Einsatz von Backpropagation-Netzen zur Abbildung des Rückfederungsverhaltens bei der Feinblechumformung

Tawil, M.

Im vorliegenden Artikel werden künstliche neuronale Netze (KNN) als Werkzeug zur Abbildung der Zusammenhänge während des Tiefziehprozesses vorgestellt. Der Fokus der im Rahmen eines DFG-Forschungsprojektes durchgeführten Untersuchung lag dabei auf KNN-Varianten, die auf dem am meisten verbreiteten Backpropagation-Algorithmus basieren. Der entscheidende Vorteil dieses Lernalgorithmus ist, nichtlineare Systeme durch sogenannte Eingabe-Ausgabe-Vektoren abzubilden und daher Ingenieurwissen direkt in die Netzstruktur einzubinden.

In the article at hand artificial neural networks, used as tools for the illustration of interrelationships during deep drawing, will be presented. The focus of the study, which was part of a DFG-research project, was placed on neural network variants that are based on the most widespread backpropagation algorithm. The crucial advantage of this learning algorithm is the capability to map non-linear systems using so called input-output vectors, thereby including engineering knowledge directly in the network structure.

1 Einleitung

Die Maßhaltigkeit bzw. Rückfederung tiefgezogener Blechteile spielt eine zentrale Rolle bei der Beurteilung der Tiefziehprozess-Genauigkeit. Eine hundertprozentige Beherrschung dieses Verfahrens stellt für die blechverarbeitende Industrie, insbesondere die Automobilhersteller und ihre Zulieferer eine große Herausforderung dar. Dabei ist die Bauteilmaßhaltigkeit ein wichtiger Faktor nicht nur bei der Gewährleistung der fehlerfreien Montage und Fügbarkeit. Auch der Automatisierungsgrad mehrerer hintereinander geschalteter Fertigungs- und Montageschritte hängt stark von der Genauigkeit der gefertigten Blechteile ab.

Am Ende des Umformvorganges erfahren die Blechteile eine dreidimensionale, inhomogene Formänderung. Die Ursache hierfür liegt im elastisch-plastischen Verhalten der Bleche, welches nach dem Kraftrückgang des Werkzeuges eine Rückfederung des tiefgezogenen Werkstückes hervorruft. Somit ist die Rückfederung eine den Kaltumformprozess begleitende Formabweichung, die verringert, korrigiert oder zumindest berücksichtigt werden muss. Dies ist mit enormem technischem und wirtschaftlichem Aufwand verbunden /1/, da die Rückfederung von einer sehr großen Anzahl von Einflussgrößen abhängig ist.

Die analytische Ermittlung der Gestaltänderung von Tiefziehteilen ist sehr kompliziert und basiert auf vereinfachten mathematischen Modellen, so dass eine genaue Aussage über die Fertigungstoleranzen wie sie heute gefordert werden, nicht möglich ist. Erst der Einsatz von Finite Elemente (FEM) Programmen zur Simulation des Umformvorganges ermöglichte gute Ergebnisse bezüglich der Dehnungsverteilung /2/. Bei der Rückfederungssimulation jedoch stoßen diese numerischen Lösungen an ihren Grenzen und liefern bei großem Aufwand und langer Erfahrung nur für einfache Blechteile befriedigende Aussagegenauigkeit. Dies liegt zum einen an den den Programmen zu Grunde liegenden Werkstoffmodellen, die dem tatsächlichen Werkstoffverhalten nicht genügen /3/.

Die vorhandenen Lösungen zur Berechnung der Rückfederung nach dem Tiefziehen reichen aus heutiger Sicht nicht aus, um qualitativ hochwertige Produkte aus Feinblech herzustellen. Der Einsatz von KNN soll hier Abhilfe schaffen. Die hervorragende Eignung von KNN zur Lösung technischer Problemstellungen liegt hauptsächlich an folgenden positiven Eigenschaften:

- Approximation komplexer nichtlinearer Zusammenhänge, ohne die Notwendigkeit eines mathematischen Modells, da das Lernen musterbasierend erfolgt
- Eliminieren geringer Datenstreuung
- Fehlertoleranz gegenüber teilweise fehlerhaften Daten
- Sehr gute Interpolations- und adäquate Extrapolationsfähigkeit

- Die Gewährleistung einer realitätsgetreuen Prozesssimulation beim Trainieren mit Daten, die der Realität entstammen
- Generalisierungsfähigkeit

Diesen Vorteilen stehen einige Nachteile gegenüber:

- Die fehlende Transparenz der in den Netzen stattfindenden Lernvorgänge erschwert die Nachvollziehbarkeit der Netzergebnisse
- Es existieren keine allgemein gültigen Theorien zur Konstruktion künstlicher neuronaler Netze

Insbesondere bei der Auswahl der geeigneten Netzkonfiguration wird vom Anwender sehr viel Erfahrung im Umgang mit dem jeweiligen Netztyp abverlangt. Fehlt diese, vergeht sehr viel Zeit, bis die optimale Konfiguration gefunden wird. Ein weiterer Aspekt, der in der Vergangenheit von vielen Anwendern wenig Beachtung bekam, ist die Beschaffenheit des der Trainingsphase zu Grunde liegenden Datensatzes. Ist in diesem der vom Netz zu beschreibende Sachverhalt nicht korrekt wiedergegeben, scheitert die Anwendung unweigerlich. Aus den genannten Gründen wurden KNN zur Lösung einiger Problemstellungen unberechtigter Weise als ungeeignet bezeichnet. Dabei hätte dort allein eine adäquate Datenaufbereitung zu einem positiven Urteil geführt.

Ziel des Projektes war daher, die Konstruktion eines Netzes zur Rückfederungsberechnung verschiedener Bauteilgeometrien mit variabler Blechstärke basierend auf realen Experimenten. Zu diesem Zweck wurden Netze mit den drei am häufigsten verwendeten Modifikationen der Standard-Backpropagation-Lernregel untersucht:

- Backpropagation mit Momentum
- Backpropagation mit adaptiver Lernrate und Momentum
- Resilient Propagation

Dabei wurden bei jedem Netztyp nach einem vollfaktoriellen Berechnungsplan verschiedene Variationen der Netztopologie und Lernparameter durchgerechnet. Abschließend wurden die Leistungsfähigkeiten der drei Algorithmen bei der Abbildung des Tiefziehprozesses miteinander verglichen.

Im folgenden wird der allgemeine Aufbau und die prinzipielle Arbeitsweise von Backpropagation-Netzen kurz beschrieben, um anschließend die Untersuchungsergebnisse vorzustellen.

2 Arbeitsweise der Backpropagation-Netze

Beim Backpropagation-Algorithmus handelt es sich um ein numerisches Lernverfahren zum überwachten Trainieren vorwärtsgerichteter, mehrschichtiger KNN. In den letzten Jahrzehnten wurde der Backpropagation-Algorithmus, angepasst an die jeweilige Problemstellung, von verschiedenen Forschern als Lernregel für unterschiedliche Netzarten implementiert. Heute existiert eine Vielzahl solcher Netze, die unter dem Sammelbegriff „Backpropagation-Netze“ bekannt sind. Sie unterscheiden sich im wesentlichen durch die zum Lernen verwendete Variante des Backpropagation-Algorithmus.

2.1 Netzaufbau

Backpropagation-Netze bestehen, in Anlehnung an das biologische Nervensystem aus vielen Einheiten, den Neuronen, die in einer oder mehrerer Schichten miteinander vernetzt sind und unabhängig voneinander denkbar einfache Rechenoperationen durchführen /4, 5/. Prinzipiell kann ein Neuron als eine autonom agierende Schaltung verstanden werden, welche eine Ausgabe erzeugt, wenn der kumulative Effekt der Eingabereize einen bestimmten Schwellenwert, den sog. „Biaswert“ b übersteigt (**Bild 1**). Resultierend daraus gilt, dass es für jeden Eingabezweig externe Eingabesignale oder Reize sowie eine entsprechende Gewichtung gibt. Jeder von außen ins Neuron eingehende Wert e wird mit einem Gewicht w multipliziert. Dabei kann dieser Wert (Signal) vom Anwender an der Eingabeschicht festgelegt worden sein oder, falls es sich nicht um ein Eingabeneuron handelt, die Ausgabe eines vorgeschalteten Neurons sein. Durch die Gewichtung wird das Signal entweder geschwächt oder verstärkt dem Neuron präsentiert.

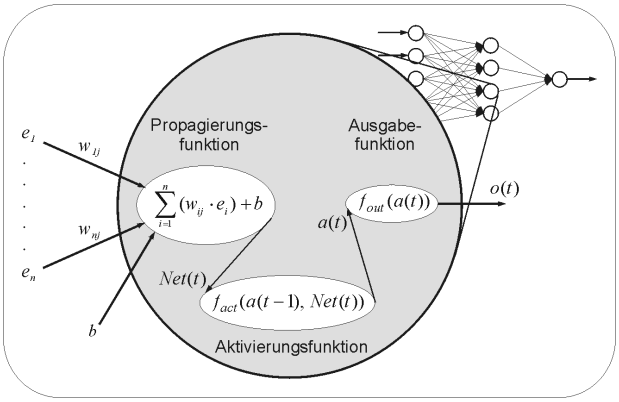

Bild 1: Modell eines künstlichen Neurons

Zur Erzeugung der Neuronenausgabe $o(t)$ müssen die Eingabewerte mehrere mathematische Funktionen durchlaufen. So wird mit der Übertragungsfunktion, Propagierungsfunktion genannt, die Gesamtfremderregung des Neurons durch Zusammenfassung der Eingangssignale zum Zeitpunkt t ermittelt. Diese Zusammenfassung kann durch Addition, Produktbildung oder Bildung des Maximal- oder Minimalwertes der gewichteten Eingaben geschehen, wobei in den bekannten Neuronen die Summation als Rechenoperation am häufigsten verwendet wird. Das Ergebnis dieser Berechnung $Net(t)$ wird im nächsten Schritt transformiert. Die Transformationsvorschrift besagt, dass mit einer Aktivierungsfunktion festgelegt wird, wie sich aus dem vorherigen Aktivierungszustand $a(t-1)$ und der momentanen Gesamtfremderregung $Net(t)$ ein neuer Aktivierungszustand $a(t)$ des Neurons berechnen lässt. Anschließend wird mit der Ausgabefunktion $f_{out}(a(t))$ der Neuronenausgang $o(t)$ ermittelt, der den nachfolgenden Neuronen als Eingabe dient oder, wenn es sich um ein Ausgabeschichtneuron handelt, als endgültige Netzausgabe gilt.

Ein Backpropagation-Netz besteht, wie in **Bild 2** gezeigt, aus einer Eingabeschicht, einer oder mehrerer verdeckter Schichten und einer Ausgabeschicht. Die jeweiligen Schichten verfügen über eine je nach Anwendungszweck unterschiedliche Anzahl von Neuronen, die innerhalb der Schicht keine Verbindungen untereinander aufweisen. D.h. es bestehen lediglich Verbindungen von den Neuronen einer Schicht zu den Neuronen der nachfolgenden Schicht.

Die Neuronen der Eingabeschicht dienen zum Repräsentieren der Daten. Sie leiten die Eingangsgrößen unverändert an die versteckten Schichten weiter. Dabei wird die Ausgabe eines jeden Eingabeneurons mit einem Gewichtsfaktor multipliziert und an alle Neuronen der versteckten Schicht verteilt.

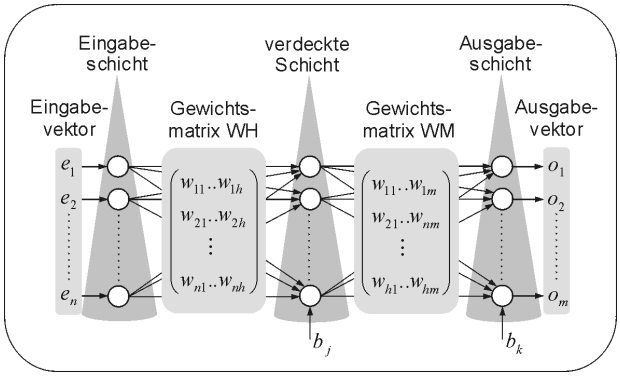

Bild 2: Topologie eines Backpropagation-Netzes

Jedes Neuron der versteckten Schicht erhält eine definierte Anzahl von gewichteten Eingängen, die der Anzahl der Eingabeneuronen entspricht. Zusätzlich zur Produktsumme aus Eingaben und Gewichtsfaktoren geht bei einigen Anwendungen ein Biaswert b_j in die Berechnung der Propagierungsfunktion jedes einzelnen Neurons ein. In den Neuronen der versteckten Schichten findet die eigentliche Informationsverarbeitung statt. Die Anzahl der versteckten Schichten sowie die darin enthaltenen Neuronen, die nichtlineare Aktivierungsfunktionen berechnen können, beeinflussen das Netzverhalten massiv und müssen daher sorgfältig gewählt werden. Backpropagation-Netze verfügen über mindestens eine versteckte Schicht zur Abbildung beliebiger nichtlinearer Zusammenhänge. Bei komplexen Netzwerken sind mehrere verdeckte Schichten durchaus möglich. Die Gewichtsfaktoren an der Schnittstelle zwischen der Eingabeschicht und der ersten versteckten Schicht bilden die Gewichtsmatrix WH und werden in der Regel, wie alle Gewichtsfaktoren des Netzes, am Anfang der Trainingsphase zufällig initialisiert.

Die Ausgaben der ersten verdeckten Schicht werden gewichtet an die nachfolgenden Schichten, beispielsweise die Ausgabeschicht weitergeleitet. Letztere dient zur Ausgabe der vom Netzwerk erzeugten Antworten auf die Eingabewerte. Die Anzahl der Ausgabeschichtneuronen entspricht meist der Anzahl der vom Netz zu erzeugenden Ausgaben. Die Gewichtsfaktoren auf den Verbindungen zwischen der letzten versteckten Schicht und der Ausgabeschicht sind in der Gewichtsmatrix WM zusammengefasst. Außerdem erhält in manchen Fällen jedes Neuron dieser Schicht einen Biaswert b_k.

2.2 Lernprozess bei Backpropagation-Netzen

Bevor ein Netz zur Berechnung einer bestimmten Ausgabe eingesetzt werden kann, müssen zwei Phasen (Training und Validierung) abgearbeitet werden. In der Trainingsphase wird dem Netz eine Trainingsmenge, bestehend aus Eingabe- und Ausgabepaaren, präsentiert. Das Netz lernt den in der Trainingsmenge vorhandenen Sachverhalt, in dem der Anwender ihm mitteilt, welche Ausgabe erwartet wird, wenn eine bestimmte Eingabe vorliegt. Die Abbildungsgenauigkeit des Netzes wird durch die Abweichung der errechneten Netzausgaben von der vorgegebenen Soll-Ausgabe definiert. Die Validierungsphase dient zur

Überprüfung der Generalisierungsfähigkeit des gerade trainierten Netzes. Dabei präsentiert der Anwender dem Netz Eingabedaten, die in der Trainingsmenge nicht enthalten waren und somit für das Netz unbekannt sind. Erst wenn das Netz das gelernte Wissen gut verallgemeinern kann, ist der Lernprozess abgeschlossen. Das während des Lernprozesses vom Netz erlangte Wissen ist in der Netzstruktur und in den Gewichtsfaktoren, mit denen die Neuronenverbindungen belegt sind, dauerhaft gespeichert.

Der Lernprozess spielt sich in der wichtigsten und gleichzeitig aufwendigsten Trainingsphase ab. Das Lernen wird beim Backpropagation-Algorithmus in drei Schritten vollzogen. Zunächst erfolgt die Berechnung der Netzausgabe über alle Netzschichten von links nach rechts. Im zweiten Schritt wird die Differenz (der Netzfehler) zwischen errechneter und gewünschter Ausgabe gebildet. Um den Netzfehler zu minimieren, wird er im dritten Schritt ausgehend von der Ausgabeschicht an die vorgeschalteten Schichten zurückgegeben, wobei die Gewichtsfaktoren nach einer Lernvorschrift iterativ angepasst werden. Das Ziel des Backpropagation-Algorithmus ist, Netzgewichte zu finden, die den Netzfehler minimieren.

3 Abbildung des Tiefziehprozesses mittels Backpropagation-Netze

Im Rahmen der durchgeführten Untersuchung wurde das Rückfederungsverhalten tiefgezogener Bauteile aus Feinblech basierend auf realen Experimenten mit Hilfe von Backpropagation-Netzen prognostiziert. Zu diesem Zweck wurden nach einem statistischen Versuchsplan zwei Tiefziehgeometrien (**Bild 3**) unter Variation verschiedener Prozessparameter und Blechstärken hergestellt.

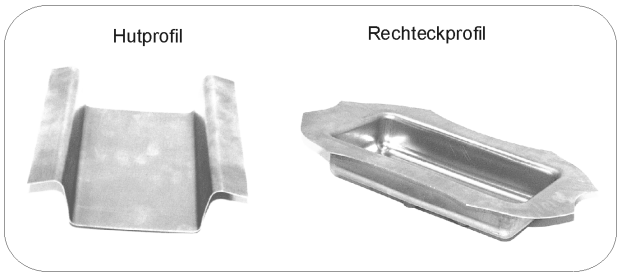

Bild 3: Untersuchte Tiefziehgeometrien

Die hergestellten Werkstücke wurden bezüglich ihrer Rückfederung vermessen. Es entstand ein statistisch gesicherter Datensatz, der die Tiefziehrealität beinhaltete. Mit diesem Datensatz konnten drei verschiedene Netztypen, die mit unterschiedlichen Modifikationen des Backpropagation-Algorithmus arbeiten, trainiert werden. Im folgenden werden die Simulationsergebnisse der jeweiligen Netztypen vorgestellt.

3.1 Backpropagation mit Momentum (BP-M)

Der hier behandelte Lernalgorithmus verwendet zur Gewichtsänderung eine konstante Lernrate und einen sog. Momentum-Term. Zur Ermittlung der besten Netzkonfiguration für die vorliegende Aufgabenstellung wurden die Neuronenzahlen in der Zwischenschicht, die Lernrate und das Momentum variiert. Dazu wurden für die betrachteten Parameter unterschiedliche Einstellwerte berücksichtigt. Es wurden für sämtliche Berechnungen jeweils 10000 Iterationsschritte festgelegt. In Voruntersuchungen für die vorliegende Problemstellung wurden Netze mit mehr als einer Zwischenschicht betrachtet. Diese lieferten schlechtere Ergebnisse im Vergleich zu Netzen mit nur einer verdeckten Schicht. Aus diesem Grund wurden hier ausschließlich Netze mit einer verdeckten Schicht betrachtet.

Für die Neuronenaktivierung in der Zwischenschicht wurde die logistische Signalfunktion und in der Ausgabeschicht die Identität ausgewählt. Diese eigneten sich sehr gut zur Lösung ähnlicher Aufgabenstellungen. Zur Berechnung des Netzfehlers wurde der mittlere quadratische Fehler (MSE) bei allen Netzen verwendet. Das error goal wurde mit 0,005 angegeben.

Die Kombinationen aller Parameterstufen wurden nach einem vollfaktoriellen Berechnungsplan ermittelt. Dieser ergab bei 4 Einstellwerten für die Neuronenzahl, 5 Einstellwerten für die Lernrate und 4 Einstellwerten für das Momentum 80 unterschiedliche Netzkonfigurationen. Dadurch erhöht sich zwar der Rechenaufwand erheblich, jedoch kann nur so die Berücksichtigung aller möglichen Parameterkombinationen gewährleistet werden. Für jede Netzkonfiguration wurden, um die Trainingsergebnis-Schwankungen aufgrund der zufälligen Gewichtsinitialisierung zu verringern, fünf Durchläufe durchgeführt. Am Ende eines jeden Trainings wurde das jeweilige Netz mit dem speziell hierfür vorbereiteten Testdatensatz bezüglich seiner Generalisierungsfähigkeit getestet.

3.1.1 Simulationsergebnis

Bei der Anwendung dieses Algorithmus zur Berechnung der Rückfederung erreichte, unter den 80 durchgeführten Berechnungen ein Netz mit 5 Zwischenschichtneuronen, einer Lernrate von 0,1 und einem Momentum von 0,3 den geringsten Generalisierungsfehler von 4,2%. Das error goal wurde hier nach 5000 Iterationsschritten in 50 Sekunden unterschritten. Der Generalisierungsfehler wurde anhand des Testmusters im Validierungsdatensatz berechnet. In **Bild 4** sind die vom Netz errechneten Rückfederungswerte gegen die experimentell ermittelten aufgetragen. Aus diesem Diagramm lässt sich eine sehr gute Korrelation der beiden Ergebnisse mit dem Korrelationskoeffizienten R=0,9983 ablesen.

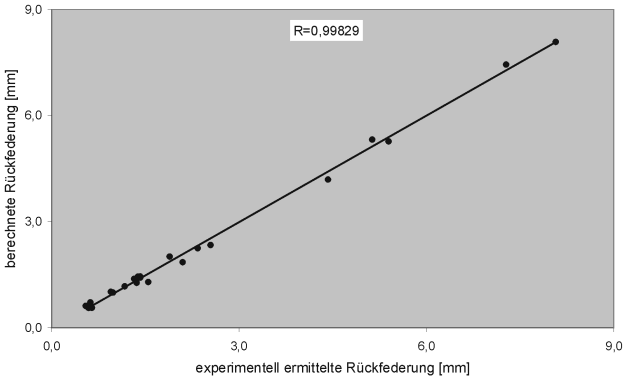

Bild 4: Berechnungsergebnis des BP-M-Netzes im Vergleich zum Experiment

3.2 Backpropagation mit adaptiver Lernrate und Momentum (BP-AM)

Dieses Verfahren unterscheidet sich vom BP-M lediglich durch die hier veränderliche Lernrate. Aus diesem Grund verlief die Untersuchung analog zu der im letzten Abschnitt durchgeführten, so dass die dort erstellten Netzkonfigurationen für den hier betrachteten BP-AM-Algorithmus übernommen wurden. Die Variabilität der Lernrate bringt zwei weitere Lernparameter, den Lernratenerhöhungs- η^+ und Lernratenverringerungsfaktor η^-, mit sich. Die Werte dieser Faktoren wurden für alle Berechnungen mit 1,05 und 0,7 angegeben, da Vorberechnungen mit hiervon abweichenden Werten keine signifikante Verbesserung der Trainingsergebnisse zeigten.

3.2.1 Simulationsergebnis

Der niedrigste Generalisierungsfehler (4,0%), der mit diesem Algorithmus erreicht wurde, ergab sich für ein Netz mit fünf Zwischenschichtneuronen, einer Lernrate von 0,1 und einem Momentum von 0,95. Das error goal wurde hier nach 2500 Iterationsschritten und 25 Sekunden Trainingsdauer erreicht. Die von diesem Netz berechneten Rückfederungen wurden denen nach dem Versuch gemessenen in **Bild 5** gegenübergestellt. Bei diesem Vergleich ergab sich ein Korrelationskoeffizient von 0,9984.

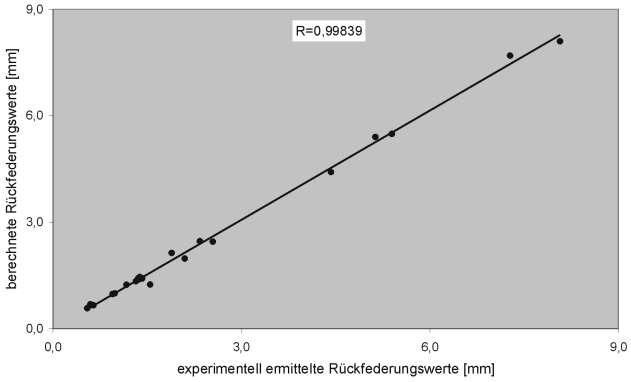

Bild 5: Berechnungsergebnis des BP-AM-Netzes im Vergleich zum Experiment

Bemerkenswert ist bei diesem Algorithmus die im Vergleich zum BP-M schnellere Konvergenz. Dies liegt sicherlich an der adaptiven Lernrate, die sich der Fehleroberfläche während des Lernprozesses anpasst. Die übrigen Feststellungen für das BP-M-Verfahren haben sich auch hier bewahrheitet.

3.3 Resilient Propagation (RProp)

Für die Simulation des Tiefziehprozesses mit Netzen, die den RProp-Algorithmus als Lernregel verwenden, wurden verschiedene Netzkonfigurationen durchgerechnet. Dabei wurde zum einen die Anzahl der verdeckten Schichten mit den darin enthaltenen Neuronen variiert. Für die Berechnung sämtlicher Netzkonfigurationen wurde der Schrittweitenerhöhungsfaktor η^+ mit 1,2, der Schrittweitenverringerungsfaktor η^- mit 0,5, die Schrittweitenoberschranke Δ_{max} mit 50 und die Schrittweitenunterschranke Δ_{min} mit 0,07 angegeben. Die Berechnung des Netzfehlers erfolgte nach dem mittleren quadratischen Fehler (MSE). Es wurde ein error goal von 0,005 vordefiniert. Die maximal durchzuführenden Iterationsschritte wurden mit 10000 festgelegt.

3.3.1 Simulationsergebnis

Die Kombination aller Parameterstufen nach einem vollfaktoriellen Berechnungsplan ergab in diesem Fall 15 unterschiedliche Netzkonfigurationen. Auffällig ist bei diesem Algorithmus seine extrem schnelle Konvergenz bei Zunahme der Neuronenzahl in der verdeckten Schicht. Dabei ist es gleichgültig, über wie viele verdeckte Schichten das Netz verfügt. Anhand der erzielten Generalisierungsfehler lässt sich erkennen, dass eine Zwischenschicht für die vorliegende Aufgabenstellung völlig ausreichend ist. Die Verwendung von mehreren Schichten führte sogar zu erheblich schlechteren Ergebnissen. So wurde das beste Ergebnis (4,3%) mit einem Netz erzielt, das 3 Neuronen in der verdeckten Schicht verwendete. Die Berechnung wurde nach 10000 Iterationsschritten unterbrochen, da das error goal immer noch erreicht war. Netze mit mehr Neuronen in der Zwischenschicht (z.B. 7) unterschritten das error goal bereits nach 155 Iterationen. Die Berechnung dauerte hier lediglich 2 Sekunden. Allerdings war diese kurze Rechendauer mit einer Verschlechterung des Generalisierungsfehlers (6,1%), im Vergleich zum 3-Neuronen-Netz, verbunden.

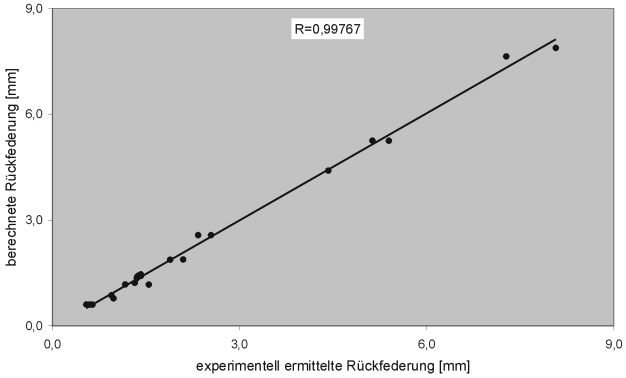

Bild 6: Berechnungsergebnis des RProp-Netzes im Vergleich zum Experiment

Der Vergleich experimentell ermittelter Rückfederungswerte mit denjenigen, die mit dem besten Netz berechnet wurden, ist in **Bild 6** gezeigt. Aus diesem Vergleich ging ein, im Vergleich zu den vorhergehenden Verfahren, niedrigerer Korrelationskoeffizient von 0,99767 hervor.

3.4 Leistungsfähigkeit der Algorithmen

Anhand der Simulationsergebnisse der verwendeten Backpropagation-Algorithmen kann festgestellt werden, dass sich die drei Verfahren zur Abbildung des Rückfederungsverhaltens tiefgezogener Bauteile aus Feinblech gut eignen. Vergleicht man die erzielten Generalisierungsfehler, resultiert, dass sich der BM-AM-Algorithmus für diese Problemstellung die besten Ergebnisse liefert. Dies bezieht sich auch auf die benötigte Trainingsdauer zum Erreichen des vorgegebenen error goals. Die im Vergleich zu den anderen Verfahren beste Leistungsfähigkeit dieses Algorithmus beruht auf der Verwendung einer adaptiven Lernrate, die sich der Fehleroberfläche anpasst. Dadurch wird die Verfahrenseffizienz bei der Suche nach Fehlerminima gesteigert.

4 Zusammenfassung

Die im Rahmen dieser Arbeit eingesetzten neuronalen Netze konnten nach einer maximalen Trainingsdauer von 90 Sekunden eine durchschnittliche Vorhersagegenauigkeit für unbekannte Eingabeparameter von 95% realisieren. In Anbetracht der relativ geringen Anzahl der verwendeten Trainingsmuster auf der einen und der Komplexität des Tiefziehprozesses auf der anderen Seite, kann dieses Ergebnis als sehr gut bezeichnet werden. Dies gilt in besonderem Maße, wenn man die Leistungsfähigkeit der neuronalen Netze mit der der numerischen Methoden vergleicht. Dort werden für ähnliche Fertigungsfälle unter großem Aufwand und mit entsprechendem know how Vorhersagegenauigkeiten weit unter 90% erzielt.

5 Literatur

/1/ Roll, K.; Rohleder, M.: Simulation der Rückfederung in der Blechumformung, zweites Industriekolloquium SFB 362 „Fertigen in Feinblech“, Clausthal-Zellerfeld, 2000

/2/ Doege, E.; Seidel, H.-J.; Schmidt-Jürgensen, R.: Vorhersage der Rückfederung tiefgezogener Blechbauteile mit FAST_FORM3D, 19th CAD-FEM Users´ Meeting 2001, International Congress on FEM Technologie, Berlin, Potsdam, 2001

/3/ Baumgart, H.; Deinzer, G.H.: Neue Werkstoffe in der Automobil-Großseriefertigung, drittes Industriekolloquium SFB 362 „Fertigen in Feinblech“, Clausthal-Zellerfeld, 2002

/4/ Reuter, M.: Medizinische Signalverarbeitung, in: Lehmann, T. M.; Meyer zu Bexten, E.: Handbuch der Medizinischen Informatik, Carl Hanser Verlag, München, Wien, 2002

/5/ Scherer, A.: Neuronale Netze, Grundlagen und Anwendungen, Vieweg Verlag, Braunschweig / Wiesbaden, 1997