherlich für andere Walzstraßen verallgemeinern lassen.

Die statistische Analyse der Ergebnisse des Altsystems und des Neusystems zeigt, dass die Entwicklung der Prozessautomatisierung von Walzstraßen in den letzten 20 Jahren beachtliche Fortschritte zu verzeichnen hat. Mit Hilfe moderner Prozessmodelle, die zum Teil von modernen Adaptionsmechanismen wie neuronalen Netzen unterstützt werden, ist es heute möglich, deutlich bessere Ergebnisse als noch vor 15 Jahren zu erzielen. Die Verbesserungen liegen dabei, je nach Anlagentyp und vor allem je nach gewalztem Produktspektrum, im Bereich zwischen 10% und 20%. Die Probleme des Altsystems liegen dabei weniger in Ungenauigkeiten der physikalisch-mathematischen Modelle, sondern in den verwendeten Adaptionsmechanismen. Diese können nur schwer mit veränderten Umgebungsbedingungen und Werkstoffen umgehen; sie zeigen demnach schlechte Generalisierungsleistungen.

Dies legt die Vermutung nahe, dass bei Optimierung der Adaptionsmechanismen eine deutliche Verbesserung der gesamten Prozessautomatisierung erzielt werden könnte. Genau das zeigen auch die Ergebnisse der Optimierung des Altsystems mit neuronalen Restfehlernetzen. Mit Hilfe dieser Netze lassen sich Resultate erzielen, die beinahe in der Größenordnung moderner Prozessautomatisierungen liegen. Die Tatsache, dass andererseits mit reinen neuronalen Netzen keine Verbesserung zu erzielen ist, bestätigt die Annahme, dass der Hauptschwachpunkt von alten Prozessautomatisierungen häufig in schlechten Adaptionsmechanismen liegt.

Die moderne Prozessautomatisierung, bei der bereits neuronale Netze als Adaptionsmechanismus eingesetzt werden, lässt sich demgegenüber nur noch minimal mit Restfehlernetzen verbessern. Eine Optimierung ist erst dann möglich, wenn statt globaler Restfehlernetze lokale Korrekturnetze verwendet werden. In diesem Fall sind noch einmal deutliche Verbesserungen möglich. Dies zeigt wiederum, dass die Optimierung der Modelladaption zu den wichtigsten Aufgaben beim Betrieb einer Prozessautomatisierung gehört.

6 Ausblick: Fehlerdiagnose und Data Mining

6.1 Diagnosenetze

Die Diagnose von Prozessmodell-, Messgerät- und Stellgliedfehlern gehört bei der Inbetriebnahme und der Wartung der Prozessautomatisierung zu den wichtigsten Aufgaben. Inhomogene Prozessschnittstellen führen jedoch bisher dazu, dass eine Diagnose enorm behindert wird. Mit Hilfe des in Kapitel 3 beschriebenen Verfahrens ist es jetzt theoretisch durch Harmonisierung der Prozessmodellschnittstellen möglich, einen Fehler direkt auf einzelne Modelle aufzuteilen.

Die Optimierung der Prozessautomatisierung hat andererseits gezeigt, dass die größten Verbesserungen durch eine Änderung der Adaptionsmechanismen erzielt werden können. Die in diesem Zusammenhang eingesetzten statistischen Methoden bzw. neuronalen Netze weisen auf eine grundsätzliche Herausforderung hin: Die erzielten Verbesserungen lassen sich kaum interpretieren. Konkret bedeutet dies, dass sich die Ergebnisse der adaptiven Teile der Prozessautomatisierung keinem Prozessmodell oder Messgerät zuordnen lassen.

Die in Abschnitt 5.3 beschriebenen Anstellungs- und Walzkraftnetze können die Qualität der rein analytischen bzw. der hybriden Prozessmodelle zwar nicht ganz erreichen und werden dort daher nicht weiter analysiert. Da diese Netze den Prozess nahezu genauso gut beschreiben wie komplexere Modelle, motiviert dies den Einsatz als Diagnosewerkzeug zur Überwachung des Prozesses; ein entsprechendes Verfahren wird nachfolgend beschrieben. Dabei lassen sich zwei Varianten unterscheiden, die einerseits Modellfehler und andererseits Anlagenfehler erkennen helfen.

6.1.1 Neuronales Referenznetz

Neuronale Referenznetze bestimmen mit Hilfe eines parallel zum Prozess geschalteten neuronalen Netzes neuronale Schätzwerte für beliebige Prozessparameter. Ein so genannter mobiler Agent überwacht die Ergebnisse des Online-Prozessmodells und vergleicht diese mit dem Output des Referenznetzes. Bei Überschreitung einer vorgegebenen Toleranzgrenze wird automatisch der potentielle Fehlerfall protokolliert und eine verantwortliche Stelle informiert (Abbildung 6.1). Das Training neuronaler Referenznetze geschieht offline in Produktionspausen oder direkt online, jedoch ohne Berücksichtigung des Prozessmodells. Ein neuronales Referenznetz ist demnach immer vom Prozessmodell unabhängig und eignet sich somit vor allem zur Diagnose einzelner Prozessmodelle.

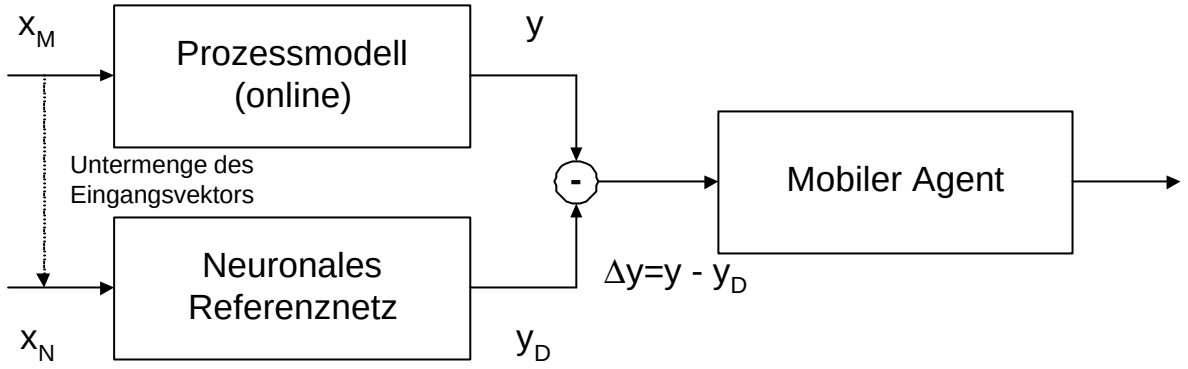


Abbildung 6.1: Neuronales Referenznetz

6.1.2 Neuronale Diagnosenetze

Im Gegensatz dazu wird bei dem in Abbildung 6.2 dargestellten neuronalen Diagnosenetz das Ergebnis der Differenz zwischen dem Ergebnis des Prozessmodells während der Stichplanberechnung und dem Output des neuronalen Netzes für die Online-Adaption des Netzes berücksichtigt. Die Differenz der Ergebnisse des Prozessmodells während der Nachberechnung mit Messwerten mit dem Output des neuronalen Diagnosenetzes lässt Rückschlüsse auf Fehler in bestimmten Messgeräten und Stellgliedern zu.

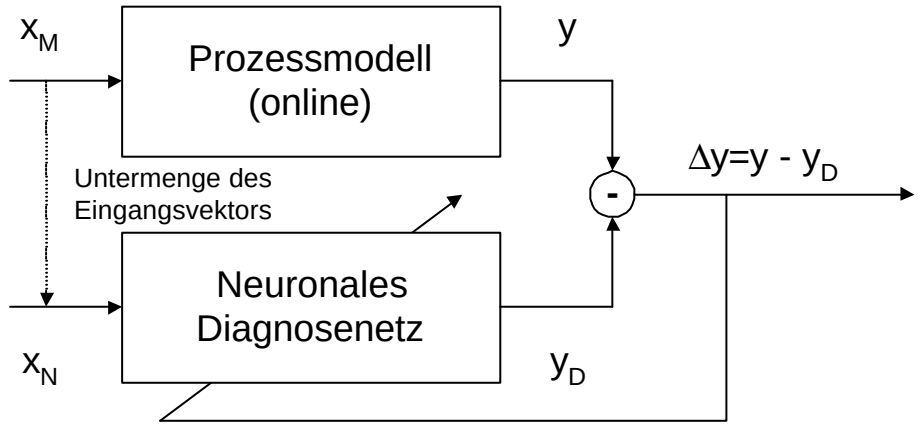


Abbildung 6.2: Neuronales Diagnosenetz

Der Einsatz von neuronalen Referenz- und Diagnosenetzen ermöglicht somit eine Identifikation von Prozessmodell-, Messgerät- und Stellgliedfehlern. Durch das unkomplizierte Verfahren zur Implementierung von neuronalen Netzen mit dem zuvor beschriebenen Framework lässt sich mit relativ geringem Aufwand ein Diagnosesystem erstellen, das Inbetriebsetzungs- und Wartungskosten minimieren hilft. Durch den Einsatz von mobilen Agenten ist zudem die Möglichkeit der Ferninbetriebsetzung und Fernwartung gegeben, mit der weitere Kosten eingespart werden können.

6.2 Data Mining

Diagnosewerkzeuge können erst dann effektiv eingesetzt werden, wenn ein einfacher, konformer Zugang zu allen Daten vorliegt - ein in der Praxis jedoch kaum anzutreffender Fall. Aktuelle Prozessrechner von Walzwerken sind dadurch gekennzeichnet, dass sie modular aufgebaut sind. Diese einzelnen Module besitzen im Allgemeinen eine eigene Datenhaltung und verwenden unterschiedlichste Typen der Datenhaltung.

Die im Framework definierte Datenstruktur für Walzwerke muss, wie in Abschnitt 3.13 beschrieben, auf die jeweilige Anlage angepasst werden. Dies geschieht am zweckmäßigsten durch Verwendung eines UML-Tools in Kombination mit einem Generator zur Erzeugung des Quellcodes. Durch die Verwendung von UML ergibt sich eine anschauliche grafische Darstellung der Datenstruktur, die unabhängig von der konkreten Implementierung ist. Durch den Generator ist es möglich, Änderungen in der Struktur sehr einfach umsetzen.

Die in Abschnitt 3.13 definierte Datenstruktur wird allerdings nicht nur auf dem Prozessrechner verwendet, denn alle Daten werden üblicherweise auch in Prozessdatenbanken abgelegt. In aktuellen Implementierungen von Prozessautomatisierungen sind diese Funktionen im Allgemeinen getrennt. Mit Hilfe des Frameworks in Kombination mit dem Generator steht jedoch ein Mechanismus zur Verfügung, eine einheitliche Datenstruktur zu schaffen.

Die ebenfalls in Kapitel 3 beschriebenen Möglichkeiten der Transformation von Datenströmen, die XML verwenden, motivieren, neue Möglichkeiten der Diagnose und Inbetriebsetzung zu untersuchen.

6.3 Erweiterte Diagnose-Infrastruktur

In Abbildung 6.3 ist die Struktur des Prozessdatenservers auf der betrachteten Anlage dargestellt. Ausgehend vom OOA-Modell der Prozessdatenstruktur werden zunächst das Datenbankschema (SQL), die eigentliche Prozessdatenstruktur als Corba-Schnittstelle (IDL) und die Prozessdatenstruktur als XML-Schema generiert. Das XML-Schema wird nachfolgend auch als Data Dictionary bezeichnet. Über die Corba-Schnittstelle können die Daten-Container-Klassen des Frameworks direkt in die Prozessdatenbank schreiben (IDL2PDB). Eine XSLT-Schnittstelle sorgt weiterhin dafür, dass direkt über XML mit der Prozessdatenbank kommuniziert werden kann (XML2PDB). Als weitere Schnittstelle steht für statistische Auswertungen eine gewöhnliche ODBC-Schnittstelle zur Verfügung.

Die geschaffene Infrastruktur ermöglicht nun drei Typen der Diagnose:

- Offline-Betrachtungen zum Gesamtprozess

- Offline-Betrachtungen zu einzelnen Prozessmodellen
- Online-Betrachtungen zum Gesamtprozess und zu einzelnen Modellen

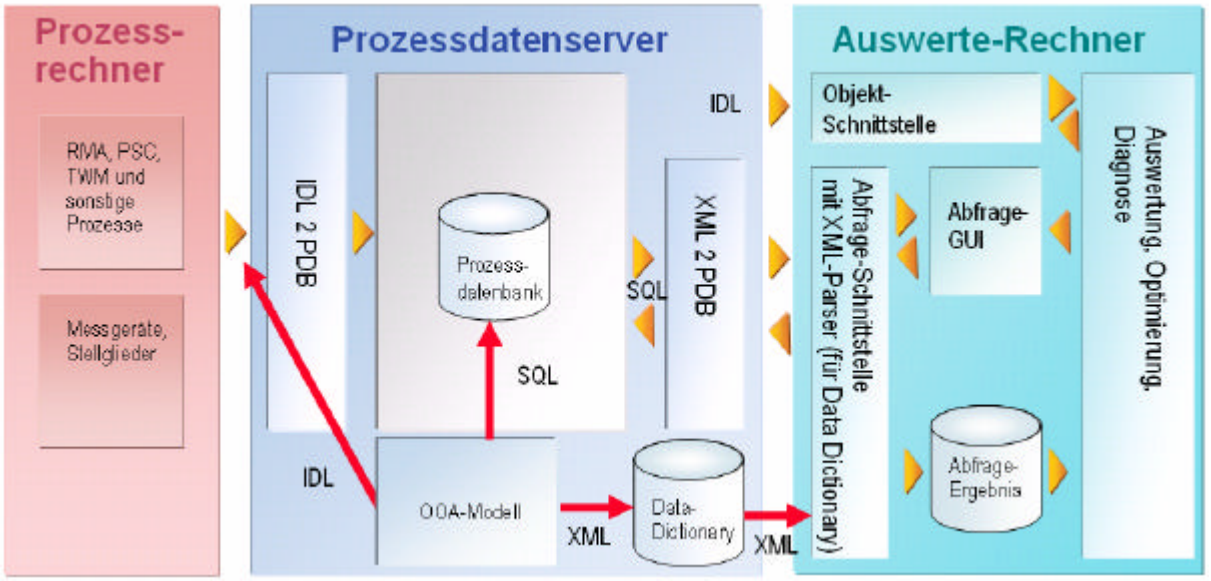


Abbildung 6.3: Der Prozessdatenserver

6.3.1 Offline-Diagnose des Gesamtprozesses

Die SQL-Schnittstelle der Prozessdatenbank kann für beliebige Auswertungen und Offline-Diagnosen verwendet werden. Diese SQL-Schnittstelle wurde auch für die deskriptive Statistik im Rahmen dieser Arbeit verwendet.

6.3.2 Offline-Diagnose einzelner Prozessmodelle

Mit Hilfe der XML-Schnittstelle ist es möglich, Modelle offline, losgelöst vom Prozess auf einem entfernten System zu betreiben und mit Hilfe der Prozessdatenbank zu speisen. Dabei lassen sich die Modelle so konfigurieren, dass die Eingaben aus der Online-Prozessdatenbank stammen und die Ergebnisse in eine Offline-Diagnosedatenbank geschrieben werden.

Diese Art der Diagnose wurde im Rahmen dieser Arbeit vor allem während des Offline-Trainings neuer neuronaler Netze verwendet. Die größte Herausforderung beim Training neuer neuronaler Netze mit veränderten Eingangsgrößen besteht darin, genügend Trainings-, Test- und Validierungsdaten zur Verfügung zu stellen. Dies stellt bei Verwendung der Prozessdatenbank kein Problem mehr dar.

6.3.3 Online-Diagnose und Ferninbetriebsetzung

Ein wesentlicher Fortschritt der beschriebenen Infrastruktur besteht darin, dass sich Diagnoserechner direkt mit dem Online-Prozess verbinden können. Dies wurde

auch in Abschnitt 4.6.5 im Rahmen der Sensitivitätsanalyse bereits beschrieben. Dadurch lassen sich neue Prozessmodelle quasi online testen, ohne sie in den Echtbetrieb einzubinden. Weiterhin lassen sich so genannte mobile Agenten installieren, die den Prozess beobachten und im Fehlerfall definierte Aufgaben ausführen können.

7 Zusammenfassung

Im Rahmen dieser Arbeit wurden Methoden zur verbesserten Modellierung des Walzprozesses entwickelt und deren Anwendung auf einer realen Walzstraße vorgestellt. Als wichtigstes Ergebnis kann dabei die signifikante Verbesserung der Dickentreffsicherheit angesehen werden.

7.1 Zusammenfassung der Methodik

Ausgehend von üblicherweise sehr heterogenen Software-Umgebungen im Umfeld der Prozessautomatisierung von Walzstraßen, wurde im Rahmen dieser Arbeit eine Methodik zum Entwurf von Prozessmodellen vorgestellt.

Der wichtigste Bestandteil dieser Methodik ist eine spezielle Klassenbibliothek, die eine Entwicklungsumgebung für nahezu beliebige Prozessmodelle bereitstellt. Durch die Integration eines Simulators für neuronale Netze lassen sich zudem durch ein standardisiertes Verfahren hybride Prozessmodelle, die sowohl analytische Algorithmen als auch neuronale Netze nutzen, implementieren.

Die Erweiterung der Klassenbibliothek für die Anwendung der Modellierung der Vorgänge im Walzspalt einer Fertigstraße beim Warmwalzen zeigt exemplarisch die Umsetzung für ein konkretes Problem. Die Besonderheit der beschriebenen Prozessmodelle liegt dabei darin, dass bekannte analytische Lösungen mit neuronalen Verfahren kombiniert wurden.

7.2 Zusammenfassung der Ergebnisse im Walzwerk

Die entwickelte Methodik wurde vor allem zur Optimierung der Dickentreffsicherheit einer Warmbreitbandstraße im Bereich der Fertigstraße angewendet; dazu wurden eine Reihe praktischer Experimente durchgeführt.

Mit Hilfe reiner neuronaler Prozessmodelle wurde zunächst der komplette Walzprozess beschrieben, um zu überprüfen, ob gänzlich auf physikalische Modelle verzichtet werden kann. Die ermittelten Netze lieferten zwar zufrieden stellende Ergebnisse waren aber – je nach Netztyp – zwischen 5% und 10 % schlechter als herkömmliche Prozessmodelle.

Eine signifikante Verbesserung der Prozessmodelle und damit der Dickentreffsicherheit konnte durch die Verwendung von neuronalen Restfehlernetzen erzielt werden. Für eine Prozessautomatisierung mit physikalischen Modellen und traditionellen Adaptionsmechanismen ergaben sich unter Verwendung eines globalen neuronalen Restfehlernetzes Verbesserungen der Dickentreffsicherheit von etwa 10 %.

Eine nochmalige Optimierung der Dickentreffsicherheit konnte erreicht werden, nachdem das globale Korrekturnetz um lokale neuronale Parameternetze ergänzt wurde, die nur Parameter einzelner Teilmodelle des Walzprozesses beeinflussen.

7.3 Relevanz der Ergebnisse und Ausblick

Die vorgestellte Klassenbibliothek zur Implementierung von Prozessmodellen im Walzwerk bietet die Möglichkeit der Trennung von analytischer Modellentwicklung und Programmierung. Dadurch ist eine schnelle Implementierung und Inbetriebsetzung möglich, die direkt durch den Entwickler des analytischen Modells erfolgen kann. Dies sorgt ebenso für eine ökonomische Effizienz wie bessere und schnellere Wartbarkeit und eine vereinfachte Erweiterbarkeit.

Die Ergebnisse der Experimente zeigen, dass sich der Walzprozess für Referenzmodelle mit Hilfe eines neuronalen Netzes darstellen lässt. Für die reale Prozessoptimierung sollte aber physikalischen Modellen der Vorzug gegeben werden; diese sollten dabei durch neuronale Adaptionsmechanismen ergänzt werden. Hierbei bieten sich sowohl Parameternetze zur Optimierung einzelner Teilmodelle, als auch Restfehlernetze zur Adaption des Gesamtprozesses an.

Weitere Anwendungsmöglichkeiten des Entwicklungsrahmens wurden in der Arbeit abschließend noch angedeutet: der Einsatz neuronaler Netze zur Diagnose von Prozessmodell-, Messgerät- und Stellgliedfehlern und die verbesserte Möglichkeit der Prozesssimulation durch konsistente Datenhaltung.

Anhang A Grundlagen der Umformtechnik

A.1 Grundbegriffe

Jeder feste Körper formt sich unter der Einwirkung äußerer Kräfte um. Während bei der elastischen Formänderung nach Beendigung der Krafteinwirkung die ursprüngliche Gestalt wieder eingenommen wird, bleibt bei plastischen Änderungen die neue Form bestehen. Dieses Verhalten wird beim Umformen ausgenutzt.

Formänderung setzt ein, wenn eine äußere Kraft so groß ist, dass die Kristalle sich auf Gleitebenen zu verschieben beginnen [Bolbrinker1989]. Eine Zunahme der plastischen Umformung bewirkt eine Behinderung des Gleitens als Resultat der Verzerrung und Zertrümmerung der Kristallstruktur. Die Veränderungen der Eigenschaften werden als *Verfestigung* bezeichnet.

Entscheidend für die Umformung von Stahl ist sein *Formänderungswiderstand*, der bei hohen Temperaturen geringer als bei Raumtemperatur ist. Der Arbeitsaufwand zur Umformung ist also bei hohen Temperaturen entsprechend niedriger; zudem bewirken hohe Temperaturen eine sofortige Rekristallisation.

Abhängig von der eingesetzten Energie zur Erwärmung des Stahls werden die Kalt- und die Warmumformung unterschieden. Bei der *Warmumformung* wird das Werkstück auf Temperaturen von 800°C bis 1280°C erwärmt; die *Kaltumformung* findet ohne Wärmezufuhr statt.

Das Walzen von Stahl ist ein schrittweises Umformen mit mehreren sich drehenden Werkzeugen, den Walzen. Ein solcher Umformschritt wird als *Walzstich* bezeichnet. Da die Walzen den Werkstoff in erster Linie auf Druck beanspruchen, zählt das Walzen zur Verfahrensgruppe *Druckumformen*.

Abhängig von der Art der Bewegungsvorgänge zwischen Walze und Werkstück lassen sich Längs-, Quer- und Schrägwalzen unterscheiden.

Beim Längswalzen treten immer Paare von Walzen auf. Die *Walzenöffnung* ergibt sich aus dem Abstand der beiden Walzen. Der *Walzspalt* stellt das Gebiet dar, in dem die Umformung erfolgt. Die Länge des Walzspaltes wird *als gedrückte Länge* l_d, die Fläche der Kontaktzone als *gedrückte Fläche* A_d bezeichnet.

Da Stahl nicht kompressibel ist, gilt das Kontinuitätsgesetz, welches aussagt, dass zu jedem Zeitpunkt durch alle Ebenen des Walzspaltes die gleichen Mengen an Stahl je Zeiteinheit fließen müssen. Dies bewirkt, dass das Walzgut während des Umformens beschleunigt wird. An der *Fließ-Scheide* stimmen Walzenumfangsgeschwindigkeit und Bandgeschwindigkeit überein. Im Bereich der *Voreilung* ist die

Geschwindigkeit etwas höher, im Bereich des *Rückstaus* ist sie deutlich geringer. Die Bandgeschwindigkeit nimmt also bei einer Walzstraße, die aus mehreren Walzgerüsten besteht, stetig zu.

Die im Folgenden beschriebene Modellierung benutzt folgende Annahmen (nach [Wiegels1976]):

- Der Walzgutquerschnitt ist rechteckig.
- Die Walzen sind gerade Kreiszylinder.
- Beide Walzen haben den gleichen Radius.
- Die Walzen sind starr (Radius r) oder elastisch verformbar. Im zweiten Fall sind sie in der Umformzone weiterhin kreisförmig (Radius r´).
- Die Walzenumfangsgeschwindigkeit und damit die Winkelgeschwindigkeit ist für beide Arbeitswalzen gleich.
- Die elastische Formänderung des Walzgutes ist vernachlässigbar.
- Bei der Umformung herrscht Konstanz des Volumens.
- Die Walzgutbreite bleibt konstant.
- Ebene Querschnitte bleiben eben, d.h., dass die Reibung gering ist. Es gibt keine Haftzone.
- Der Reibwert ist entlang der Walzbahn konstant.

A.2 Umformzone

Die *Umformzone* ist der Bereich des Walzspaltes zwischen der *Eintrittsebene* E und der *Austrittsebene* A des Walzgutes. Die Länge der Umformzone wird durch die *gedrückte Länge* l'_d angegeben. Im Folgenden wird der Begriff *Walzspalt* als Synonym zur Umformzone verwendet. Die Koordinate x kennzeichnet dabei die Koordinate entgegen der Walzrichtung. Der Nullpunkt liegt in der Austrittsebene A, entsprechend besitzt x in der Eintrittsebene E den Wert l'_d.

Der *Greifwinkel* α_0 bezeichnet den Winkel zwischen Eintritts- und Austrittsebene. Der Winkel wird dabei im Mittelpunkt der abgeplatteten Walze mit Radius r´ abgetragen. Es gilt:

$$\sin \alpha_0 = \frac{l'_d}{r'} \qquad (A.1)$$

Die *Stichabnahme* oder *Höhenänderung* Δh wird aus der Differenz der einlaufenden und der auslaufenden Höhe des Walzgutes bestimmt:

$$\Delta h = h_E - h_A > 0 \qquad (A.2)$$

Die *Walzguthöhe* im Walzspalt lässt sich als Funktion des *Walzwinkels* α oder der Ortskoordinate x angeben:

$$h(a) = h_A + 2r'(1-\cos a) \approx h_A + r' a^2 \qquad (A.3)$$

$$h(x) = h_A + 2r'\left(1-\sqrt{1-\frac{x^2}{r'^2}}\right) \approx h_A + \frac{x^2}{r'} \qquad (A.4)$$

Als *mittlere Walzguthöhe* h_m wird der Mittelwert aus einlaufender und auslaufender Höhe definiert:

$$h_m = \frac{1}{2}(h_E + h_A) \qquad (A.5)$$

Die gedrückte Länge lässt sich mit (A.4) bestimmen zu:

$$l'_d = \sqrt{r'|\Delta h| - \frac{\Delta h^2}{4}} \approx \sqrt{r'|\Delta h|} \qquad (A.6)$$

Als weitere Größe wird die *gedrückte Fläche* A'_d als Produkt aus gedrückter Länge und der Breite definiert:

$$A'_d = l'_d \cdot b \qquad (A.7)$$

Die verschiedenen Größen sind in Abbildung A.1 illustriert.

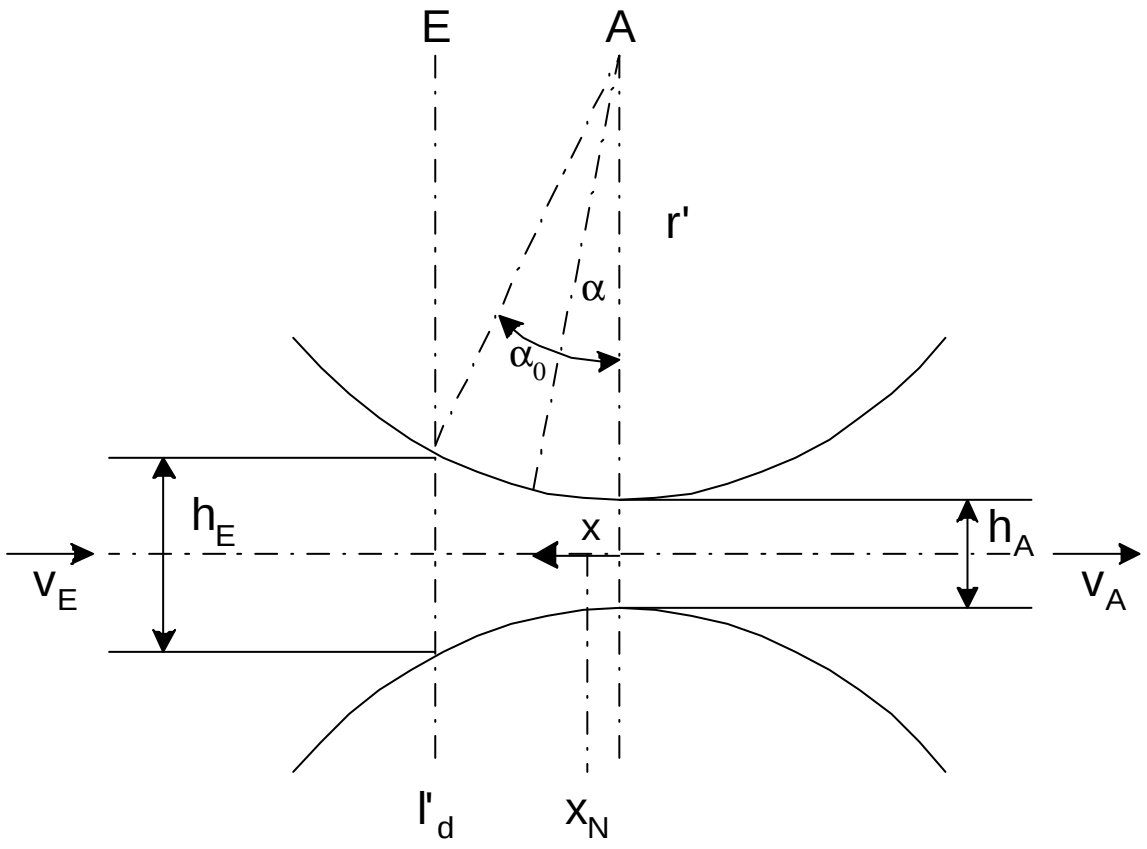


Abbildung A.1: Geometrische Größen im Walzspalt

A.3 Walzenabplattung

Nach Voraussetzung soll der durch die Walzkraft abgeplattete Berührbogen wieder kreisförmig sein. Der abgeplattete Radius der Arbeitswalzen in der Berührzone lässt sich näherungsweise mit folgender Gleichung bestimmen, die in der Literatur als „Hitchcock-Formel“ bezeichnet wird [HitTri1935]:

$$r' = r\left(1 + c_{Hitch}\frac{F_W}{b|\Delta h|}\right) \; mit \;\; c_{Hitch} = \frac{16}{p}\cdot\frac{1-n^2}{E_W} \qquad (A.8)$$

wobei F_W die Walzkraft darstellt. Die so genannte Hitchcock-Konstante c_{Hitch} kann mit Hilfe des E-Moduls der Walze bestimmt werden.

Experimentelle und theoretische Untersuchungen (z.B. [Orowan1943] und [PawSch1969]) haben gezeigt, dass diese Näherungsformel bis zu einem Verhältnis von r‘/r = 2 hinreichend genau ist, bei größeren Verhältnissen liefert (A.8) zu große Werte für den abgeplatteten Radius.

A.4 Kenngrößen der Walzspaltgeometrie

Die Walzspaltgeometrie wird von vier voneinander unabhängigen Längenmaßen eindeutig bestimmt, dies können z.B. h_E, h_A, r‘ und b sein. Für breitungsfreies Walzen reichen drei Längenmaße.

Eine eindeutige Beschreibung ist auch durch die Verwendung von mindestens zwei Längenverhältnissen möglich. Beim Flachwalzen wird üblicherweise die *relative Stichabnahme* ε und das Verhältnis aus gedrückter Länge und mittlerer Höhe l_d/h_m zur Beschreibung der Walzspaltgeometrie verwendet. Die *relative Stichabnahme* ist definiert:

$$e = \frac{h_E - h_A}{h_E} \qquad (A.9)$$

Mit Hilfe dieser Kenngrößen lässt sich die Abhängigkeit anderer Walzkenngrößen von der Walzspaltgeometrie in dreidimensionalen Diagrammen darstellen. Alternativ können auch ebene Diagramme mit Kurvenscharen für verschiedene ε verwendet werden.

Eine weitere Kenngröße für die Formänderung stellt der so genannte *Umformgrad* dar. Dieser ist definiert:

$$j = \ln\frac{h_A}{h_E} \qquad (A.10)$$

Im Gegensatz zur gesamten relativen Stichabnahme kann der *Gesamtumformgrad* einer mehrgerüstigen Straße durch Addition der Umformgrade jedes Gerüstes bestimmt werden.

Ebenfalls von Bedeutung ist die zeitliche Ableitung des Umformgrades. Diese Kenngröße wird als *Formänderungsgeschwindigkeit* definiert:

$$\dot{j} = \frac{dj}{dt} = \frac{\frac{dh}{h}}{dt} = \frac{1}{h}\frac{dh}{dt} = \frac{1}{h}\frac{dh}{dx}\frac{dx}{dt} = \frac{1}{h}\frac{dh}{dx}v(x) \qquad (A.11)$$

A.5 Kontinuitätsgleichung und Walzgutgeschwindigkeit

Aus der vorausgesetzten Volumenkonstanz im Walzspalt lassen sich die Geschwindigkeiten in der Umformzone bestimmen. Aus der Volumenkonstanz folgt zunächst ein konstanter Volumenstrom Φ:

$$\Phi = \dot{V} = v(x) \cdot A(x) = v(x) \cdot h(x) \cdot b(x) = v(x) \cdot h(x) \cdot b = konst \qquad (A.12)$$

Wenn die Breitung vernachlässigt wird (b(x) = b), ergibt sich:

$$v(x) \cdot h(x) = v_E \cdot h_E = v_A \cdot h_A \qquad (A.13)$$

Damit kann mit (A.4) der Geschwindigkeitsverlauf im Walzspalt angegeben werden:

$$v(x) = \frac{v_E}{\frac{h_A}{h_E} + \frac{x^2}{r' h_E}} = \frac{v_A}{1 + \frac{x^2}{r' h_A}} \qquad (A.14)$$

In der Realität sind v_E und v_A schwierig zu bestimmen, daher wird ein Zusammenhang zwischen der Walzenumfangsgeschwindigkeit v_u und der Bandgeschwindigkeit gesucht. Zunächst wird die so genannte *Relativgeschwindigkeit* zwischen Walzgut und Walzenoberfläche definiert:

$$v_{rel} = v(x) - v_u \cos\alpha \qquad (A.15)$$

Es werden zwei Bereiche unterschieden. In der *Nacheilzone* ist die Relativgeschwindigkeit negativ: Das Band ist langsamer als die Walzenoberfläche. Im Bereich der *Voreilzone* ist das Band schneller als die Walzenoberfläche, entsprechend ist die Relativgeschwindigkeit hier positiv. Die Grenze, an der beide Geschwindigkeiten gleich sind, wird als *Fließscheide* x_N bezeichnet.

Es lassen sich die beiden Größen *bezogene Voreilung* κ_A und *bezogene Nacheilung* κ_E definieren:

$$k_A = \frac{v_{rel}(x=0)}{v_u} = \frac{v_A - v_u}{v_u} = \frac{v_A}{v_u} - 1 > 0 \qquad (A.16)$$

$$k_E = \frac{v_{rel}(x=l_d')}{v_u} = \frac{v_E - v_u}{v_u} = \frac{v_E}{v_u} - 1 < 0 \qquad (A.17)$$

Die Lage der Fließscheide hängt im Wesentlichen von den Parametern Reibung und Längszug ab. Beim längszugfreien Walzen halten sich die Reibungskräfte in der Vor- und Nacheilzone das Gleichgewicht. Treten Zugkräfte am eintretenden Walzgut auf, wird die Walzgeschwindigkeit verringert und die Fließscheide wandert zum Austritt. Zugkräfte am austretenden Walzgut erhöhen entsprechend die Walzgeschwindigkeit, und die Fließscheide liegt näher am Eintritt. Für längszugfreies Walzen lässt sich die Fließscheidenlage bei bekanntem Reibwert μ nach der Gleichung von Hoff und Dahl bestimmen [HofDah1955]:

$$x_N = \frac{1}{2} l_d' \left(1 - \frac{l_d'}{2mr'} \right) \qquad (A.18)$$

Aus der Kontinuität und der verschwindenden Relativgeschwindigkeit an der Fließscheide ergibt sich mit (A.4) nach [Wiegels1976a]:

$$v_u \left(1 - \frac{1}{2} \frac{{x_N}^2}{r'^2} \right) \left(h_A - \frac{{x_N}^2}{r'^2} \right) = v_A h_A = v_E h_E \qquad (A.19)$$

Somit können sowohl Ein- als auch Austrittsgeschwindigkeit rechnerisch aus der Walzenumfangsgeschwindigkeit bestimmt werden. Beim Walzen mit Längszug muss zur Bestimmung der Fließscheidenlage auf ein so genanntes Voreilungsmodell zurückgegriffen werden, das eine bessere Näherung als Gleichung (A.19) verwendet, bei der die Entwicklung der Taylor-Reihe bereits beim quadratischen Summanden abgebrochen wird.

Mit Hilfe der Bandgeschwindigkeiten lässt sich eine weitere Kenngröße, die *Berührzeit* t_B, definieren - den Quotienten aus gedrückter Länge und Bandaustrittsgeschwindigkeit:

$$t_B = \frac{l_d'}{v_A} \qquad (A.20)$$

A.6 Berücksichtigung der Breitung

In den bisherigen Betrachtungen ist die Änderung der Walzgutbreite unberücksichtigt geblieben. Die Änderung der Walzgutbreite im Walzspalt, auch Breitung genannt, hat einen Einfluss auf die Fließscheidenlage, die Voreilung und die Umformgeschwindigkeit und damit auf die Kontinuitätsgleichung. Bei der Modellierung

der Vorgänge im Walzspalt muss dieser Effekt mit Hilfe eines so genannten Breitungsmodells (siehe Abschnitt 2.2.4.5) berücksichtigt werden.

Anhang B Aufbau eines Walzwerkes

Warmbreitbandwalzwerke (WBW) sind komplexe Anlagen. Zunächst werden einige Begriffe eingeführt, bevor in den nachfolgenden Unterabschnitten der Aufbau eines Warmbreitbandwalzwerkes am Beispiel eines Werkes eines großen europäischen Stahlproduzenten beschrieben wird.

B.1 Walzwerke

In einem Walzwerk sind alle Anlagen zusammengefasst, die zur Herstellung von Walzerzeugnissen benötigt werden. Abhängig von der Art der Umformung werden Warm- und Kaltwalzwerke unterschieden. Im Folgenden sollen nur Warmwalzwerke interessieren, in denen *Vorbrammen*, im Weiteren meistens kurz *Brammen* genannt, zu *Warmband* verarbeitet werden. Diese Warmbreitbandwerke bestehen aus:

- Öfen
- Vorstraße (z.B. mit einem Reversiervorgerüst)
- Coilbox (optional)
- Fertigstraße
- Bandkühlung und Haspel(n)

Zum Walzwerk gehören außerdem Lager, Transport- und Führungseinrichtungen und umfangreiche Regelungs-, Steuerungs- und Messsysteme.

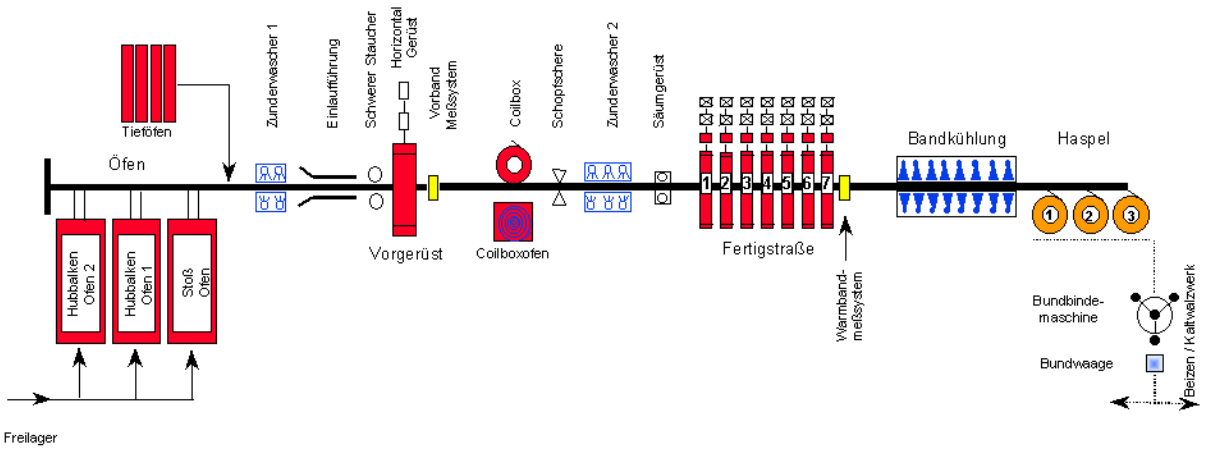


Abbildung B.1: Schema eines Warmbreitbandwerkes

B.2 Walzstraßen

Da das Fertigprodukt nur selten in einem Durchgang gewalzt werden kann, werden mehrere Walzgerüste zu einer so genannten *Walzstraße* zusammengefasst, in der

dann entsprechend der Anzahl der Gerüste mehrere *Walzstiche* durchgeführt werden.

Warmbreitbandwalzwerke besitzen im Allgemeinen zwei Walzstraßen. In der *Vorstraße* bzw. dem *Vorgerüst* wird die Bramme vorverarbeitet, um anschließend in der meist sechs- oder siebengerüstigen *Fertigstraße* ausgewalzt zu werden.

Im Walzwerk stellen die *Walzgerüste* die zentralen Anlagenteile dar. Jedes Walzgerüst besteht aus zwei *Walzenständern*, die einteilig (geschlossen) oder mit abnehmbaren Oberhaupt (offen) gebaut werden, wobei in Warmbreitbandwalzwerken überwiegend geschlossene Varianten eingesetzt werden. Bei geschlossenen Ständerformen erfolgt der Walzenwechsel durch ein seitliches Fenster. Walzenständer müssen die hohen auftretenden Walzkräfte aufnehmen können und dürfen sich dabei nur wenig dehnen.

Die Walzenlager sind höhenverstellbar in den Fenstern eingebaut. Sie sorgen für eine richtige Führung der Walzen und übertragen die Walzkräfte über das Anstellsystem auf den Walzständer. Es kommen Gleit- und Wälzlager zum Einsatz.

Die *Anstellvorrichtungen* dienen dem horizontalen und vertikalen Positionieren der Walzen. Sie können mechanisch, elektromechanisch oder hydraulisch ausgeführt sein.

Beim Walzen von Warmbreitband werden im Allgemeinen Vierwalzengerüste eingesetzt. Diese bestehen aus zwei Arbeitswalzen und zwei Stützwalzen.

Die *Walzen* eines Walzgerüstes stellen die eigentlichen Umformwerkzeuge dar. Der Teil der Walzen, der in Kontakt mit dem Werkstück tritt, wird als *Ballen* bezeichnet. Beim Walzen von Flachprodukten werden Arbeits- und Stützwalzen unterschieden. Zwischen den *Arbeitswalzen* geschieht die Umformung des Werkstückes, während die *Stützwalzen* eine Durchbiegung der Arbeitswalzen verhindern sollen und dementsprechend haben Stützwalzen meist größere Durchmesser als Arbeitswalzen. Im Allgemeinen treten Arbeits- und Stützwalzen immer paarweise auf, und es lässt sich dann eine Symmetrieebene angeben, die senkrecht zur Walzebene liegt. Zwei-, Vier- und Sechswalzengerüste unterscheiden sich nur durch die Anzahl ihrer Stützwalzen. Durch verschiedenste Störgrößen werden das Profil und die Planheit des gewalzten Werkstückes beeinflusst. Um diese Einflüsse auszuschalten, werden die Walzen mit an die Betriebsbedingungen angepassten Schliffen versehen.

Die Applikationen dieser Arbeit werden in der Warmbreitbandstraße der ThyssenKrupp Stahl AG in Bochum durchgeführt. Daher wird diese Anlage in den nächsten Abschnitten beschrieben (nach [KHS1995] und [TKS1998]).

B.3 Brammenlager

Vor der Walzung werden die Brammen im Allgemeinen zunächst eingelagert. Normalstahl-Brammen werden im *Freilager* auf einer Fläche von 4000 m² in Stapeln von maximal 14 Stücken (max. 90.000 t) und Edelstahlbrammen werden zur Zurichtung in der *Brammenscheiferei* im *Hallenlager* gestapelt (ca. 20.000 t). Die *Vorbrammen* sind zwischen 600 und 1620 mm breit, zwischen 130 und 260 mm dick, zwischen 4000 und 9600 mm lang und wiegen typischerweise etwa 20 t.

Brammen, die zu verplanten Walzprogrammen gehören, werden unter Berücksichtigung der Walzreihenfolge und Ofennummer auf dem *Vorsortierplatz* gestapelt.

B.4 Wiedererwärmung

Das Walzwerk besitzt drei erdgasbefeuerte Wiedererwärmöfen und zwar einen Stoß- und zwei Hubbalkenöfen. Beide Ofentypen sind Durchlauföfen, bei denen mehrere Brammen, in der so genannten Brammensäule, quer zur Transportrichtung liegen. Alle Öfen besitzen *Austragemaschinen*, die die wiedererwärmten Brammen auf den *Warmrollgang* heben.

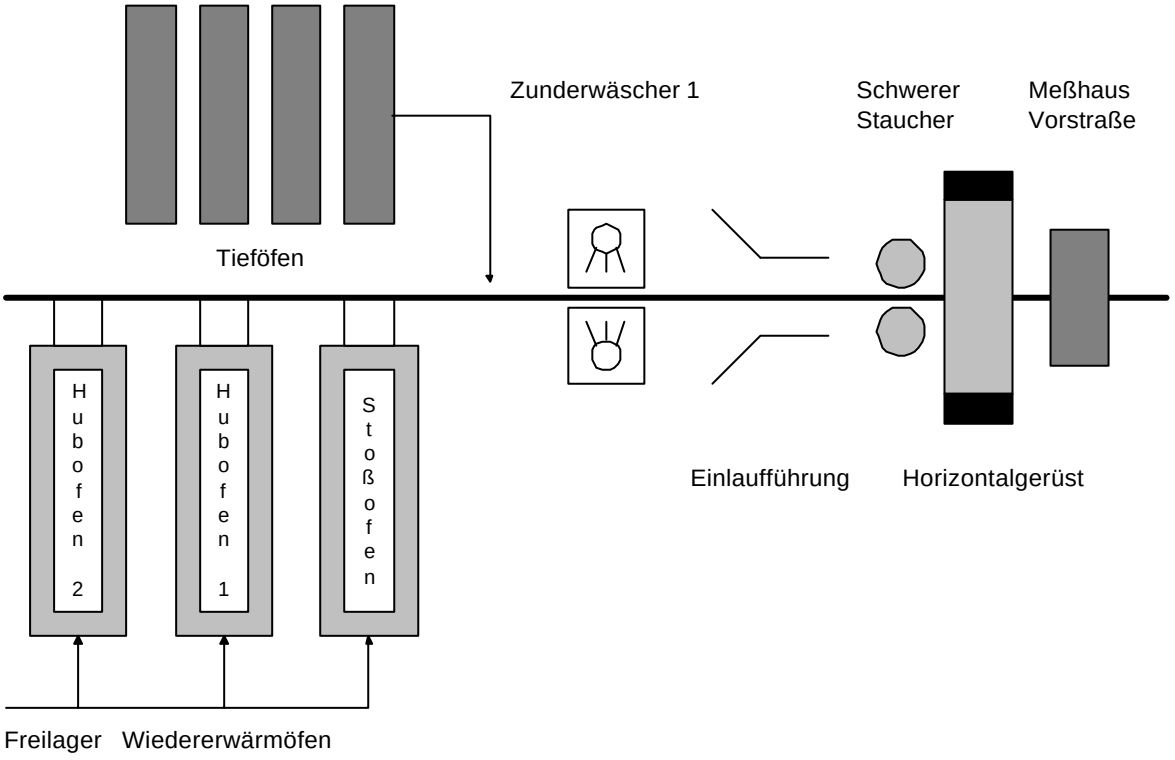


Abbildung B.2: Öfen und Vorstraße

Der *Stoßofen* nutzt einen starken *Brammendrücker*, um die Brammensäule durch den Ofen zu „stoßen“. Die Brammen bewegen sich zunächst über heißdampfgekühlte Gleitschienen und anschließend in der *Ausgleichszone* am Ende des Ofens über

einen ungekühlten Festherd. Dort verflachen die durch die Schienenkühlung entstandenen Temperaturgradienten zum größten Teil.

Die beiden *Hubbalkenöfen*, die neuerer Bauart als der Stoßofen sind, beruhen auf einem anderen Prinzip: Hier liegen die Brammen auf heißdampfgekühlten *Festbalken*. Der Transport durch den Ofen findet mit Hilfe von *Hubbalken* statt, die die Brammensäule anheben und wiederum auf Festbalken absetzen. Auch Hubbalkenöfen besitzen einen Brammendrücker, der jedoch nur dazu dient, die Brammen vom Kaltrollgang in den Zugriffsbereich der Hubbalken zu stoßen. Hubbalkenöfen erreichen, da die Brammen mit Abstand eingesetzt werden können, eine bessere Durchwärmung der Brammen als Stoßöfen und beanspruchen zudem die Brammenunterseiten deutlich weniger.

B.5 Vorstraße

Vor dem Eintritt in die Vorstraße werden die Brammen zunächst mit einem *Zunderwascher*, der mit 110 bar Presswasser arbeitet, entzundert, um anschließend mit Hilfe des *Zentrierers* mittig auf dem Rollgang ausgerichtet zu werden.

Die Vorstraße besteht aus einem Vertikalgerüst, dem *Schweren Staucher,* mit zwei senkrechten Walzen zur Beeinflussung der Breitung und einem Horizontalgerüst, das als Quarto-Gerüst ausgelegt ist.

Im Allgemeinen werden die Brammen in Abhängigkeit von der Stahlsorte und den Abmessungen der Rohbramme in fünf bis neun Stichen zu einem *Vorband* mit einer durchschnittlichen Dicke von 35 mm und einer mittleren Länge von 70 m ausgewalzt.

Nach der Walzung werden als wesentliche Qualitätsmerkmale des Vorbandes seine Dicke, seine Breite und die Temperatur gemessen.

B.6 Fertigstraße

In der Fertigstraße wird das Vorband auf die gewünschten geometrischen Formen – Dicke, Breite und Profil – ausgewalzt. Aus metallurgischen Gründen ist zudem die Auslauftemperatur hinter der Fertigstraße vorgeschrieben. Um diese konstant zu halten, sind zwei Fälle zu unterscheiden: der *Durchlaufbetrieb* und der *Coilboxbetrieb*

Vor der Fertigstraße ist eine Coilbox installiert, in der das ca. 70 m lange Vorband zu einem Coilboxring aufgewickelt wird, um dadurch die Oberflächen des Vorbandes und damit die Wärmestrahlung zu reduzieren. Bei Nutzung des Coilbox wird das Vorband unmittelbar nach dem Aufwickeln zur etwa 90 s dauernden Walzung wieder

abgewickelt. Beim so genannten Durchlaufbetrieb, bei dem die Coilbox nicht benutzt wird, verringert sich die Temperatur des Vorbandes während der ebenfalls etwa 90 s dauernden Walzung um etwa 200 K. Zur Kompensation muss zur Aufrechterhaltung der Temperatur hinter der Fertigstraße die Geschwindigkeit der Walzung erhöht werden.

Der Coilboxbetrieb wird vor allem bei sehr dünnen Bändern verwendet, bei denen eine Nachbeschleunigung die Endwalztemperatur nicht vollständig kompensieren kann. Durch den Einsatz der Coilbox lassen sich zudem Ofentemperaturen energiesparend absenken und ungleichmäßige Temperaturverteilungen im Vorband reduzieren.

Hinter der Coilbox werden die Vorbänder über Seitenführungen zentriert, und in der *Schopfschere* werden der Kopf und das Ende der Vorbänder, die eine Fischmaulform besitzen und die in der Fertigstraße nicht ausgewalzt werden können, abgeschnitten. Die Schopfschere kann zudem dazu verwendet werden, das Vorband in mehrere kleinere Stücke zu unterteilen.

Hinter einem weiteren Zunderwäscher, der mit 110 bar Wasserdruck arbeitet, sorgt ein vertikales *Säumgerüst* für einen zentrierten Einlauf des Bandes in die Fertigstraße.

Die Fertigstraße besteht aus sieben horizontalen Quarto-Gerüsten F1 bis F7. Zur Einstellung des Walzspaltes besitzen die ersten vier Gerüste eine elektromechanische und die letzten drei eine hydraulische Anstellung, die alle unter Last verfahren werden können. Zwischen den Walzgerüsten befinden sich sechs *Schlingenheber*, die den Bandvorrat zwischen den Gerüsten regeln. Zur Entzunderung besitzen die ersten drei Gerüste zusätzlich Spritzdüsen, die mit 55 bar Wasserdruck arbeiten.

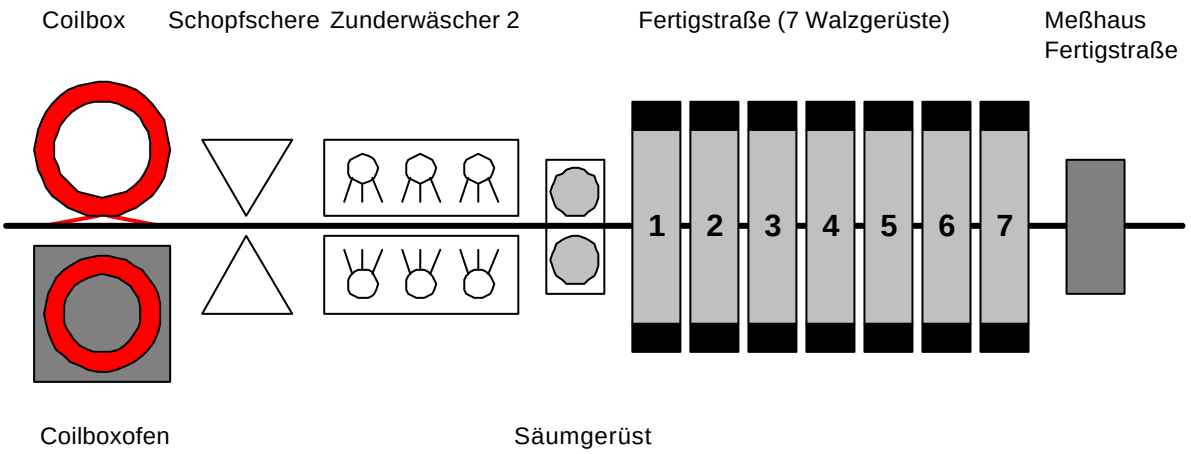


Abbildung B.3: Coilbox und Fertigstraße

Im Einlauf der Fertigstraße wird im Bereich der Schopfschere und der Coilbox jeweils die Temperatur und hinter der Fertigstraße werden in einem speziellen Messhaus die Dicke, das Dickenprofil, die Temperatur und die Temperatur gemessen.

B.7 Kühlstrecke

In der *Kühlstrecke* wird das Band unter Berücksichtigung eines gewünschten Temperaturverlaufes, mit dem die Materialeigenschaften eingestellt werden, abgekühlt. Die Kühlstrecke ist etwa 150 m lang und besteht aus drei Abschnitten: Die *laminare Zone* wird aus je 12 über- und unterhalb des Rollgangs angebrachten Laminargruppen gebildet, die *Spritzwasserkühlung* besteht aus 16 Gruppen oberhalb und 8 Gruppen unterhalb des Bandes, und die *Trimmstrecke* besitzt 8 Gruppen einzeln ansteuerbarer unterseitiger Laminargruppen.

B.8 Haspel und Coillager

Das aus der Kühlstrecke auslaufende Band wird mit Hilfe von Seitenführungen zentriert und in eine der drei so genannten *Haspeln* geleitet. Dort werden Wickelrollen direkt nach dem Einlauf des Bandkopfes auf den Wickeldorn geschwenkt. Der erste Haspel arbeitet weitgehend pneumatisch und besitzt vier Wickelrollen, die anderen beiden Haspeln sind hydraulische Haspel mit drei Wickelrollen.

Die aufgewickelten *Bunde* (Coils) werden mit einem Kippstuhl auf ein Förderband gelegt und dann inspiziert, fixiert, mit einem Stahlband verschweißt, gewogen und anschließend ins *Bundlager* (Coillager) eingelagert bzw. in das Lager des angrenzenden Kaltwalzwerkes transportiert, dort kühlen die Bunde vor dem Versand oder der Weiterverarbeitung ab. Die Lagerkapazität liegt bei 65.000 t bzw. 3.000 Coils.

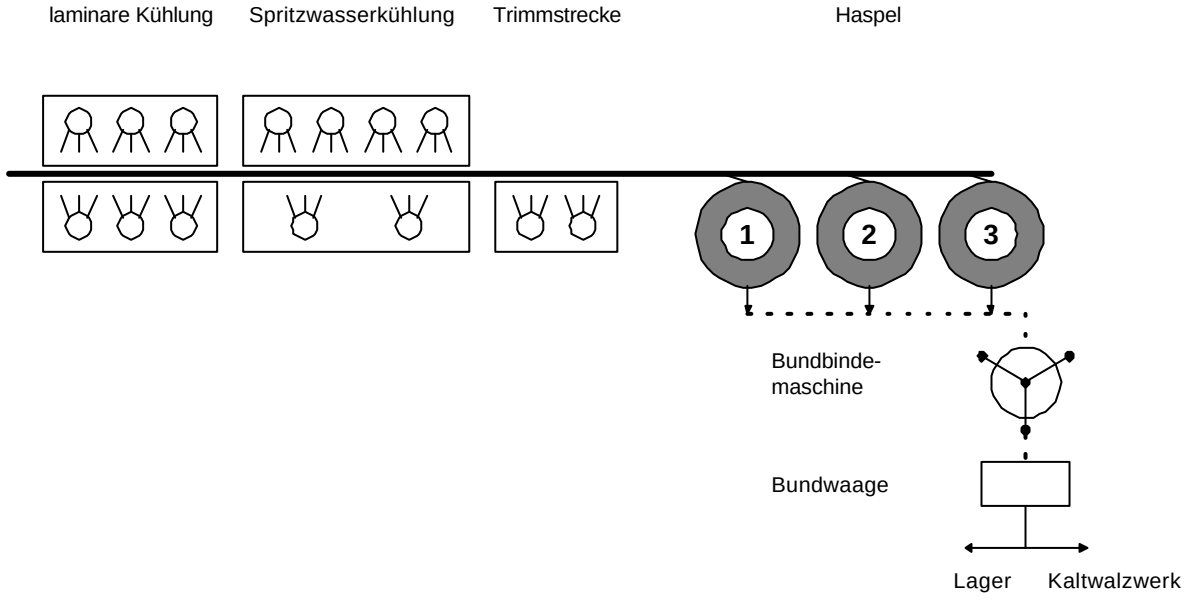


Abbildung B.4: Kühlstrecke und Haspel

Das betrachtete Warmbreitbandwalzwerk, das 1966 in Betrieb genommen wurde, erzeugt durchschnittlich 300.000 t Warmbreitband pro Monat; dabei werden neben Normalstählen (weiche unlegierte Güten, allgemeine Baustähle, microlegierte Stähle) auch eine große Anzahl von C-Stählen (legierte und unlegierte Einsatz-Vergütungs-Werkzeugstähle) und RSH-Stählen (rost-, säure- und hitzebeständige Güten) verarbeitet. Es lassen sich Endwalzendicken zwischen 1,2 und 20 mm bei einer Bandbreite von 600 bis 1630 mm erreichen; für dünne Bänder bedeutet das eine Bandlänge von über 1 km.

Anhang C Grundlagen der Softwaretechnik

C.1 UML – Unified Modelling Language

Die Unified Modelling Language – oder kurz UML – ist der Standard zur Spezifizierung, Visualisierung, Konstruktion und Dokumentation von komplexen Softwaresystemen. UML stellt dabei verschiedene Diagramme bereit, mit denen sich das betrachtete System modellieren lässt.

Der Mittelpunkt der Notation stellt die grafische Visualisierung von Klassen dar, daher werden nachfolgend nur die als Klassen-Diagramme bezeichneten Elemente der Unified Modelling Language (*UML*) beschrieben. Klassendiagramme nutzen folgende Grundkonzepte (nach [Balzert1996]):

- Ein *Objekt* ist ein Gegenstand des Interesses, insbesondere einer Beobachtung, Untersuchung oder Messung. Ein Objekt besitzt definierte Eigenschaften und reagiert mit einem festgelegten Verhalten mit seiner Umgebung. Jedes Objekt ist einzigartig, es besitzt eine *Objekt-Identität*. Die Begriffe *Instanz* und *Exemplar* werden in dieser Arbeit als Synonym für den Begriff des Objektes verwendet.

- Eine *Klasse* kennzeichnet eine Menge von Objekten mit gemeinsamen Eigenschaften und gemeinsamen Verhalten. Eine Klasse, die einen Mechanismus besitzt, um Objekte zu erzeugen, wird als *konkrete Klasse* bezeichnet. Eine Klasse, von der keine Objekte erzeugt werden können, heißt *abstrakte Klasse*.

- Die *Attribute* einer Klasse beschreiben die Eigenschaften der Klasse. Jedes Attribut besitzt einen bestimmten Datentyp, der festlegt, welche Art von Daten ein Attribut aufnehmen kann.

- Eine *Operation* ist eine ausführbare Tätigkeit im Sinne eines Algorithmus. Operationen legen das Verhalten eines Objektes fest. Einen Spezialfall stellen *Klassenoperationen* dar, die sich auf mehrere oder alle Objekte einer Klasse auswirken. *Implizite Operationen* sind solche Operationen, die nahezu jede Klasse besitzt und die daher nicht mit in das Klassendiagramm aufgenommen werden.

- Eine *Botschaft* ist die Aufforderung eines Clients an einen Server, eine Dienstleistung zu erbringen. Der Server interpretiert die Botschaft und führt eine Operation aus. Die Menge aller Botschaften einer Klasse wird *Protokoll* genannt.

- *Vererbung* kennzeichnet die Fähigkeit einer Klasse über die Eigenschaften und das Verhalten einer anderen Klasse zu verfügen. In einer Vererbungsstruktur heißen Klassen, von denen geerbt wird, *Oberklassen*. Erbende Klassen nennt man dementsprechend *Unterklassen* oder *Basisklassen*. Im Gegensatz zur *Einfachvererbung*, bei der eine Unterklasse nur eine Oberklasse besitzt, existieren bei der *Mehrfachvererbung* mehrere Oberklassen. Ein wichtiges Mittel der Vererbung stellt die Redefinition von Operationen dar. Dies tritt ein, wenn eine Unterklasse die gleiche Methode wie ihre Basisklasse definiert. Als Synonym für die Vererbung wird auch häufig von Generalisierung und Spezialisierung gesprochen.

- *Polymorphismus* ist eines der wichtigsten Konzepte der objektorientierten Software-Entwicklung. Unter diesem Begriff wird die Fähigkeit verschiedener Empfängerobjekte unterschiedlicher Klassen verstanden, auf dieselbe Botschaft auf eine jeweils klassentypische Art zu reagieren. Dies bedeutet, dass ein Client nicht wissen muss, zu welcher Klasse ein Empfäger-Objekt gehört.

Die objektorientierte Analyse (OOA) erweitert die Grundkonzepte um folgende Elemente:

- Eine *Assoziation* modelliert Beziehungen zwischen Objekten gleichrangiger Klassen. Es sind auch Assoziationen zwischen Objekten derselben Klasse möglich; diese werden als *rekursive Assoziationen* bezeichnet. Die *Kardinalität* einer Beziehung gibt die Anzahl der Objekte an, mit denen ein Objekt in einer Beziehung stehen kann oder muss. Synonym zum Begriff Kardinalität wird häufig der Begriff *Multiplizität* verwendet. Eine *Rolle* definiert die Funktion eines Objektes in einer Beziehung.

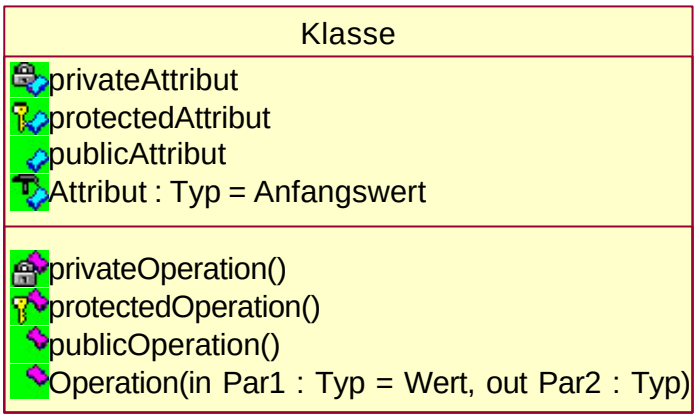
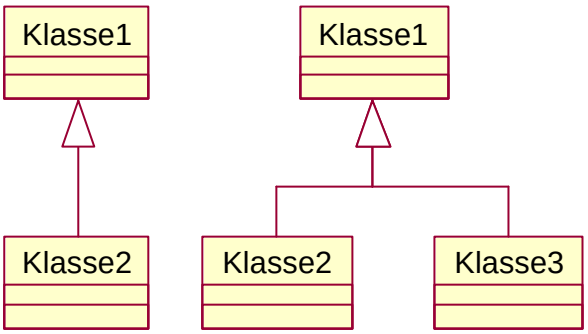
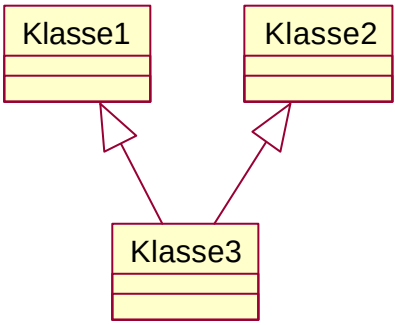
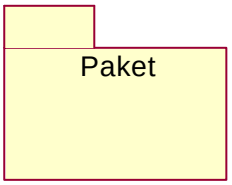
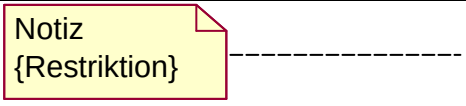
- Eine *Aggregation* ist eine besondere Form der Assoziation. Sie beschreibt Beziehungen, bei denen sich Objekte einer Klasse aus Objekten einer anderen Klasse zusammensetzen. Eine Aggregation definiert somit eine Rangfolge von Objekten.

- Eine *Komposition* ist ein in der Realität häufig vorkommender Spezialfall der Aggregation. Die Kardinalität auf Seiten des Aggregats ist hier immer gleich eins. Außerdem können Teilobjekte nicht ohne das Aggregat existieren.

- Mit *Paketen* lassen sich Problemstellungen gliedern. Jedes Paket stellt dabei ein konsistentes Teilmodell dar.

- *Abhängigkeiten* sind eine Form von Beziehungen von Elementen eines objektorientierten Modells. Die Abhängigkeit beschreibt die Änderung, die die Änderung eines unabhängigen Elementes auf ein abhängiges Element nach sich ziehen kann.

In Tabelle C.1 sind die wichtigsten Elemente eines Klassendiagramms noch einmal zusammengestellt.

Klasse, Attribut, Operation	
Einfachvererbung	
Mehrfachvererbung	
Paket	
Notiz, Restriktion	

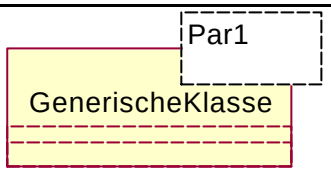
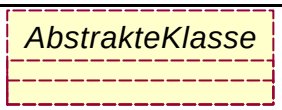
Generische Klasse	
Abstrakte Klasse	
Assoziation	
Aggregation	
Komposition	

Tabelle C.1: Elemente eines Klassendiagramms

C.2 XML

C.2.1 Allgemeines

Im Jahr 1996 begann das World Wide Web Consortium (W3C) damit, eine neue Standardauszeichnungssprache zu entwickeln, die deutlich einfacher als der bisherige Standard SGML und gleichzeitig deutlich strukturierter als HTML sein sollte. Das Ergebnis stellt XML dar; XML besitzt folgende wesentliche Eigenschaften:

- XML unterstützt ein großes Spektrum an Anwendungen.
- XML ist kompatibel zu SGML.
- XML lässt sich leicht über das Internet nutzen.
- Die Verarbeitung von XML-Dokumenten ist relativ leicht zu implementieren.
- XML-Dokumente sind für den Menschen lesbar und interpretierbar.
- Das Design von XML ist formal, kurz und prägnant.

Ein XML-Dokument besteht aus Elementen, Attributen und Kommentaren. Die Regeln, die die Gültigkeit eines Dokumentes festlegen, können in Form einer DTD oder eines XML-Schemas definiert werden. XML-Dokumente müssen außerdem *wohlgeformt* sein, wobei wohlgeformt informell bedeutet, dass ein Dokument ein oder mehrere Elemente enthalten muss, die von einem Wurzelobjekt eingebettet werden, und dass eingebettete Elemente vollständig eingeschlossen werden.

C.2.2 DTD und XML-Schema

Die DTD (Document Type Definition) ist die ursprüngliche Version zur Definition von Regeln eines XML-Dokumentes. Mit ihr lassen sich in einfacher Weise Elemente und Attribute definieren. Allerdings lässt eine DTD keine Gültigkeitsüberprüfung von Datentypen und eine Überprüfung von Wertebereichen zu. Zudem ist eine DTD ein Dokument, das selbst kein gültiges, wohlgeformtes XML-Dokument ist.

Wenn datenorientierte XML-Dokumente betrachtet werden, ist eine Gültigkeitsüberprüfung von Datentypen und eine Überprüfung von Wertebereichen häufig erforderlich. Daher werden Schemata verwendet, mit denen dieses Manko von DTDs gelöst wird. XML-Schemata sind Regeln, die selbst in XML geschrieben sind. Sie bieten neben den Möglichkeiten von DTDs auch die Zuordnung von Datentypen, die Verwendung von Namespaces und die Definition von Wertebereichen. Die wichtigsten Elemente eines XML-Schemas sind:

- schema: Das *schema*-Element ist das Stammelement. Alle *element*-Elemente eines Schemas sind einem *schema*-Element untergeordnet.
- simpleType: Ein *simpleType*-Element repräsentiert einen einfachen Datentypen und kann entweder geordnet (*ordered*) oder ungeordnet (*unordered*) sein. Für elementare Datentypen hat das W3C bereits eine Reihe simpleTypes definiert.
- complexType: Ein *complexType*-Element definiert einen komplexen Datentypen, der aus untergeordneten Elementen und Attributen bestehen kann.
- element: Das *element*-Element definiert ein Element, das seine Ausprägung im Wesentlichen durch die Attribute *name* und *type* erhält, wobei der Typ jeder im Schema definierte *simpleType* oder *complexType* sein kann.

C.2.3 XSLT – XML-Transformation

Ein großer Vorteil von XML besteht darin, dass es einen Mechanismus zum Weiterverarbeiten gibt: XSLT (Extensible Stylesheet Language Transformations). XSLT ist Teil einer größeren Spezifikation: der XSL oder Extensible Stylesheet Language, die sich vor allem mit dem formatierten Darstellen von Dokumenten auseinandersetzt. Der reine Formatierungsbereich der Spezifikation, der auf so genannten Formatierungsobjekten basiert, wird auch als XSL-FO bezeichnet. XSLT ist als Mechanismus zur Transformation von XML-Dateien in XSL-FO-basierende Daten entworfen worden.

XSLT ist eine Sprache zur Transformation von XML-Dokumenten in andere textbasierte Dokumente; dies können wiederum XML-Dokumente oder auch formartierte Textdateien sein. Im Rahmen dieser Arbeit wird XSLT zur Transformation von XML-

Dokumenten eines Schemas zu XML-Dokumenten eines anderen Schemas verwendet.

Aus der Sicht von XSLT handelt es sich bei XML-Dokumenten um Bäume, die aus Knoten bestehen. Insgesamt unterscheidet XSLT sieben Knotenarten:

- Wurzelknoten: Er stellt den gesamten Baum und damit das gesamte XML-Dokument dar.
- Elementknoten: Er besteht aus dem Teil, der im XML-Dokument durch ein Start- und ein passendes Ende-Tag begrenzt wird.
- Attributknoten: Er enthält den Inhalt eines Attributes.
- Namensraumknoten: Es stellt eine Namensraumdeklaration dar.
- Verarbeitungsanweisungsknoten: Er enthält den Text einer Verarbeitungsanweisung.
- Textknoten: Er enthält Zeichensequenzen.
- Kommentarknoten: Er enthält den Inhalt eines Kommentars.

Mit Hilfe von XSLT kann aus einem beliebigen Eingangsbaum ein beliebiger Ausgangsbaum erzeugt werden. Dabei bietet XSLT sehr komplexe Möglichkeiten zur Transformation von Quellknoten auf Zielknoten. Ein wichtiger Teil von XSLT sind so genannte Suchmuster, mit denen Teile des Eingangsbaums selektiert werden können und die mit Hilfe der im nächsten Abschnitt beschriebenen Sprache XPath definiert werden.

Im Rahmen dieser Arbeit werden nur XML-zu-XML-Transformationen betrachtet.

C.2.4 XPath

XPath ist eine Sprache zum Referenzieren einzelner Teile eines XML-Dokumentes. Das W3C definiert die Aufgabe von XPath wie folgt (Übersetzung aus [Holzner2002]):

„Der Hauptzweck von XPath ist es, Teile eines XML-Dokuments zu adressieren. Als Unterstützung für diesen Hauptzweck bietet es außerdem elementare Möglichkeiten für die Bearbeitung von Zeichenketten, Zahlen und Booleschen Werten. XPath verwendet eine kompakte, nicht XML-basierte Syntax, um die Verwendung von URIs und XML-Attributwerten zu vereinfachen. XPath arbeitet mit der abstrakten, logischen Struktur eines XML-Dokuments, nicht mit seiner oberflächlichen Syntax. XPath bekam seinen Namen aufgrund der Verwendung von Pfad-Notationen für die

Navigation durch die hierarchische Struktur eines XML-Dokuments, ähnlich wie in URLs.“

Im Rahmen dieser Arbeit wird XPath einerseits zur Definition von Suchmustern bei der Transformation von XML-Dokumenten und andererseits als Abfragesprache verwendet.

Anhang D Grundlagen neuronaler Netze

D.1 Einleitung

D.1.1 Definition

Neuronale Netze (NN), oft auch als künstliche neuronale Netze (KNN) bezeichnet, sind informationsverarbeitende Systeme, die aus einer großen Anzahl einfacher Einheiten (Neuronen) bestehen, die sich Information in Form der Aktivierung der Neuronen über gerichtete Verbindungungen zusenden [Zell1994].

D.1.2 Eigenschaften

Neuronale Netze besitzen gegenüber herkömmlichen Ansätzen folgende positive Eigenschaften:

- Lernfähigkeit
- Parallelität
- Verteilte Wissensrepräsentation
- Hohe Fehlertoleranz
- Assoziative Speicherung von Information
- Robustheit gegen Störungen
- Spontane Generalisierung

Im Vergleich mit anderen Methoden stehen dem folgende negative Eigenschaften entgegen:

- Wissenserwerb ist nur durch Lernen möglich
- Keine Analyse des eigenen Wisssens
- Logisches Schließen ist schwer
- Lernen ist langsam

D.1.3 Künstliche neuronale Netze

Das biologische Vorbild nachahmend, versucht man durch Zusammenschaltung mehrerer künstlicher Neuronen das Gehirn nachzubilden. Diese Neuronen besitzen mehrere Eingänge, einen „Zellkern“ und einen Ausgang. Im Zellkern wird eine gewichtete Summe gebildet, die von einer i.a. nichtlinearen Aktivierungsfunktion bewertet und an den Ausgang geschickt wird.

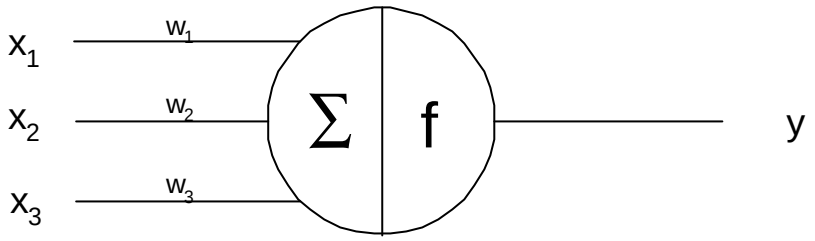


Abbildung D.1: Funktionsweise eines künstlichen Neurons

Ein solches Netz wird nicht wie herkömmliche Rechner programmiert, sondern „trainiert“. Es lernt, indem es seine Netzgewichte ändert, anhand von Übungsbeispielen ähnlich wie das menschliche Gehirn.

D.1.4 Einsatz neuronaler Netze

Es gibt zwei grundlegende Aufgaben, die mit Hilfe neuronaler Netze gelöst werden können. Dies sind die Mustererkennung und die Funktionsapproximation.

Ziel der *Mustererkennung* ist es, verrauschten Eingangsgrößen bestimmte Objekte zuzuordnen. Beispiele für solche Klassifikationsprobleme sind:

- Bildverarbeitung
- Objekt-Identifizierung
- Qualitätskontrolle (Klangprüfung)
- Kundenklassifikation

Im Gegensatz dazu wird bei der *Funktionsapproximation* versucht, aus Beispieldaten einen Zusammenhang zu lernen, der später zur Berechnung neuer Werte verwendet wird. Der Vorteil dieser Methode gegenüber konventionellen statistischen Verfahren besteht darin, dass man keine Kenntnis über die zu lernenden Funktionen benötigt und dass diese Zusammenhänge nichtlinear sein können. Außerdem können die Daten beliebig verteilt sein; es gibt keine Normalverteilungsvoraussetzungen, wie sie z.B. die multiple lineare Regression fordert. Wird ein solches Netz in der Steuerung oder Regelung eingesetzt, dient es dort meist als adaptives empirisches Modell. Es wird also ausgenutzt, dass sich neuronale Netze online weitertrainieren lassen. Beispiele für Funktionsapproximationen sind:

- Prozesssteuerung und –regelung
- Robotersteuerung
- Datenfilterung
- Finanzdatenprognose
- Wetterprognose

Im Rahmen dieser Arbeit interessieren vor allem Approximationsnetzwerke.

D.2 Komponenten neuronaler Modelle

Im vorigen Abschnitt wurden bereits erste Eigenschaften neuronaler Netze beschrieben. Bevor sie in den nächsten Abschnitten eingehend diskutiert werden, wird zunächst der Aufbau eines neuronalen Netzes dargestellt.

Neuronale Netze bestehen aus folgenden Komponenten (nach [Zell1994]):

- Neuronen
- Verbindungsnetzwerk
- Propagierungsregel
- Lernregel

D.2.1 Neuronen

Neuronen sind die kleinsten Elemente in einem neuronalen Netz. Sie sind vor allem durch ihren Aktivierungszustand, ihre Aktivierungsfunktion und ihre Ausgabefunktion gekennzeichnet.

Der *Aktivierungszustand* eines Neurons i $a_i(t)$ gibt den Grad seiner Aktivierung zum Zeitpunkt t an. Die *Aktivierungsfunktion* f_{act} beschreibt, wie sich ein neuer Aktivierungszustand $a_i(t+1)$ aus der alten Aktivierung, einem Schwellenwert $\theta_j(t)$ und der Neuroneneingabe $x_j(t)$ ergibt. Die *Ausgabefunktion* f_{out} berechnet die Ausgabe y_j eines Neurons j aus der Aktivierung:

$$y_j(t)=f_{out}(a_j(t)) \qquad \text{(D.1)}$$

Neuronale Netze verwenden zur Berechnung der Netzausgabe im Allgemeinen nichtlineare Funktionen; daher ist oft entweder die Aktivierungs- oder die Ausgabefunktion nichtlinear. Da als Ausgabefunktion meistens die Identität verwendet wird, werden in den wenigsten Netzen Aktivierungs- und Ausgabefunktion getrennt betrachtet.

D.2.2 Arten von Verbindungsnetzwerken

Das Verbindungsnetzwerk zwischen den Neuronen eines neuronalen Netzes wird im Allgemeinen in einer Matrixschreibweise angegeben, dabei stellt w_{ij} das Gewicht zwischen den Neuronen i und j dar. Ein negatives Gewicht gibt an, dass das Neuron i seinen Nachfolger j hemmt, ein positives Gewicht hingegen sorgt dafür, dass Neuron j von seinem Vorgänger i angeregt wird. Wenn keine Verbindung zwischen zwei Neuronen besteht, ist der Betrag von w_{ij} Null.

Abhängig von der Besetzung der Matrix, lassen sich Netze ohne Rückkopplung (*feedforward-Netze*) und Netze mit Rückkopplung unterscheiden.

Bei den Netzen ohne Rückkopplung existiert kein Pfad, der von einem Neuron direkt oder indirekt zu einem Vorgänger zurückführt. Es lassen sich zwei Typen unterscheiden:

- Ebenenweise verbundene feedforward-Netze sind in mehrere Schichten (*layer*) aufgeteilt, es existieren nur Verbindungen von einer Schicht zur nächsten.
- Allgemeine feedforward-Netze dürfen neben Verbindungen zwischen aufeinanderfolgenden Schichten auch solche besitzen, die einzelne Ebenen überspringen (*shortcut connections*).

Netze mit Rückkopplung lassen sich in Netze mit direkter Rückkopplung (*direct feedback*), mit indirekter Rückkkopplung (*indirect feedback*) und mit Rückkopplungen innerhalb einer Schicht (*lateral feedback*) unterteilen. Zu dieser Gruppe zählen auch die vollständig verbundenen Netze.

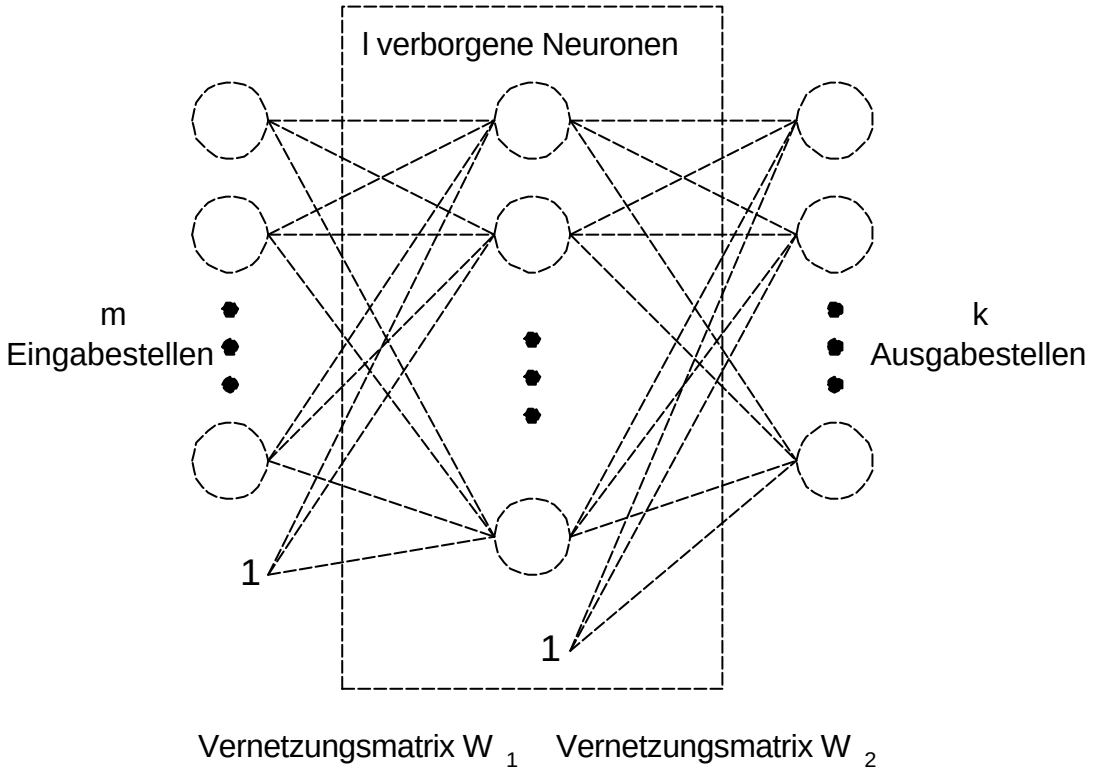


Abbildung D.2: feedforward-Netz mit 2 Verbindungsschichten

In dieser Arbeit werden vor allem einfache feedforward-Netze verwendet. Innerhalb eines solchen Netzes werden die Neuronen, die die Eingabe in das Netz leiten, als Eingabeneuronen (*input unit*) bezeichnet. Neuronen der Ausgabeschicht, so genannte Ausgabeneuronen (*output unit*), geben die Ausgabe des Netzes nach außen. Die Neuronen, die nur zur Verarbeitung vorhanden sind, sind für einen außenstehenden Betrachter nicht zu sehen und heißen daher verdeckte Neuronen (*hidden unit*). Entsprechend bezeichnet man die einzelnen Ebenen des Netzes als Eingabe-

schicht (*input layer*), Ausgabeschicht (*output layer*) und verdeckte Schicht (*hidden layer*). Abhängig von der Anzahl der einstellbaren Verbindungsschichten, wird ein Netz als n-stufig bezeichnet. Das Netz aus Abbildung D. ist dementsprechend zweistufig.

D.2.3 Propagierungsregel

Die Propagierungsfunktion gibt an, wie sich der Eingang eines Neurons aus den Ausgängen der anderen Neuronen und den Verbindungsgewichten berechnet. Im Allgemeinen berechnet sie sich zu:

$$x_j(t) = \sum_{i=1}^{n} y_i(t) w_{ij} = y(t) \cdot w_j(t) \qquad \text{(D.3)}$$

also aus der Summe der gewichteten Ausgaben der Vorgängerneuronen.

D.2.4 Lernregel

Die Lernregel ist ein Algorithmus, mit dem das Netz lernt, zu einer vorgegeben enEingabe eine bestimmte Ausgabe zu erzeugen.

Theoretisch bieten sich verschiedene Möglichkeiten des Lernens (nach [Zell1994]):

- Entwicklung neuer Verbindungen
- Löschen existierender Verbindungen
- Modifikation der Stärke w_{ij} von Verbindungen
- Modifikation des Schwellenwertes von Neuronen
- Modifikation der Aktivierungs-, Propagierungs- oder Ausgabefunktion
- Entwicklung neuer Neuronen
- Löschen von Neuronen

Dabei stellt die Alternative 3 die mit Abstand am häufigsten verwendete Form dar, die Alternativen 1, 2 und 4 lassen sich zudem auf diese Methode zurückführen. Verfahren, die versuchen, eine optimale Netztopologie zu erzielen, sind noch nicht sehr verbreitet, werden aber vermutlich in Zukunft an Bedeutung gewinnen.

Das Lernen stellt also zumeist eine Methode zur Modifikation der Verbindungsgewichte dar, wobei drei Typen des Lernens unterschieden werden:

- Beim *überwachten Lernen* (*supervised learning*) steht dem Netz beim Training neben dem Eingabemuster auch ein vollständiges Ausgabemuster zur Verfügung.

- Beim *bestärkenden Lernen* (*reinforcement learning*) ist nur bekannt, ob ein Ergebnis richtig klassifiziert wurde oder nicht.

- *Unüberwachtes Lernen* (*unsupervised learning*) benötigt keine Ausgabemuster. Es wird versucht, ähnliche Eingaben in ähnliche Kategorien (*cluster*) zu klassifizieren.

Die grundlegende Lernregel ist das *Hebbsche Lernen*, das in seiner ersten Form 1949 von Donald O. Hebb formuliert wurde [Hebb1949]. Sie besagt, dass wenn ein Neuron j von einem Neuron i eine Eingabe erhält und beide gleichzeitig stark aktiviert sind, dann das Verbindungsgewicht w_{ij} ebenfalls zu erhöhen ist. Mathematisch ergibt sich für die Änderung des Gewichts:

$$\Delta w_{ij} = h\, y_i\, a_j \qquad \text{(D.4)}$$

Dabei stellt η eine Konstante dar, die sogenannte Lernrate. Alle im Folgenden besprochenen Lernregeln lassen sich auf die Hebbsche Lernregel zurückführen.

Ein einfaches Lernverfahren für einstufige Netze mit linearer Aktivierungsfunktion ist die Delta-Regel. Die Änderung der Gewichte ist hier proportional zur Differenz δ_j der aktuellen Aktivierung a_j und der erwarteten Aktivierung t_j (*teaching input*):

$$\Delta w_{ij} = h y_i (t_j - a_j) = h\, y_i d_j \qquad \text{(D.5)}$$

Aufgrund der linearen Aktivierungsfunktion lässt sich δ_j auch als Differenz der beobachteten Ausgabe y_j und der erwarteten Ausgabe t_j bestimmen.

Eine Verallgemeinerung der Delta-Regel für feedforward-Netze mit nichtlinearen, aber monoton und differenzierbaren Aktivierungsfunktionen, den so genannten *Multi-Layer-Perceptrons* (MLPs), stellt die Backpropagation-Regel dar.

In den folgenden Abschnitten werden diese und weitere Lernregeln beschrieben. Zunächst soll jedoch der einfachste Netztyp, das lineare Netz, betrachtet werden.

Anhang E Clusteranalyse mit neuronalen Netzen

E.1 Lernende Vektorquantisierung (LVQ)

Die lernende Vektorquantisierung nutzt so genannte Codebook-Vektoren, die derart über den gesamten Eingaberaum verteilt sind, dass sie ihn möglichst gut abdecken. Typischerweise entfallen mehrere Codebook-Vektoren auf eine Klasse von Eingabemustern. Derjenige Codebook-Vektor, der einem Eingabevektor am nächsten ist, bestimmt dessen Klasse. Ein Codebook-Vektor ist durch den Eingangsvektor W_j des Kohonen-Neurons j gegeben. Ein derartiges Netz ist in Abbildung E. angegeben. Im Folgenden sind die bekanntesten Lernverfahren für LVQ beschrieben.

E.1.1 LVQ1

LVQ1 bestimmt mit Hilfe einer speziellen Norm ||.|| welches Neuron c den Gewichtsvektor W_c besitzt, dem der Eingabevektor X am ähnlichsten ist. Eine einfache Möglichkeit ist, das Maximum des Skalarproduktes von X und W_j zu suchen.

Das Lernverfahren besteht darin, dass der Gewichtsvektor des Gewinnerneurons noch ähnlicher zum Eingabevektor X gemacht wird, wenn beide zur gleichen Klasse gehören, ansonsten unähnlicher. Da die Ähnlichkeit durch die Differenz der beiden Vektoren dargestellt werden kann, lässt sich dies durch die Addition bzw. Subtraktion eines Teils dieser Differenz erreichen. Häufig wird der Bruchteil $\alpha(t)$ zeitlich veränderlich gewählt und zwar derart, dass er mit der Zeit monoton fällt. Bei LVQ1 bleiben alle anderen Gewichtsvektoren unverändert. Es ergibt sich:

$$W_c(t+1) = \begin{cases} W_c(t) + a(t)[X(t) - W_c(t)] \; falls \; Klasse(W_c) = Klasse(X) \\ W_c(t) - a(t)[X(t) - W_c(t)] \; falls \; Klasse(W_c) \neq Klasse(X) \end{cases} \quad \text{(E.1)}$$
$$W_j(t+1) = W_j(t) \; \forall \, j \neq c$$

Eine Erweiterung von LVQ1 stellt OLVQ1 (*optimized learning vector quantization*) dar. Dabei wird jedem Codebook-Vektor eine eigene Lernrate $\alpha_j(t)$ zugeordnet.

E.1.2 LVQ2.1

Eine andere Möglichkeit der Erweiterung stellt LVQ2.1 dar. LVQ2.1 bestimmt nicht nur den nächsten Codebook-Vektor W_i, sondern auch den übernächsten W_j. Nur wenn die nachfolgenden drei Bedingungen erfüllt sind, findet eine Änderung der Gewichtsvektoren statt [Kohonen+1992] (aus [Zell1994]):

1. W_i und W_j gehören zu unterschiedlichen Klassen.
2. X gehört einer der beiden Klassen an.

3. X liegt in einem "Fenster" entlang der Mittelsenkrechte zwischen den beiden Klassen.

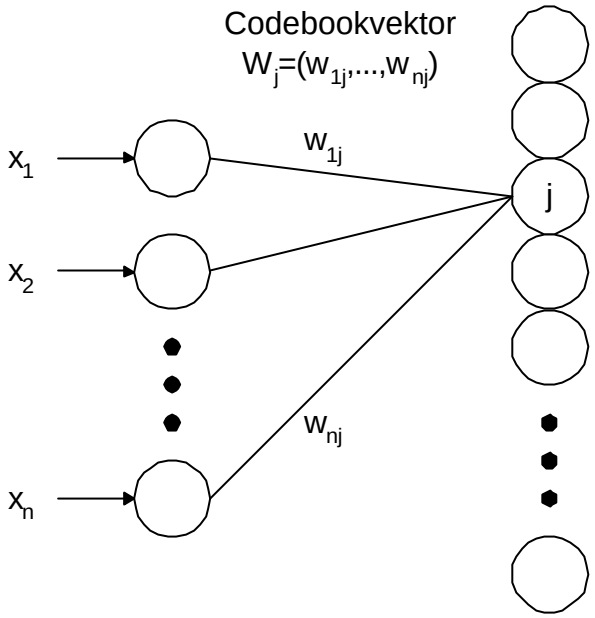


Abbildung E.1: Aufbau eines LVQ-Netzes

Wenn d_i und d_j die Abstände zwischen dem Eingangsvektor X und den Codebook-Vektoren W_i und W_j bezeichnen, dann liegt X in dem beschriebenen "Fenster", wenn gilt:

$$\min\left(\frac{d_i}{d_j},\frac{d_j}{d_i}\right) > s, mit\ s=\frac{1-v}{1+v} \qquad \text{(E.2)}$$

Nach [Kohonen+1992] sollte die Breite des Fensters v zwischen 0,2 und 0,3 betragen. Es ist zu beachten, dass es sich bei dem Fenster um kein Rechteck handelt. Sind diese Bedingungen erfüllt, dann ergeben sich die neuen Gewichtsvektoren zu:

$$\begin{aligned} W_i(t+1) &= W_i(t)+a(t)[X(t)-W_i(t)] \\ W_j(t+1) &= W_j(t)-a(t)[X(t)-W_j(t)] \end{aligned} \qquad \text{(E.3)}$$

Dabei gehören W_i und X zu derselben Klassen und W_j zu einer anderen.

E.1.3 LVQ3

Der Nachteil von LVQ2.1 besteht darin, dass nur noch die Grenzflächen der Klassengrenzen modifiziert werden, jedoch nicht dafür gesorgt wird, dass die Verteilung der Codebook-Vektoren die Dichte der Eingangsvektoren approximiert. Aus diesem

Grund wurde LVQ3 entwickelt, das die Gewichtsvektoren auch ändert, wenn W_i und W_j zur gleichen Klasse gehören. Für diesen Fall ergibt sich für die neuen Codebook-Vektoren:

$$W_i(t+1) = W_i(t) + ea(t)[X(t) - W_i(t)]$$
$$W_j(t+1) = W_j(t) + ea(t)[X(t) - W_j(t)] \qquad \text{(E.4)}$$

Dabei geben [Kohonen+1992] für den Dämpfungsfaktor e Werte von 0,1 bis 0,5 an, die allerdings stark von der Breite des Fensters abhängen. Gehören Wi und Wj zu unterschiedlichen Klassen wird weiterhin (E.3) verwendet.

E.2 Selbstorganisierende Karten (SOM)

Selbstorganisierende Karten (*self organizing maps*, SOM) sind eng mit LVQ verwandt und können als Weiterentwicklung unter Berücksichtigung einer lokalen Nachbarschaftsbeziehung der Neuronen betrachtet werden [Kohonen1990].

Wie bei LVQ handelt es sich bei SOMs um einschichtige Netze, die jedoch mit einem unüberwachten Lernverfahren trainiert werden. Auch hier kommen Codebook-Vektoren zum Einsatz. Allerdings existiert zusätzlich die angesprochene lokale Nachbarschaftsbeziehung zwischen den Neuronen in einem Gitter. Meistens wird ein hexagonales oder quadratisches zweidimensionales Gitter verwendet; aber auch ein- und dreidimensionale Formen sind anzutreffen. Die Nachbarschaftsbeziehung erreicht eine spezielle Art der Vektorquantisierung, die die Gewichtsvektoren durch das Training derart anordnet, daß sie die Eingangsvektoren räumlich geordnet darstellen. Man kann sich vorstellen, daß die Gewichtsvektoren wie in einem elastischen Gitternetz angeordnet sind. Bei Änderung eines Vektors werden auch benachbarte Vektoren geändert. Je größer der Abstand zum „Gewinnerneuron“ ist, desto geringer werden diese Vektoren geändert.

Das Gitternetz der Neuronen definiert eine topologieerhaltende Abbildung der n-dimensionalen Eingangsvektoren auf die m-dimensionalen Gitter. Eine solche Abbildung läßt sich ausgezeichnet zum Auffinden von Ähnlichkeitsbeziehungen in einem hochdimensionalen Eingangsraum verwenden. Die Abbildung wird iterativ mit dem Lernverfahren für SOMs ermittelt.

E.2.1 Lernverfahren der SOM

Das Lernverfahren der SOM ist ein unüberwachtes Lernverfahren. Es wird wie beim LVQ-Training ein n-dimensionaler Eingangsvektor X mit allen Codebook-Vektoren W_j in einer beliebigen Norm verglichen. Meistens kommt die euklidsche Norm bzw. das Skalarprodukt zum Einsatz. Wie bei LVQ wird auf diese Art ein Gewinnerneuron c ermittelt, dessen Gewichtsvektor W_c dem Eingangsvektor am ähnlichsten ist.

Im Unterschied zu LVQ wird jetzt jedoch nicht nur das Gewinnerneuron geändert, sondern alle Neuronen in dessen topologischer Nachbarschaft. Es ist zu beachten, daß diese Neuronen nicht diejenigen sein müssen, die dem Eingangsvektor am zweitähnlichsten sind. Es ergibt sich für die Gewichtsänderung eines beliebigen Neurons j in Abhängigkeit vom Gewinnerneuron c:

$$W_j(t+1) = W_j(t) + h(t)h_{cj}(t)[X(t) - W_j(t)] \qquad \text{(E.5)}$$

wobei η(t) die zeitlich veränderliche Lernrate und $h_{cj}(t)$ die so genannte Distanzfunktion darstellt. Für diese lässt sich auch vereinfachend schreiben:

$$h_{cj}(t) = h\left(\left\| r_c - r_j \right\|, t\right) = h(z,t) = h'(z,d) \qquad \text{(E.6)}$$

Dabei geben r_c und r_j die Gewichtsvektoren der beiden Neuronen c und j an; z bezeichnet den Abstand dieser Vektoren im Gitter. Anstelle des Zeitparameters verwendet man aus Implementierungsgründen häufig einen Distanzparameter d, der die Größe der Umgebung angibt, mit $d \to 0$ für $t \to \infty$. Einige typische Distanzfunktionen sind in Abbildung E. dargestellt.

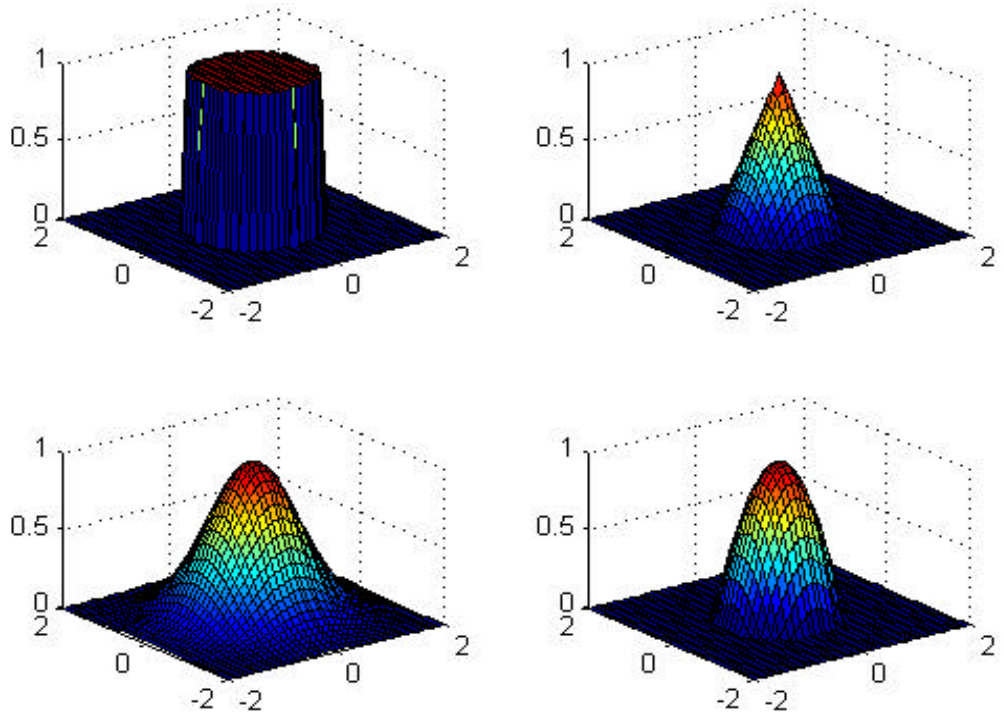


Abbildung E.2: Distanzfunktionen für die Kohonen-Karte

E.3 Hinweise zu LVQ und SOM

Selbstorganisierende Karten werden unüberwacht trainiert und bieten sich insofern für solche Daten an, für die keine Klasseninformation vorliegt. In diesem Fall stellen sie ein Werkzeug zur Clustererkennung der Eingangsdaten dar. Außerdem können sie benutzt werden, um neue Eingangsmuster erkannten Klassen zuzuordnen.

Wenn bereits bekannt ist, dass der Eingaberaum nur in eine endliche Zahl Klassen zerfällt, dann sollte die Repräsentation dieser Klassen besser durch LVQ erfolgen.

Literatur

[Archer2001]
Archer, T; Inside C Sharp; Microsoft Press; München; 2001

[Balzert1996]
Balzert, H.; Lehrbuch der Software-Technik: Software-Entwicklung; Spektrum Akademischer Verlag; Heidelberg; 1996

[Balzert2001]
Balzert, Heide; UML kompakt: mit Checklisten; Spektrum Akademischer Verlag; Heidelberg; 2001

[BauHau1989]
Baum, E. B.; Haussler, D.; What size net gives valid generalization ?; Neural Computation; 1, 1989

[Bellman1961]
Bellman, R.; Adaptive Control Processes: A Guided Tour; Princeton University Press; 1961

[Berger1975]
Berger, B.; Die elastische Verformung der Walzen von Quarto-Walzgerüsten und die Beeinflussung der Walzspaltform durch Walzenbiegeeinrichtungen; Dissertation; TU Clausthal; 1975

[Berger1976]
Berger, B.; Die elastische Verformung der Walzen und des Walzgerüstes; VDEh-Kontaktstudium; Düsseldorf; 1976

[BePaFu1976]
Berger, B.; Pawelski, O; Funke, P.; Die elastische Verformung der Walzen von Vierwalzengerüsten; Archiv Eisenhüttenwesen 47; 1976

[Bishop1995]
Bishop, C.; Neural Networks for Pattern Recognition; Oxford University Press; 1995

[BlaPre1998]
Blaha, Michael; Premerlani, William; Object-Oriented Design of Database Applications; Rose Architect; 1998

[BlaPre1998a]
Blaha, Michael; Premerlani, William; Using UML to Design Database Applications; Rose Architect; 1998

[BMRSS1998]
Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M.; Pattern-orientierte Software-Architektur; Addison-Wesley; Bonn; 1998

[Böhme1993]
Böhme, Johann F.; Stochastische Signale; Teubner; Stuttgart; 1993

[Bolbrinker1989]
Bolbrinker; Stahlfibel; Stahleisen; Düsseldorf; 1989

[Booch1994]
Booch, G.; Object-oriented Analysis and Design with Applications; Benjamin/Cummings Publishing Company; 1994

[BoRuJa1999]
Booch, G.; Rumbaugh, J.; Jacobson, I.; The Unified Modeling Language User Guide; Addison-Wesley; 1999

[Broese1997]
Broese, Einar; Neuronale Netze in der Prozessautomatisierung; Siemens-Dokumentation der Walzenmodelle für die Fertigstraße Bochum; unveröffentlicht; 1997

[Bruns1974]
Bruns, E.; Wärmeübergang und Temperaturfelder beim Warmwalzen von Stahl unter besonderer Berücksichtigung des Zundereinflusses; Dissertation; TU Clausthal-Zellerfeld; 1974

[Brüstle1992]
Brüstle, Roland; Biege-Modell für 6-Walzen-Gerüste; Interner Siemens-Bericht; unveröffentlicht (veröffentlicht als Patentschrift); 1992

[BulHoc1996]
Bulsari, Abhay; Hocksell, Eerik; Neuronal network systems for hardened components; Steel Technology International; 1996

[Cacoulos1966]
Cacoullos, T.; Estimation of a multivariate density; Annals of the Institute of Statistical Mathematics (Tokio); 18(2), 1966

[CaEdTh1992]
Carlton, A. J.; Edwards, W. J.; Thomas, P. J.; Formulae for cold rolling analyses; 1992

[Cattel1996]
Cattel, R. G. G. (Editor); The Object Database Standard; ODMG-93, Release 1.2; Morgan Kaufmann Publishers; San Francisco; 1996

[CoaYou1991]
Coad, P.; Yourdon, E.; Object-oriented Analysis; 2. Auflage; Yourdon Press; Prentice Hall; Englewood Cliffs, 1991

[CoaYou1991a]
Coad, Peter; Yourdon, Edward; Object-oriented design; Prentice Hall; Englewood Cliffs; 1991

[Codd1990]
Codd, E. F.; The Relational Model for Database Management; Addison-Wesley; 1990

[CooMcC1958]
Cook, P. M.; McCrum, A. W.; The calculations of load and torque in hot flat rolling; The British Iron and Steel Association (BISRA); Sheffield; 1958

[CseMen1999]
Cser, L.; Menzel, T.; Data Mining in der Qualitätssicherung umformtechnischer Prozesse; Umformtechnik 2000 PLUS; Meisenbach Bamberg; 1999

[Czichos1996]
Czichos, Horst; Hütte – Die Grundlagen der Ingenieurwissenschaften; Springer Verlag; Berlin; 1996

[DahKop1993]
Dahl, W. (Hrsg.); Kopp, R. (Hrsg.), Umformtechnik, Plastomechanik und Werkstoffkunde; Verlag Stahleisen; Düsseldorf; 1993

[Date1990]
Date, C. J.; An Introduction to Database Systems, Volume 1; Addison-Wesley; 1990

[Degner1993]
Degner, Michael; Untersuchung der Optimierung der Maßhaltigkeit von warmgewalzten Bändern; Dissertation; Technische Universität Clausthal; 1993

[DrRaWi1979]
Drexler, W.; Rademacher, W.; Willner, L.; Regelung und Simulation einer kontinuierlichen Warmwalzstraße; Betriebsforschungsinstitut - Bericht Nr. 656; Düsseldorf; 1979

[DSoWil1999]
D´Souza, D.; Wills, A. C.; Objects, Components, and Frameworks with UML: The Catalysis Approach; Addison-Wesley; 1999

[FaScJo1999]
Fayad, M. E.; Schmidt, D. C.; Johnson, R. E.; Building Application Frameworks: OO Foundations of Framework Design; John Wiley and Sons; New York, NY; 1999

[GHJV1995]
Gamma, E.; Helm, R.; Johnson, R.; Vlissades, J. O.; Design Pattern: Elements of Reusable OO Software; Addison-Wesley; Reading, MA; 1995

[Ginzburg1989]
Ginzburg, Vladimir B.; Steel-rolling technology: theory and practice; Marcel Dekker; New York; 1989

[Hannemann1998]
Hannemann, Jan; Vergleich mathematischer Modelle und neuronaler Netze – bei der Analyse und Prognose von Umweltdaten; Diplomarbeit; Fachbereich Mathematik/Informatik; Angewandte Systemwissenschaften; Universität Osnabrück; 1998

[Harms1975]
Harms, H.-W.; Walzguttemperaturen in neuzeitlichen Warmbreitbandfertigstaffeln – Grenzen hoher Walzgeschwindigkeiten und Möglichkeiten gezielter Walzgutkühlung; Dissertation; RWTH Aachen; 1975

[Hartung1984]
Hartung, Joachim; Elpelt, Bärbel; Klösener, Karl-Heinz; Statistik: Lehr- und Handbuch der angewandten Statistik; R. Oldenbourg Verlag; München; 1984

[Haykin1994]
Haykin, Simon; Neural networks; Macmillan College Publishing; New York; 1994

[Hebb1949]
Hebb, D. O.; The Organization of Behaviour; Wiley; New York; 1949 (auch in [Rosenfeld 88])

[Hilbert1900]
Hilbert, D.; Mathematische Probleme; Nachrichten der Akademie der Wissenschaften Göttingen; 1900

[Hinkfoth1989]
Hinkfoth, R.; Die Anwendung der elementaren Plastizitätstheorie in der Umformtechnik, speziell beim Warm- und Kaltwalzen; Freiberger Forschungshefte; Band 242; VEB Deutscher Verlag für Grundstoffindustrie; Leipzig; 1989

[Hinton1992]
Hinton, Geoffrey E.; Wie neuronale Netze aus Erfahrung lernen; Spektrum der Wissenschaft; November 1992

[HitTri1935]
Hitchcock, J. H.; Trinks, W.; Roll neck bearings; Publication by the American Society of Mechanical Engineers; New York; 1935

[HofDah1955]
Hoff, H.; Dahl, T.; Grundlagen des Walzverfahrens; Verlag Stahleisen; Düsseldorf; 1955

[Holzner2002]
Holzner, Steven; XSLT – Anwendung und Referenz; XML-Transformationen, XPath, Einsatz mit Java, JSP und ASP; Markt+Technik Verlag; München; 2002

[Jacobson1992]
Jacobson, Ivar; Christerson, Magnus; Jonsson, Patrik; Övengaard, Gunnar; Object-oriented Software-Engineering - A Use Case driven Approach; Addison-Wesley; 1992

[JäTaWi2000]
Jäckel, Ingolf; Tamler, Horst; Wick, Carsten; NEUROLL – Draft Final Technical Report; Task 7: Thickness control; Project BE96-3061; Bochum; 2000

[Jansen1995]
Jansen, Michael; Globale Modellbildung und garantiert stabile Regelung von Robotern mit strukturierten neuronalen Netzen; VDI Forschungsberichte; VDI Verlag; Düsseldorf; 1995

[JBFGP+1998]
Jansen, Michael; Broese, Einar; Feldkeller, Björn; Gramckow, Otto; Poppe, Thomas; Schlang, Martin; Sörgel, Günther; Application of neural networks to process control in steel industry; IFAC; 1998

[KHS1995]
Krupp Hoesch Stahl AG; Anlagenbeschreibung des Bochumer Warmbreitbandwalzwerkes der Krupp Hoesch Stahl AG; unveröffentlicht; 1995

[Kohonen1982]
Kohonen, Teuvo; Self-organized formation of topologically correct feature maps; Biological Cybernetics; 43, 1982

[Kohonen1990]
Kohonen, Teuvo; The Self-Organizing Map; Proceedings of the IEEE; Volume 78(9); September 1990

[Kohonen1992+]
Kohonen, Teuvo; Kangas, J.; Laaksonen, J.; Torkkola, K.; LVQ-PAK, The Learning Vektor Quantization Program Package, Version 2.1; LVQ Programming Team of the Helsinki University of Technology; Laboratories of Computer and Information Science; 1992

[Kolmogorov1957]
Kolmogorov, A. N.; On the representation of continuous functions of serveral variables by superposition of continuous function of one variable and addition; Doklady Akademiia Nauk SSSR 114;1957

[Lange1988]
Lange, K.; Lehrbuch der Umformtechnik, Band II – Massivumformung; Springer; Berlin; 1988

[Langer1976]
Langer, U.; Wärmebilanz beim Warmwalzen; Stahl und Eisen; Heft 25/26; 1976

[Lenze1997]
Lenze, Burkhard; Einführung in die Mathematik neuronaler Netze – mit C-Anwendungsprogramme im Internet; Logos Verlag Berlin; 1997

[LePiCs1999]
Lenard, J.; Pietrzyk, M.; Cser, L.; Mathematical and Physical Modelling of Hot Rolling; Elsevier; 1999

[LipMah1967]
Lippmann, H.; Mahrenholtz, O.; Plastomechanik der Umformung metallischer Werkstoffe – Band 1; Springer Verlag; 1967

[Loomis1993]
Loomis, M.E.S.; Making object persistent; JOOP, Oct. 1993; pp. 25-28

[LSGK1994]
Lindhoff, Dieter; Sörgel, Günter; Gramckow, Otto; Klode, Klaus-Dieter; Erfahrungen beim Einsatz neuronaler Netze in der Walzwerksautomatisierung; Stahl und Eisen; 4, 1994

[MGPS1996]
Martinez, Thomas; Gramckow, Otto; Protzel, Peter; Sörgel, Günter; Neuronale Netze zur Steuerung von Walzstraßen; atp; R. Oldenbourg Verlag; Nr. 10; 1996

[Masters1993]
Masters, Timothy; Practical Neural Network Recipes in C++; AP Professional; 1993

[McCPit1943]
McCulloch, W. S.; Pitts, W; A logical calculus of the ides immanent in nervous activity; Bulletin of Mathematical Biophysics; 5, 1943

[Michel1999]
Michel, Thomas; XML kompakt – Eine praktische Einführung; Carl Hanser Verlag; München; 1999

[Niemann2000]
Niemann, Christoph; Datenbankfähige Client/Server-Anwendungen – Generierung aus OOA-Modellen; Dissertation; Spektrum Akademischer Verlag; Heidelberg; 2000

[Orowan1943]
Orowan, E.; The calculation of roll pressure in hot and cold flat rolling; Proc. Instn. Mech. Engrs. 150; 1943

[OrtGon1998]
Ortega, Francisco; González, Juan A.; HSM Force Pre-Setting using artificial neural networks and statistical techniques; 1998

[Parzen1962]
Parzen, E.; On estimation of a probability density function and mode; Annals of Mathematical Statistics; Vol. 33; 1962

[Pawelski1969]
Pawelski, O.; Berechnung des Temperaturverlaufs in der Fertigstraße einer Warmbreitbandstraße; Stahl und Eisen; Nr. 89; 1969

[Pawelski1969a]
Pawelski, O.; Berechnung der Wärmedurchgangszahl für das Warmwalzen und Schmieden; Archiv für das Eisenhüttenwesen; Heft 10; 1969

[Pawelski1971]
Pawelski, O.; Berechnung des Tremperaturfeldes in Arbeitswalzen von Warm- und Kaltwalzwerken; Archiv für das Eisenhüttenwesen; Heft 10; 1971

[PawBru1976]
Pawelski, O.; Bruns, E.; Wärmeübergang und Temperaturfelder beim Warmwalzen von Stahl unter besonderer Berücksichtigung des Zundereinflusses; Stahl und Eisen; Heft 18; 1976

[PawSch1969]
Pawelski, O.; Einfluss der Walzenabplattung auf Walzspaltform und Druckverteilung beim Kaltwalzen; Archiv Eisenhüttenwesen; Nr. 40; 1969

[PetPlü1994]
Peters, Harald; Plüm, Hans-Dieter; Künstliche neuronale Netze und deren Anwendung in der Stahlindustrie; Stahl und Eisen; Nr. 4; 1994

[PreTre1994]
Preuß, T.; Tresp, V.; Neuro-Fuzzy; atp-Automatisierungstechnik Praxis; Heft 5; 1994

[RBPEL1993]
Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.; Objektorientiertes Modellieren und Entwerfen; Hanser, München; Prentice-Hall International, London; 1993

[Rocha1992]
da Rocha, Armando Freitas; Neural Nets – A Theory for Brains and Machines; Springer Verlag; Berlin; 1992

[Rogers1996]
Rogers, Joey; Objekct-oriented Neural Networks in C++; Harcourt Publishers Ltd.; 1996

[Rojas1996]
Rojas, Raúl; Theorie der neuronalen Netze - Eine systematische Einführung; Springer Verlag; Berlin; 1996

[Rosenfeld1988]
Rosenfeld, E. (Editor); Neurocomputing: Foundations of Research; MIT Press; 1988

[RuJaBo1999]
Rumbaugh, J.; Jacobsen, Ivar; Boooch, Grady; The Unified Language Reference Manual; Addison-Wesley; Reading Massachusetts; 1999

[RumMcC1986]
Rummelhart, D. E.; McClelland, J. L.; Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1 & 2; The MIT Press; 1986

[ScHeFe1994]
Schöneburg, Eberhard; Heinzmann, Frank; Fedderson, Sven; Genetische Algorithmen und Evolutionsstrategien; Addison-Wesley; Bonn; 1994

[SchKum2001]
Schneider, Manfred; Kummerow, Alexander; Rechnergestützte Prozessdiagnose bei Qualitätsproblemen in Flachwalzwerken; VDI-Berichte; Nr. 1590; 2001

[Schöneburg1993]
Schöneburg, Eberhard (Hrsg.); Industrielle Anwendung Neuronaler Netze – Fallbeispiele und Anwendungskonzepte; Addison-Wesley; Bonn; 1993

[Schwarzer1971]
Schwarzer, J.; Entwicklung eines mathematischen Modells zur Ermittlung der Auffederung eines Quarto-Gerüstes bei besonderer Berücksichtigung des Walzeneinbaus; Dissertation; Clausthal; 1971

[Schwenzfeier1979]
Schwenzfeier, W.; Walzwerktechnik – Ein Leitfaden für Studium und Praxis; Springer; Wien; 1979

[SFJKP+1996]
Schlang, Martin; Feldkeller, Björn; Jansen, Michael; Köhler, G.; Poppe, Thomas; Schäffner, C.; Broese, Einar; Gramckow, Otto; Sörgel, Günther; Neuronale Netze zur Prozeßsteuerung in der Stahlverarbeitung; VDI Berichte Nr. 1282; 1996

[Siemens1997]
Siemens AG; Prozessautomatisierung von Warmbreitbandstraßen; Solutions For Metals, Mining, and More; Siemens AG; Erlangen; 1997

[Siemens1999]
Siemens AG; Anlagendokumentation Prozessautomatisierung Fertigstraße TKS Bochum; Erlangen; 1999

[Sims1952]
Sims, R. B.; Calculation of roll force and torque in hot rolling mills; Proc. Instn. Mech. Engrs. 166; 1952

[Specht1990]
Specht, D. F.; Probabilistic neural networks and the polynomial adaline as complementary techniques for classification; IEEE Transactions on Nerual networks; 1(1); March 1990

[Specht1990a]
Specht, D. F.; Probabilistic neural networks; Nerual networks; 3, 1990

[Sturm2000]
Sturm, Jake; XML-Lösungen programmieren; Die wichtigsten XML-Technologien und –Spezifikationen praxisnah erklärt; Microsoft Press; Unterschleißheim; 2000

[TKS1998]
Thyssen Krupp Stahl AG; Warmbandwerk Bochum; unveröffentlicht; 1998

[VapChe1971]
Vapnik; Chervonenkis; On the uniform convergence of relative frequencies of events to their propabilities; Theory of Probability and its Applications; 16(2), 1971

[Vasters2000]
Vasters, Clemens; BizTalk Server 2000: A Beginner´s Guide; Osborne Verlag; 2000

[Vasters2001]
Vasters, C. et al.; .NET Crashkurs: C#, .NET Framework, ASP.NET, VB.NET, ADO.NET, Managed C++; Microsoft Press; München; 2001

[Wagner1999]
Wagner, Stefan; Optimierung Neuronaler Netze durch Evolutionsstrategien und weitere Verknüpfungsmöglichkeiten in der Computational Intelligence; Dissertation; Fachbereich Maschinenbau; Gerhard-Mercator-Universität, Duisburg; 1999

[Werbos1974]
Werbos, P. J.; Beyond regression: new tools for prediction and analysis in behavioral science; Ph.D. thesis; Harvard University; Cambridge; 1974

[Wick1996]
Wick, Carsten; Konstruktion eines Schemagenerators für relationale Datenbanksysteme und einer Schnittstelle zu C++; Diplomarbeit; Lehrstuhl für Softwaretechnik; Ruhr-Universität Bochum; 1996

[Wiegels1976]
Wiegels; Geometrie des Walzspaltes, Formänderung; VDEh-Kontaktstudium; Düsseldorf; 1976

[Wiegels1976a]
Wiegels; Kinematik des Walzspaltes, Formänderungsgeschwindigkeit; VDEh-Kontaktstudium; Düsseldorf; 1976

[Willms2000]
Willms, André; C++ STL – Verstehen, anwenden, erweitern; Galileo Computing; Bonn; 2000

[Zell1994]
Zell, Andreas; Simulation neuronaler Netze; R. Oldenbourg Verlag; München; 1994

Lebenslauf

Persönliche Daten

Name	Carsten Wick
Anschrift	Friedrich-Hebbel-Str. 4b, 58456 Witten
Geburtsdatum	20. April 1972
Geburtsort	Bochum
Eltern	Dietmar Richard Wick, Dipl.-Ingenieur
	Marlis Ursula Wick, geb. Schopmeier, Übersetzerin
Familienstand	verheiratet mit
	Anke Wick, geb. Eßer, Dipl.-Sozialpädagogin
Staatsangehörigkeit	Deutsch

Ausbildung

1978-1982	Grundschule Bochum
1982-1991	Graf-Engelbert-Gymnasium in Bochum,
	Abschluss: Abitur
1991-1996	Diplomstudium Elektrotechnik an der
	Ruhr-Universität in Bochum, Abschluss: Diplom
1997-2002	Promotionsstudium als freier Doktorand bei
	Herrn Univ.-Prof. Dr.-Ing. Mathias Uhle am Fachgebiet
	Messtechnik der Universität Dortmund

Beruflicher Werdegang

1997-2000	Krupp Hoesch Stahl AG bzw. ThyssenKrupp Stahl AG,
	Bochum, Doktorand
1996-1998	Schleupen AG, Moers, Programmierer
1998-2000	Selbstständiger Softwareberater und Programmierer
2000-Sep. 2002	/S/I/S/Z/ GmbH, Dortmund, Projektleiter
seit Okt. 2002	BTI Central Europe, Köln, Leiter Development