Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 82 (1991)

Heft: 13

Artikel: Konzept und Bau einer Lernmaschine für neuronale Netze

Autor: Gunzinger, Anton / Müller, Urs

DOI: https://doi.org/10.5169/seals-902979

Konzept und Bau einer Lernmaschine für neuronale Netze

Anton Gunzinger und Urs Müller

Während heute für den Vorwärtspfad von neuronalen Netzen sehr schnelle Implementationen zur Verfügung stehen, kann das Lernen (Rückwärtspfad) solcher Netze selbst mit sehr leistungsfähigen Rechnern Stunden oder sogar Wochen dauern. In diesem Artikel wird ein mit digitalen Signalprozessoren aufgebautes Mehrprozessorsystem vorgestellt, das die Lernzeit neuronaler Netze um zwei Grössenordnungen, das heisst auf Minuten bis Stunden reduzieren soll.

Alors que pour le trajet en direct de réseaux neuraux on dispose aujourd'hui d'implémentations très rapides, l'apprentissage (trajet en indirect) de tels réseaux peut durer des heures, voire des semaines, même en disposant de calculateurs très performants. Dans cet article est présenté un système multiprocesseur basé sur des processeurs de signaux numériques qui réduit la durée d'apprentissage des réseaux neuraux de deux ordres de grandeur, donc à des durées de minutes à quelques heures.

Adresse der Autoren
Dr. Anton Gunzinger und *Urs Müller*, Dipl. El.-Ing. ETH, Institut für Elektronik, ETH Zentrum, 8092 Zürich

Im Bereich der Mustererkennung, Klassifikation und Regelungstechnik gibt es viele Aufgaben, die sich mit klassischen numerischen Klassifikatoren oder regelbasierten Systemen nur schwer lösen lassen. Die ersteren versagen, weil sie oft nur linear beschränkte Klassifikationsräume zulassen, die zweiten, weil es sehr oft schwierig ist – besonders für natürliche Umgebungen –, Regeln anzugeben. Eine Alternative für diese Anwendungen bieten neuronale Netze. Diese werden im Gegensatz zu klassischen Methoden nicht programmiert, sondern trainiert. Sie sind in der Lage, anhand von vorgezeigten Beispielen (d.h. anhand von Eingangs- und Ausgangsinformation) selbständig Regeln zu finden (und zu verallgemeinern). Ihr Hauptvorteil liegt demnach logischerweise bei Anwendungen mit schwer definierbaren Regeln, wie sie in den Gebieten der Mustererkennung, Signalverarbeitung (Ton- und Bildverarbeitung) und Regelungstechnik auftreten.

Obwohl neuronale Netze in gewissen Anwendungsgebieten entscheidende Vorteile gegenüber klassischen Methoden besitzen, haben sie sich industriell noch nicht durchgesetzt, dies hauptsächlich aus folgenden Gründen:

- Das Trainieren der Netze, die Lernphase, dauert mit den heute verfügbaren Rechnern relativ lange. Dies hat seinen Grund besonders darin, dass zur Optimierung der Netzparameter (Netzgrösse, Verknüpfung) relativ viele Experimente (Simulationen) nötig sind, wobei die einzelnen Simulationen viel Zeit beanspruchen.
- Die heute verwendeten Lernalgorithmen konvergieren relativ langsam, und es kann keine Angabe über die Qualität des gefundenen Optimums angegeben werden.
- Es fehlt die notwendige Erfahrung für die Lösung komplexer Aufgaben mit Hilfe neuronaler Netze.
- Das Vertrauen der Ingenieure in die «ungenauen» neuronalen Netze ist noch nicht vorhanden.

In den meisten Anwendungen werden die neuronalen Netze auf einem herkömmlichen Computer simuliert; dadurch werden sie entsprechend langsam abgearbeitet. Für die relativ einfachen Berechnungen im Vorwärtspfad kann auch auf spezielle, zum Teil analoge Systeme [1] (im Vorwärtspfad genügt deren beschränkte Auflösung) zurückgegriffen werden; der kompliziertere Lernvorgang muss in der Regel simuliert werden. Selbst mit sehr leistungsfähigen Rechnern kann das Lernen eines einzelnen Netzes dabei Stunden oder sogar Wochen dauern [2].

Viele Rechenfunktionen von neuronalen Netzen können grundsätzlich parallel abgearbeitet werden. Diese könnten deshalb auch auf parallelen Rechnern implementiert werden, wobei allerdings, bedingt durch die starke Vernetzung des neuronalen Netzes, die Kommunikation im Mehrprozessorsystem zum Engpass werden kann.

In diesem Artikel wird ein Mehrprozessorsystem vorgestellt, dessen Kommunikation durch spezielle Hardware unterstützt wird. Diese Spezialhardware kann die Daten von den einzelnen Prozessoren autonom einsammeln bzw. unter diesen verteilen. Es wird weiter gezeigt, wie das heute am meisten verwendete neuronale Netz, das Mehrschicht-Perzeptron, auf diese Rechnerarchitektur abgebildet werden kann. Die damit erzielbaren Resultate sind sehr ermutigend; es scheint, dass sich mit diesem System die Rechenzeit beim Lernen

von neuronalen Netzen um eine bis zwei Grössenordnungen reduzieren lässt, womit diese Implementation zu den derzeit schnellsten auf der Welt gehört.

Das Mehrschicht-Perzeptron

Eine Übersicht über die verschiedenen Typen von neuronalen Netzen gibt [3]. Das heute am häufigsten verwendete neuronale Netz ist das Mehrschicht-Perzeptron (Bild 1). Ein Mehrschicht-Perzeptron besteht aus mehreren Schichten; einer Eingangsschicht (Input Layer), meistens einer bis zwei verdeckten Schichten (Hidden Layer) und einer Ausgangsschicht (Output Layer). Jedes Neuron der Schicht n ist mit jedem Neuron der Schicht n-1 verbunden und jeder Verbindung ist ein Gewicht zugeordnet (Bild 1).

Das neuronale Netz arbeitet in zwei Betriebsmodi, im Arbeitsmodus und im Lernmodus. Im Arbeitsmodus (Feed-Foreward) wird der auszuwertende Signalvektor an die Eingangsschicht angelegt (z.B. das Bild eines Buchstabens). Nach der Propagationszeit steht in der Ausgangsschicht das Resultat (z.B. klassifizierter Buchstabe) zur Verfügung, falls das Netz vorher richtig gelernt hat, das heisst falls die einzelnen Gewichte in den Neuronenverbindungen richtig eingestellt wurden.

Für das Trainieren eines neuronalen Netzes muss eine gewisse Anzahl Eingangsvektoren mit den gewünschten zugehörigen Ausgangsvektoren bereitgestellt werden. Die Gewichte im neuronalen Netz müssen nun so modifiziert werden, dass bei vorgegebenem Eingangsvektor der vorgegebene Ausgangsvektor erzeugt wird. Dazu wird an das Netz zuerst ein Eingangsvektor angelegt und daraus ein Ausgangsvektor berechnet. Nun wird dieser berechnete Ausgangsvektor mit dem Soll-Ausgangsvektor verglichen und die Abweichung (Fehler) bestimmt. Dieser Fehler wird jetzt anteilsmässig auf die einzelnen Neuronen umgerechnet; anschliessend werden die Gewichte so verändert, dass der Fehler kleiner wird (Back-Propagation). Durch mehrmaliges Durchlaufen dieses Prozesses wird der Fehler minimalisiert.

Dieses Verhalten eines neuronalen Netzes (inkl. Lernmechanismus) kann mathematisch folgendermassen beschrieben werden [4]: Das Ausgangssignal a_j eines Neurons berechnet sich aus der gewichteten Summe (Gewicht w_{jk}) der Eingangssignale a_k, verzerrt durch eine Nichtlinearität.

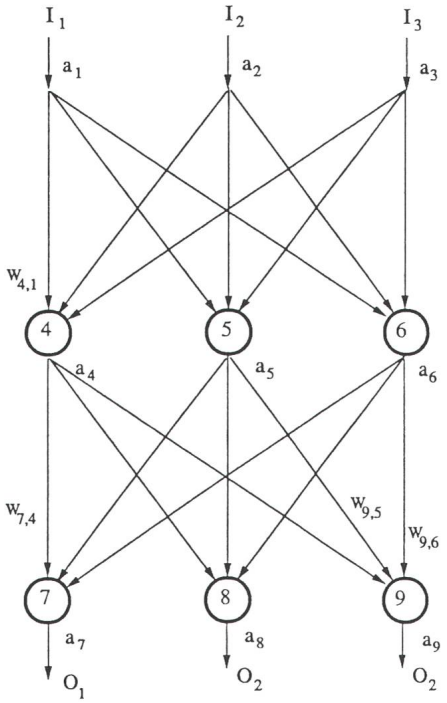

Bild 1 Aufbau eines Mehrschicht-Perzeptrons

$$a_j = f(\sum_k (w_{jk} \cdot a_k)) = f(s_j) \quad (1)$$

Für den Lernvorgang werden eine Anzahl Lernbeispiele (Index^p) mit Eingangsmuster $\boldsymbol{i}^p$ und den entsprechenden Ausgangsvektoren $\boldsymbol{t}^p$ benötigt. Im ersten Schritt lässt man das Netz für ein bestimmtes Eingangsmuster einen eigenen Ausgangsvektor $\boldsymbol{o}^p$ produzieren. Dann wird der mit der Steilheit der Nichtlinearität (im Arbeitspunkt des jeweiligen Neuronausgangssignals) gewichtete Fehler des Netzwerkes in den Ausgangsneuronen (Gl. 2) und ebenso in den Neuronen der anderen Schichten (Gl. 3) bestimmt. Eine Herleitung des Verfahrens findet sich in [4].

$$\delta_j^p = f'(s_j^p) \cdot (t_j^p - o_j^p) \quad (2)$$

$$\delta_j^p = f'(s_j^p) \cdot \sum_k (\delta_k^p \cdot w_{kj}) \quad (3)$$

$$\Delta w_{ji}^p = \eta \cdot \delta_j^p \cdot a_i^p \quad (4)$$

Aus dem Fehler, dem konstanten Lernfaktor η und der Neuronenaktivität a_j wird für jedes Gewicht ein Korrekturwert (Gl. 4) berechnet und anschliessend zum vorhandenen Gewicht addiert (Gl. 5).

$$w_{ji}^{p+1} = w_{ji}^p + \Delta w_{ji}^p \quad (5)$$

Dieser Vorgang wird für jede Präsentation eines Lernbeispiels wiederholt. Um die Verallgemeinerungsqualität zu bestimmen, wird oft ein Teil der Beispiele nicht trainiert. Mit diesen Beispielen können die Lernfortschritte des Netzes gemessen werden. Als Nichtlinearität wird oft die sogenannte Sigmoidfunktion (Gl. 6) verwendet. Diese begrenzt das Ausgangssignal auf den Wertebereich [0...1].

$$f(x) = \frac{1}{1 + e^{-x}} \quad (6)$$

Die Ableitung der Sigmoidfunktion an der Stelle x ist:

$$f'(x) = f(x) \cdot (1 - f(x)) \quad (7)$$

Architektur des Mehrprozessorsystems

Heute kann man auf dem Markt sehr leistungsfähige und kostengünstige Mikroprozessoren erhalten. So sind zum Beispiel Signalprozessoren mit bis zu 50 Millionen Gleitkommainstruktionen pro Sekunde für wenige 100 $ pro Stück zu haben. Durch geeignete Zusammenschaltung von 30 bis 100 solcher Prozessoren mittels spezieller Zusatzschaltungen zur Kommunikation kann ein Rechnersystem realisiert werden, das mit seiner Rechenleistung für eine Palette von Algorithmen durchaus mit jener konventioneller Supercomputer konkurrieren kann, aber um den Faktor 100...1000 weniger kostet. Voraussetzung dafür ist allerdings, dass im System genügend Kommunikationsleistung vorhanden ist und dass das System von einem Benützer leicht programmiert werden kann. Am Institut für Elektronik der ETH-Zürich befindet sich ein solches Mehrprozessorsystem im Aufbau. Das Mehrprozessorsystem hat den Namen MUSIC (Multi Signalprozessor System with Intelligent Communication) erhalten. Es soll als Low-Cost Supercomputer zur Signalverarbeitung, zur Simulation in Physik und Chemie und für neuronale Netze eingesetzt werden.

Jedes einzelne Prozessorelement (PE) ist mit einem digitalen Signalprozessor bestückt und verfügt über eine

maximale Rechenleistung von 50 MFlops (Million Floatingpoint Operations per Second). Es sind Systeme mit bis zu 60 Prozessorelementen geplant; damit kann eine Spitzenleistung von 3 GFlops erreicht werden. Die einzelnen Prozessorelemente sind über ein Hochgeschwindigkeitsnetzwerk (Ring) miteinander verbunden. Das Netzwerk verfügt über «verteilte Intelligenz» und ist in der Lage, Daten autonom von den einzelnen PEs einzusammeln und wieder zu verteilen (Bild 2).

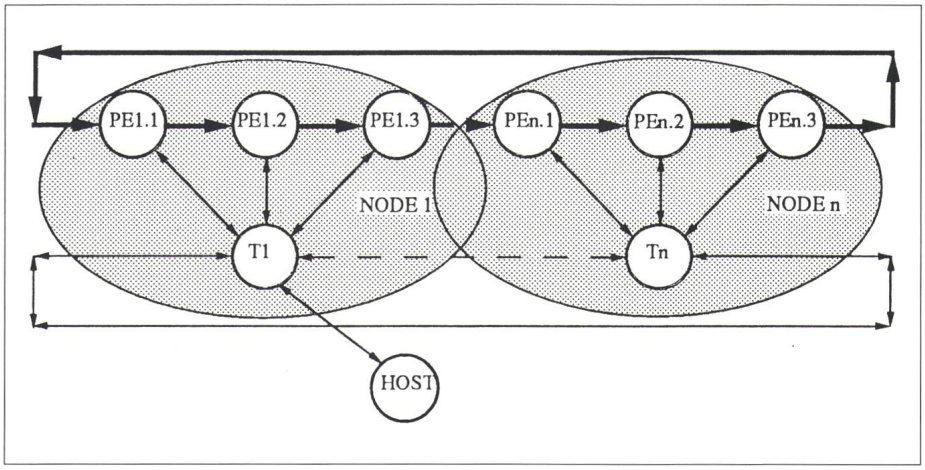

Bild 2 Architektur des Music-Systems

Das Bild 3 zeigt das Blockschaltbild einer Music-Karte. Ein mit einem Transputer realisierter Knotenmanager steuert eine Gruppe von PEs (in diesem Fall 3). Als Recheneinheit wird der digitale Signalprozessor DSP 96002 von Motorola mit 50 MFlops Spitzen-Rechenleistung eingesetzt. Jeder Signalprozessor ist mit einem schnellen statischen Speicher (SRAM) von 128 kByte als Programm- und einem dynamischen Videoschieberegister-Speicher (VRAM) von 2 bzw. 8 MByte für Datenspeicher ausgerüstet. Bei den dynamischen Speichern handelt es sich um sogenannte Video-DRAMs. Diese verfügen über ein integriertes Hochgeschwindigkeits-Schieberegister zum schnellen I/O-Verkehr. Die Daten können mit einer Taktrate von bis zu 20 MHz vom Schieberegister übernommen bzw. ausgegeben werden. Die Kommunikation mit dem Hochgeschwindigkeitsnetzwerk des Mehrprozessorsystems erfolgt über diese Videoschieberegister. Als Netzwerkkontroller wird ein programmierbares Gatearray, ein sogenanntes LCA (Logic Cell Array) von Xilinx eingesetzt. Der Netzwerkkontroller nimmt die Daten aus dem in der Kette vorangehenden PE auf (Ringarchitektur), speichert sie wenn nötig im Video-DRAM ab bzw. ersetzt sie durch Daten aus dem Video-DRAM und gibt sie anschliessend an den nächsten Netzwerkkontroller weiter. Der Datenbus ist 40 Bit breit: 32 Bit Daten und 8 Bit Zusatzinformation (Absenderadresse, Gültigkeit des Datenwerts). Die maximale Datenrate soll im Endausbau 20 MHz betragen.

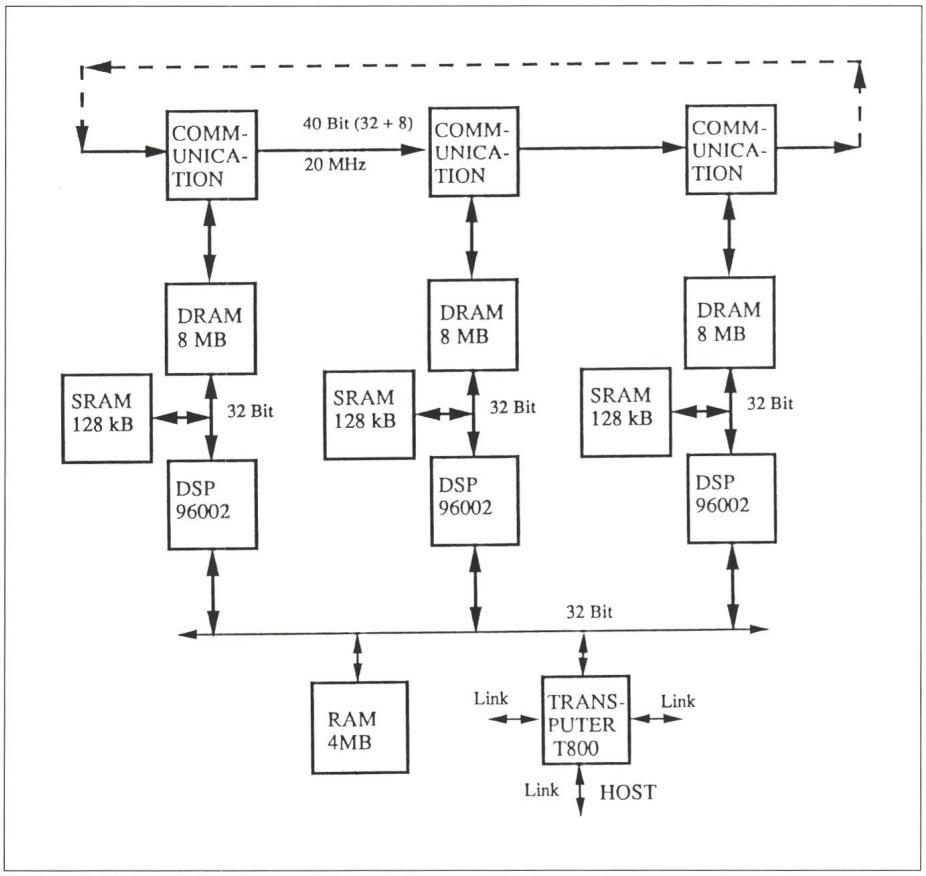

Bild 3 Blockschaltbild einer Music-Karte

Das zentrale Problem in einem Mehrprozessorsystem ist die Verteilung der Arbeit auf die einzelnen Prozessor- (PE) und Kommunikationselemente. Beim Music-Konzept wird als Grundoperation angenommen, dass ein bestimmter Eingabe-Datensatz sich durch einen (komplexen) Algorithmus eindeutig in einen neuen Ausgabe-Datensatz transformieren lässt (Bild 4). Damit kann jedem Prozessor die Aufgabe übertragen werden, einen Teil der Ausgabedaten zu berechnen. Die Daten müssen dann nur noch eingesammelt und bei einer iterativen Anwendung wieder an die einzelnen Prozessoren verteilt werden. Diese Kommunikationsaufgabe wird durch das Spezialnetzwerk des Music-Systems autonom, mit der maximalen Datenrate von 20 MHz ausgeführt. Da es sich bei den verwendeten Datensätzen meist um 1-, 2- oder 3-dimensionale Arrays handelt, werden diese Datenstrukturen durch das Netzwerk besonders unterstützt. In der Regel wechseln sich Berechnungs- und Kommunikationsphase ab (Bild 5). Bei einigen Anwendungen kann mit der Kommunikation bereits während der Berechnungsphase begonnen werden.

Damit das System seine maximale Rechenleistung erreichen kann, müs-

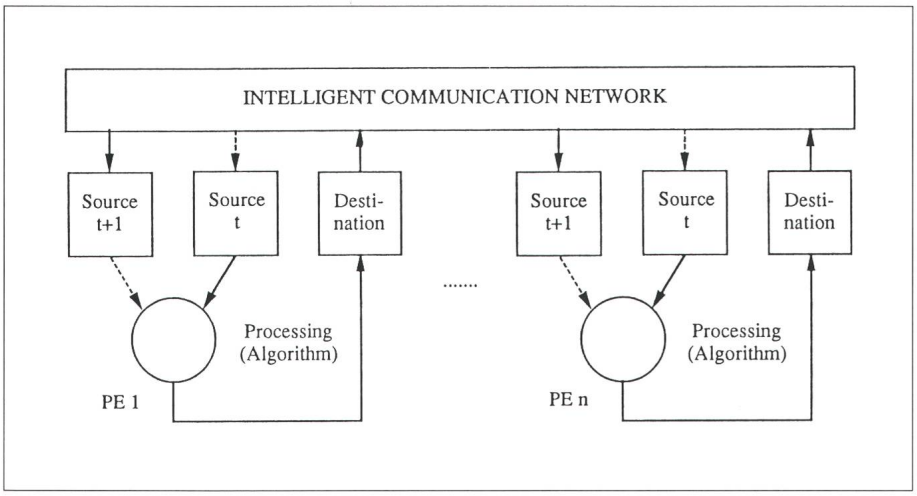

Bild 4 Funktionsweise
Aus einem Eingabedatensatz werden durch die einzelnen PEs Teile des Ausgabedatensatzes berechnet; diese Teildatensätze werden eingesammelt und stehen für die nächste Iteration wieder als Eingabedatensätze zur Verfügung

sen die einzelnen Prozessoren gleichmässig ausgelastet sein. Für viele der vorgesehenen Anwendungen ist die Rechenzeit datenunabhängig; damit kann ein Lastausgleich in der Regel durch eine gleichmässige Verteilung des Datensatzes auf die einzelnen PEs erfolgen. Im Falle datenabhängiger Rechenzeit kann die Grösse des Datensatzes bei iterativen Verfahren anhand der Rechenzeit in der vorausgehenden Iteration dynamisch (durch den Knotenmanager) angepasst werden. Damit kann ein sehr guter Lastausgleich erreicht werden [5].

Der Speed-up-Faktor gibt den Rechengeschwindigkeitsgewinn in einem Mehrprozessorsystem als Funktion der Anzahl der verwendeten PEs an. Es kann gezeigt werden, dass sich bei einem Komplexitätsfaktor von 1000 (pro 1000 Instruktionen wird ein neuer Datenwert produziert und über das Kommunikationsnetzwerk übermittelt) und bei 40 PEs ein Speed-up von 38,5 und bei 100 PEs von 90,91 erreicht werden kann [6]. In der Tabelle I sind die Eckdaten des Music-Systems zusammengefasst. Ein System mit 20 Karten findet in einem 19′-Gehäuse Platz, ein Hochleistungsrechner für den Bürotisch.

	1 PE	1 Board	10 Boards	20 Boards
PEs	1	3	30	60
DSP 96002	1	3	30	60
Peak Performance (MFLOPS)	50	150	1500	3000
SRAM (kB)	128	384	3840	7680
DRAM (MB)	8	24	240	480
Transputer	–	1	10	20
T-DRAM (MB)	–	4	40	80
Power Consummation (W)	10	30	300	600

Tabelle I Wichtigste Daten des Music-Systems

Um die Leistungsfähigkeit des Music-Systems mit anderen Rechnern zu vergleichen, wurde die Berechnung von Fraktalen implementiert. Die Rechenzeit für die Grundfigur mit einer Auflösung von 256 mal 256 Bildpunkten (max. 256 Iterationen pro Bildpunkt und 32 Bit Gleitkommaarithmetik) sind in der Tabelle II für verschiedene Rechner zusammengestellt. Die Rechenleistung für das Music-10-System wurde dabei geschätzt, da im Moment noch nicht genügend Karten zur Verfügung stehen. Die mit dem Music-System erreichbaren Rechenleistungen sind beachtlich.

Parallele Implementation neuronaler Netze

Im folgenden wird gezeigt, wie das Mehrschicht-Perzeptron auf das Music-System abgebildet werden kann. Der rechenaufwendigste Teil für das Mehrschicht-Perzeptron ist die Summation des Produktes aus Aktivität und Gewicht für den Vorwärtspfad sowie die Summation des Produktes von Fehler und Gewicht im Rückwärtspfad. Ein Beispiel aus der Bildverarbeitung: Der Eingabevektor bestehe aus den Werten von 256 Bildpunkten, und in der ersten Schicht seien 512

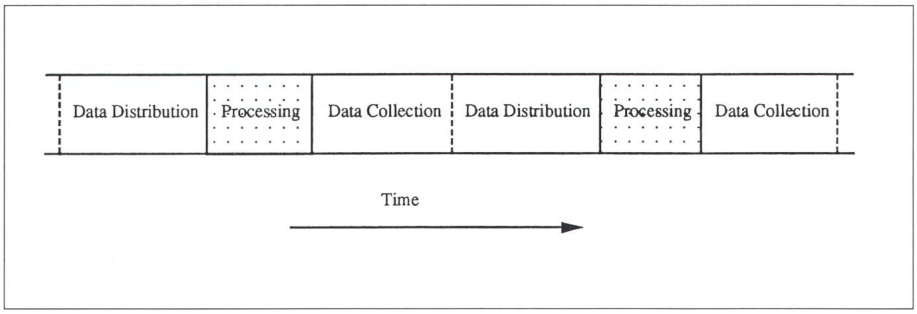

Bild 5 Berechnungs- und Kommunikationsphase wechseln sich im Music-System ab

Neuronen vorhanden. Dies bedeutet, dass rund 130 000 Multiplikationen/Additionen ausgeführt werden müssen. Die nichtlineare Funktion wird mit 512 Berechnungen vergleichsweise wenige Male ausgewertet. Glücklicherweise eignen sich Signalprozessoren besonders für die Summation von Produkten, da diese Operation auch vielen Signalverarbeitungsalgorithmen wie Filterung, Vektormultiplikation oder Fouriertransformation zugrunde liegt.

Die Verteilung des neuronalen Netzes auf dem Music-System wird im folgenden anhand eines Beispiels gezeigt. Der Einfachheit halber sei ein neuronales Netz mit 3 Eingängen, 3 Neuronen in der versteckten Schicht und 3 Ausgangsneuronen gegeben.

Dieses Netz soll jetzt auf 3 PEs verteilt werden. In praktischen Anwendungen können natürlich viel mehr Neuronen und PEs vorhanden sein, wobei die Anzahl der Neuronen und PEs unabhängig voneinander sind. Die grösste Effizienz wird erreicht, wenn alle PEs etwa gleichviel Neuronen behandeln müssen.

Zur Berechnung des Vorwärtspfades gilt, dass die Aktivitäten der einzelnen Neuronen (Gl. 1) innerhalb der Schicht n parallel ausgerechnet werden können, sobald die Aktivitäten der Schicht n-1 bekannt sind. Nach der Berechnung der Aktivität der Schicht n müssen die Daten $\boldsymbol{a}^p$ eingesammelt und auf alle PEs kopiert werden; diese Daten werden in allen PEs abgespeichert, da sie für die Berechnung der Aktivität der nächsten Schicht im Vorwärtspfad und für das Lernen benötigt werden. In Bild 6 wurde dieser Sachverhalt grafisch dargestellt.

Analog zum Vorwärtspfad erfolgt die Berechnung für den Rückwärtspfad, wobei jetzt das Fehlersignal δ^p (Gl. 2 und 3) übertragen wird (Bild 6). Das Fehlersignal δ^p wird ebenfalls in allen PEs abgespeichert. Nun muss nur noch das «Lernen» erfolgen, das heisst die Gewichte der einzelnen Neuronen müssen nach (Gl. 4 und 5) aufdatiert werden. Dabei ergibt sich aber ein Problem: die aufdatierten Gewichte w_{ji}^{p+1} sind in den meisten Fällen nicht auf den PEs verfügbar, auf denen sie im Vorwärtspfad gebraucht werden. Beispiel: das Gewicht $w_{8,4}$ wird auf PE1 aufdatiert (Bild 6b) und für den Vorwärtspfad in PE2 benötigt (Bild 6a). Um dieses Problem zu beheben, könnten die neuen Gewichte nach dem Lernen über das Kommunikationsnetzwerk eingesammelt und verteilt werden; da aber die Anzahl der Gewichte mit $O(n^2)$ wächst (bei einer Schicht mit n Eingängen und n Neuronen), würde auch die Kommunikation mit $O(n^2)$ anwachsen und bald wäre eine Sättigung des Kommunikationsnetzwerkes (für das Music-System maximal 20 Millionen Verbindungen pro Sekunde) erreicht. Damit wäre keine weitere Steigerung der Leistungsfähigkeit des Systems durch zusätzliche PEs mehr möglich.

Wie kann dieser gravierende Nachteil behoben werden? Yoon schlägt vor [7], dass die Anpassung der Gewichte gleichzeitig auf zwei verschiedenen PEs erfolgen soll: einmal für den Vorwärtspfad und einmal für den

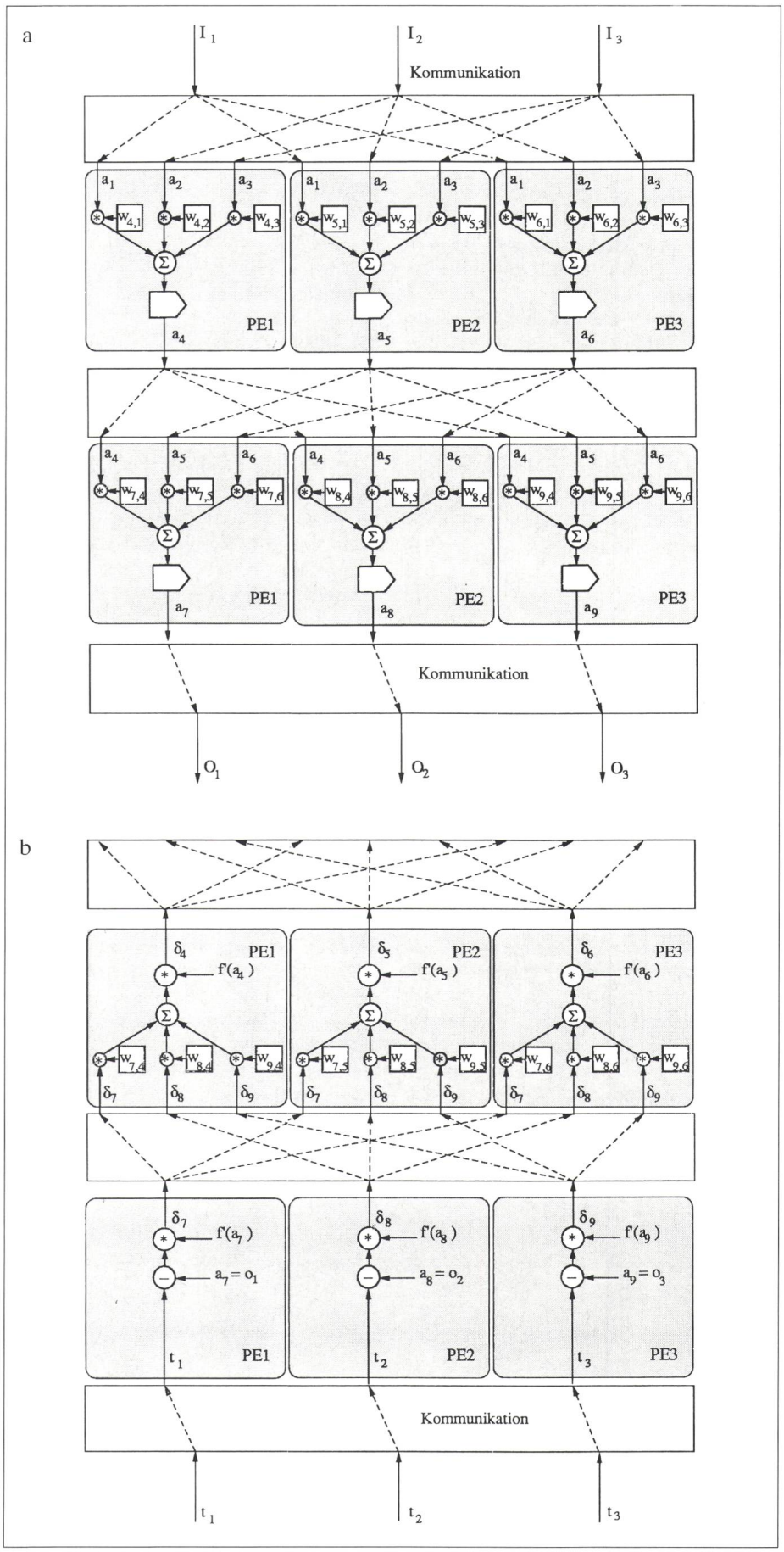

Bild 6a, 6b **Partitionierung des neuronalen Netzes**
a Vorwärtspfad
b Rückwärtspfad

Rückwärtspfad. Zur Anpassung der Gewichte werden der Lernfaktor, die Aktivität eines Neurons, der Fehler und der alte Gewichtswert benötigt. Die Aktivität und die Fehler der Neuronen sind bereits auf allen PEs abgespeichert. Wenn die Aufdatierung der Gewichte auf unterschiedlichen PEs erfolgt, muss dafür gesorgt werden, dass die sich entsprechenden Gewichte mit dem gleichen Startwert initialisiert werden.

Durch diese zusätzlichen Berechnungen (Aufdatieren der Gewichte auf mehreren PEs) kann die Kommunikation von der Ordnung $O(2n+n^2)$ auf $O(2n)$ reduziert werden. Damit tritt eine Sättigung der Systemsleistung infolge der begrenzten Kommunikationsleistung erst viel später auf (hier bei maximal 10^{12} Millionen Verbindungen pro Sekunde).

Computer	Feed Foreward	Back Propagation
MAC II	30 kCPS	
IBM 386	30 kCPS	
SUN-3	30 kCPS	
TMS320C25	5 MCPS	0.4 BMCPS
DSP96'002	8 MCPS	2.8 MBCPS
Delta II	11 MCPS	2.7 MBCPS
WARP		20 MBCPS
IBM 3090		30 MBCPS
Music-1 (3 DSP)	24 MCPS	6 MBCPS
Music-10 (30 DSP)	240 MCPS	60 MBCPS

Tabelle III Geschätzte Zieldaten für das Music-System im Anwendungsbereich neuronaler Netze

Resultate, Ausblick

Eine Karte mit drei Signalprozessoren und einem Transputer als Knotenmanager des für die Simulation des neuronalen Netzes verwendeten Music-Systems wurde in Betrieb genommen. Der Lernalgorithmus für neuronale Netze ist auf dem Simulator funktionsfähig, er wird im Moment auf das Music-System übertragen.

Die Tabelle III zeigt die abgeschätzten Zieldaten für das Music-System im Anwendungsbereich neuronaler Netze. In diesem Anwendungsgebiet wird die Leistungsfähigkeit eines Rechners in Anzahl Verbindungen pro Sekunde (Connections per Second im Vorwärtspfad bzw. Backward-Propogation Connections per Second für den Lernmodus) angegeben. Die Werte für das Music-System sind hochgerechnet. Sie sollten praktisch verifiziert werden können, sobald weitere Karten betriebsbereit sind. Wenn diese Daten erreicht werden, wird die hier vorgestellte Lernmaschine für neuronale Netze zu den weltweit schnellsten Implementationen gehören, und dies bei mässigem Arbeits- und Materialaufwand.

Einige weiterführende Arbeiten sind geplant: Während die Unterteilung des neuronalen Netzes und seine Zuordnung an die einzelnen Prozessorelemente heute noch manuell erfolgen muss, soll diese Aufgabe in Zukunft von einem Codegenerator übernommen werden. Dieser Codegenerator soll auch für die Implementation von nicht vollständig verknüpften Mehrschicht-Perzeptrons verwendet werden können, wie sie zum Beispiel zum Erkennen handgeschriebener Schriftzeichen verwendet werden [2;8]. Schliesslich erhoffen wir uns, dass dank der hohen Lerngeschwindigkeit des Rechners grundlegend neue Erkenntnisse auf den Gebieten Entwurf, Lernen und Anwendung von neuronalen Netzen erarbeitet werden können. Falls dieses Ziel erreicht wird, könnte dieses Projekt ein entscheidender Schritt in Richtung industrieller Nutzung neuronaler Netze sein.

Language	Processor	Time (s)	Performance (MFlops)
Pascal	80286/287	780	0,0437
Occam	T800	22	1,5
C	TMS 320C30	5	6,8
Assembler	TMS 320C30	2,5	13,6
Assembler	DSP 96002	2	17
Assembler	Music-1 (3 DSP)	0,7	49
Assembler	Music-10 (30 DSP)	0,07	490

Tabelle II Erwarteter Leistungsvergleich mit anderen Rechnern
Berechnungsbeispiel für Fraktale: Die Rechenzeiten für die Berechnung der Grundfigur mit 256 mal 256 Bildpunkten, maximal 256 Iterationen pro Bildpunkt und 32 Bit Gleitkommaarithmetik, sind dargestellt

Dank

An dieser Stelle möchten wir dem Vorsteher des Instituts für Elektronik, Prof. Dr. W. Guggenbühl, für seine Unterstützung dieses Projekts herzlich danken. Danken möchten wir auch den Studenten U.M. Franz, H. von der Mühll und H. Walther für den Bau und die Inbetriebnahme des Music-Systems im Rahmen ihrer Diplomarbeit.

Literatur

[1] *H.P. Graf, L.D. Jackel, W.E. Hubbard:* VLSI Implementation of a Neural Network Model, IEEE-Comp. March 1988, p. 41...49.
[2] *B. Zehnder, G. Crelier:* Schnelle Zeichenerkennung mit neuronalem Netz auf einem Signalprozessor, Diplomarbeit am Institut für Elektronik, ETH-Zürich, 1991.
[3] *J.-F.Leber, M.B.Matthews:* Neuronale Netzwerke: eine Übersicht. Bull. SEV/VSE 80 (1989)15.
[4] *D.E. Rummelhart, G.E. Hinton, R.J. Williams:* Learning Internal Representation by Error Propagation in Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1, Bradford Books, Cambridge 1986.
[5] *J.E. Boillat, P.G.Kropf:* A Fast Distriputed Mapping Algorithm, Proc. on Conpar 90.
[6] *A. Gunzinger:* Architecture and Realization on a Multi Signalprocessor System. To be published in Proc. on the Int. Conf. on Digital Signal Processing, Sept 4-6, 1991, Florence, Elsevier, Amsterdam 1991.
[7] *H. Yoon, J.H. Nang, S.R. Maeng:* Parallel Simulation of Multilayered Networks on Distributed-Memory Multiprocessors, Microprocessing and Microprogramming 29(1990), pp. 185...195, North Holland.
[8] *Y. Le Cun:* Generalization and Network Design Strategies, in Pfeiffer et al. (Editors), Connection in Perspective, Elsevier, Amsterdam 1989.