Massiv parallele Simulation von neuronalen Netzwerken auf der MasPar MP-1

Niels Mache
Universität Stuttgart
Institut für Parallele und Verteilte Höchstleistungsrechner
Breitwiesenstr. 20–22, D–7000 Stuttgart 80, Fed. Rep. of Germany
E–mail: mache@agamemnon.informatik.uni-stuttgart.de

21. September 1992

Zusammenfassung

Die Simulation und speziell das Training von neuronalen Netzwerken stellt mit zunehmender Netzwerkgröße hohe Anforderungen an die Leistungsfähigkeit der verwendeten Computerhardware. Der hier vorgestellte Simulator ermöglicht die Propagierung und das Training von mehrstufigen feedforward-Netzwerken auf der MasPar MP-1216A, einem parallelen SIMD Computer mit 16384 Prozessoren. Diese parallele Implementierung der feedforward-Netzwerke erreicht bei 3-schichtigen Netzwerken Propagierungsraten von bis zu 348 Millionen CPS und Trainingsraten von 129 Millionen CUPS mit dem Backpropagation Verfahren. In diesem Papier werden die der Implementierung zugrundeliegenden Algorithmen und die Schwierigkeiten, die sich aufgrund der hohen Propagierungsrate ergeben, beschrieben.

1 Einleitung

Am IPVR[1] wurde ein effizienter und portabler Simulator für neuronale Netzwerke, der SNNS[2], entwickelt. Dieser Netzwerksimulator ist auf Unix Workstations lauffähig und ermöglicht die Generierung, das Training und die grafische Darstellung von neuronalen Netzwerken. Der erprobte und vielfach genutzte SNNS wurde nun durch einen parallelen Simulatorkern für mehrschichtige feedforward-Netzwerke erweitert. Dem Anwender steht damit ein vielseitiges Werkzeug zur parallelen Simulation von neuronalen Netzwerken zur Verfügung. Der parallele SNNS ist modular aus einer grafischen Benutzeroberfläche, einem sequentiellen Simulator und aus dem parallelen Simulatorkern aufgebaut. Der hier beschriebene parallele Simulatorkern besteht analog zum sequentiellen Kern aus einem funktionalen Interface, internen Prozeduren zur Verwaltung des Netzwerks und benutzerdefinierbaren Funktionen. Zu den benutzerdefinierbaren Funktionen gehören die Aktivierungs-, Lern-

[1] Institut für Parallele und Verteilte Höchstleistungsrechner
[2] Stuttgarter Neuronale Netze Simulator

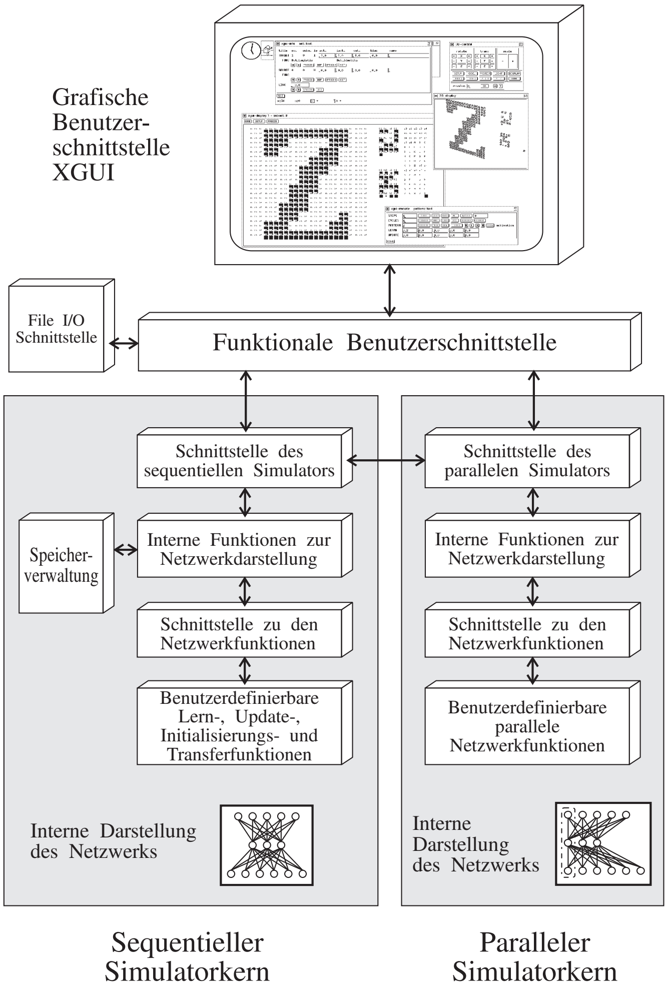

Abbildung 1: Die Struktur des parallelen SNNS bestehend aus der grafischen Benutzeroberfläche, dem sequentiellen und dem parallelen Simulatorkern.

und Initialisierungsfunktionen des Netzwerks. Der Simulatorkern ist auch als Stand-Alone Anwendung lauffähig, d.h. ohne grafisches Benutzerinterface, und kann in andere Applikationen eingebunden werden. In Abbildung 1 sind die einzelnen Komponenten des SNNS dargestellt.

2 Die Architektur der MasPar MP-1

Die MasPar MP-1 ist ein nach dem SIMD (Single Instruction, Multiple Data) Prinzip arbeitendes, massiv paralleles Computersystem. Die hier benutzte MasPar MP-1216A verfügt über 16384 Prozessoren, die in einem zweidimensionalen 128×128 Array angeordnet sind. Zur Kommunikation zwischen den Prozessorelementen (PEs) stehen zwei verschiedene Kommunikationsmechanismen zur Verfügung: einen 3-stufigen Router der bis zu 1024 gleichzeitige Verbindungen zwischen beliebigen Prozessorelementen erlaubt und einem toroidal verbundenen zweidimensionalen Gitter (X-Netz) zur direkten Kommunikation zwischen benachbarten Prozessorelementen. Die Bandbreite des X-Netzes liegt mit maximal 24 GBit/s deutlich über der des Routers mit max. 1,5 GBit/s. Die Steuerung der Prozessorelemente übernimmt die Array Control Unit (ACU). Das Unix Subsystem, eine DEC-Station, stellt die Programmierumgebung (MPPE) und Dienstleistungsprogramme zur Verfügung. Die Programmierung der MasPar MP-1 kann wahlweise in der Programmiersprache MPL (MasPar Parallel Application Language) oder in Fortran erfolgen. Für die Implementierung des parallelen Simulators wurde die Programmiersprache MPL gewählt. Die Sprache MPL basiert auf der Programmiersprache C und ermöglicht die effiziente und flexible Umsetzung von parallelen Algorithmen. Programme für die MasPar MP-1 bestehen üblicherweise aus einem parallelen und einem sequentiellen Teil. Der parallele, in MPL implementierte Teil wird auf der ACU der MasPar ausgeführt und die Daten in SIMD-Manier vom Prozessorarray verarbeitet, den sequentiellen Teil des Programms führt das Unix Subsystem aus. Die sequentiellen und parallelen Programmteile werden unabhängig voneinander abgearbeitet und können bei Bedarf synchronisiert werden. Der Datenaustausch zwischen der ACU bzw. dem Prozessorarray der MasPar und dem Subsystem geschieht mittels Transferfunktionen über den Bus des Subsystems. Die Leistung der MasPar MP-1 ist, wie die aller Parallelrechner, sehr von dem Kommunikationsaufwand der verwendeten Algorithmen abhängig und beträgt in günstigen Fällen (ohne Kommunikation der Prozessorelemente und mit Pipelining der Speicherzugriffe) etwa 1200 MFLOPS bei 32-Bit Genauigkeit.

3 Backpropagation auf der MasPar MP-1

Beim Backpropagation-Verfahren auf der MasPar repräsentiert ein Prozessorelement alle Hidden und Output Units in einer „Spalte“ des Netzwerkes (siehe Abb. 2), d.h. die Darstellung des Netzwerkes durch die Prozessoren ist „1-dimensional“ und die Berechnung der Units erfolgt für jeweils eine Schicht des Netzwerkes parallel. Die 1-dimensionale Netzwerkdarstellung durch die Prozessoren erleichtert die gleichzeitige Berechnung mehrerer Netzwerke, d.h. die parallele Berechnung verschiedener Trainingssätze im 2-dimensionalen Prozessorarray. Zur Darstellung eines Netzwerkes wird die Anzahl der Prozessoren benötigt, die der maximalen Anzahl der Neuronen in einer Schicht, mit Ausnahme der Eingabeschicht, entspricht, d.h. die „Breite“ des Netzwerkes bestimmt die Menge der Prozessoren. Falls die Eingabeschicht des Netzwerkes genau die Schicht ist, die am meisten Neuronen enthält, übernimmt ein zusätzlicher Prozessor die Speicherung der Eingabemuster. Die Anzahl der benötigten Prozessoren N_{procs} ist bei gegebener Anzahl $N_{nodes}(k)$ der Neuronen in der Ebene k eines Netzwerkes mit N_{layers}-Ebenen für $k = 0, \ldots, N_{layers} - 1$:

$$N_{procs} = \begin{cases} \max N_{nodes}(k) & \text{falls } N_{nodes}(0) \leq \max N_{nodes}(k) \\ & \text{für } k = 1, \ldots, N_{layers} - 1, \\ \max N_{nodes}(k) + 1 & \text{sonst.} \end{cases}$$

Jeder Prozessor speichert lokal die synaptischen Gewichte der Eingangsverbindungen der Neuronen, die einer „Spalte“ des Netzwerkes entsprechen. Jeder Prozessor verfügt über eine „Ablage“ und einen Akkumulator zur Speicherung von Zwischenergebnissen und eine bei vollverbundenen Netzwerken unidirektionale, ansonsten bidirektionale Kommunikationsverbindung mit dem linkem Nachbarprozessor. Während der Vorwärts- und Rückwärtspropagierung werden die in den „Ablagen" gespeicherten Zwischenwerte nach jedem Berechnungsschritt zyklisch um eine Stelle nach links verschoben. Die Anzahl der „Ablagen", die zur Berechnung einer Schicht des Netzwerkes benötigt werden, ist so groß wie das Maximum aus der Anzahl der Neuronen in der Vorgänger- und in der zu berechnenden Schicht. Ein Sonderfall tritt ein, wenn die Eingabeschicht mehr Neuronen enthält als die übrigen Schichten des Netzwerkes, d.h. wenn die Anzahl der benötigten „Ablagen" größer ist als die Anzahl der zur Darstellung des Netzwerkes verwendeten Prozessorelemente. In diesem Fall speichert bei der Vorwärtspropagierung ein zusätzlicher Prozessor das aktuelle Eingabemuster der Eingabeschicht und übernimmt damit die Funktion der fehlenden „Ablagen". Für die Anzahl der „Ablagen" $N_{tray}(k)$, die zur Berechnung der ersten Schicht des Netzwerkes benötigt werden, gilt:

$$N_{tray}(1) = \begin{cases} N_{nodes}(1) + 1 \equiv N_{procs} & \text{falls } N_{nodes}(0) > \max N_{nodes}(k) \\ & \text{für alle } k \in [0, N_{layers} - 1], \\ \max\left(N_{nodes}(0), N_{nodes}(1)\right) & \text{sonst.} \end{cases}$$

Für alle anderen Schichten $k = 2, \ldots, N_{layers} - 1$ des Netzwerkes gilt:

$$N_{tray}(k) = \begin{cases} N_{nodes}(k-1) & \text{falls } N_{nodes}(k-1) > N_{nodes}(k), \\ N_{nodes}(k) & \text{sonst.} \end{cases}$$

Die Abbildung 3 stellt die Vorwärtspropagierung in der ersten Netzwerkschicht schematisch dar. Die Anordnung der Gewichte im lokalen Speicher der Prozessoren ist nicht immer um eine Stelle verschoben, sondern hängt davon ab, ob die Anzahl der Neuronen in der zu berechnenden Schicht größer als in der Vorgängerschicht ist oder nicht. Für den Fall, daß die Anzahl der Neuronen in der Schicht $k-1$ größer oder gleich der Anzahl der Neuronen in der Schicht k ist, d.h. wenn $N_{nodes}(k-1) >= N_{nodes}(k)$ gilt, so ist die Anordnung der Gewichte im lokalen Speicher der Prozessoren im Vergleich zu der Anordnung im linken Nachbarprozessor um jeweils eine Stelle nach „unten" verschoben [3], d.h. die Verbindungsgewichte werden mit dem Skew-Faktor 1 gespeichert. Ist die Anzahl der Neuronen in der Vorgängerschicht $k-1$ jedoch kleiner als die Anzahl der Neuronen in der Schicht k ($N_{nodes}(k-1) < N_{nodes}(k)$), so bedeutet dies, daß zur Berechnung der Schicht k mehr „Ablagen" (und Prozessorelemente) verwendet werden als es Neuronen in der Vorgängerschicht $k-1$ gibt. Die „Ablagen" der überzähligen Prozessoren werden dann mit einem besonderen Floating-Point Wert (*NAN (Not A Number)* oder *Maxfloat*) gefüllt. Dieser Wert dient hier dazu, die Prozessorelemente zyklisch an- und auszuschalten: ein Prozessorelement beteiligt sich nur dann an einer Berechnung, wenn seine „Ablage" nicht den speziellen Gleitkommawert enthält. In den Abbildungen ist dieser Wert durch einen Strich in der Ablage dargestellt. Die Verbindungsgewichte der Eingangsverbindungen werden für die Prozessoren $1, \ldots, N_{nodes}(k-1)$ mit dem Skew-Faktor 1 gespeichert. Die übrigen an der Berechnung beteiligten Prozessoren $N_{nodes}(k-1)+1, \ldots, N_{nodes}(k)$ speichern die Verbindungsgewichte in der normalen Reihenfolge. Durch die verschobene Speicherung der Verbindungsgewichte ist die Berechnung von Wx (W ist die Gewichtsmatrix und x der Eingabevektor) in der Arbeitsphase und die Berechnung von $W^T x$ bei der Fehlerpropagierung ohne den Austausch der Verbindungsgewichte möglich. Die Funktion des parallelen Backpropagation-Verfahrens

[3] Bei einer totalen Vernetzung der Units mit ihren Vorgängerunits.

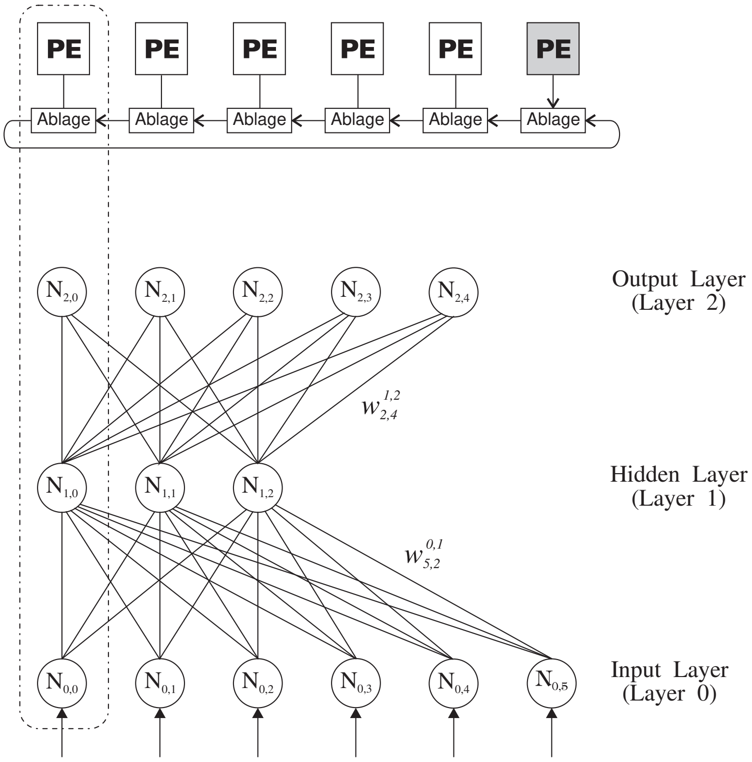

Abbildung 2: Die Neuronen und die Verbindungen eines Multilayer Perceptrons werden spaltenweise auf die Prozessoren der MasPar abgebildet. Die Anzahl der benötigten Prozessoren ist gleich der maximalen Anzahl der Neuronen in einer Schicht mit Ausnahme der Eingabeschicht. Falls die Anzahl der Eingabeneuronen größer ist als die Anzahl der Neuronen in den anderen Schichten des Netzwerkes, d.h. wenn $N_{nodes}(0) > \max N_{nodes}(k)$ für $k = 1, \ldots, N_{layers} - 1$, speichert ein weiterer Prozessor das Eingabemuster während der Vorwärtspropagierung von Schicht 0 auf Schicht 1 (in der Abbildung das Prozessorelement ganz rechts).

wird im folgenden für schichtweise vollvernetzte feedforward-Netzwerke und das Lernverfahren aus [RHW86] (mit *sigmoider* Transferfunktion) am Beispiel eines 6-3-5 Netzwerks erklärt.

3.1 Die Vorwärtspropagierung

Für die Berechnung der Aktivierung der Units, d.h. die Berechnung von Wx werden zuerst die Eingabewerte x_j^k in den „Ablagen" der Units gespeichert. Die Eingabewerte x_j^0, d.h die Eingabewerte der Units aus Layer 1, sind die Elemente des Eingabemusters n, d.h. $x_j^0 = I_j(n)$. Ansonsten stehen die Eingabewerte x_j^k für die Ausgabewerte o_j^k der Units. Während der Berechnung werden die in den „Ablagen" gespeicherten Eingabewerte nach links rotiert. Jeder Prozessor i addiert bei jedem Berechnungsschritt den Wert $w_{j,i}^{k-1,k}\, x_j^{k-1}$ zum Inhalt seines Akkumulators, der mit 0 initialisiert wurde, hinzu. Diese

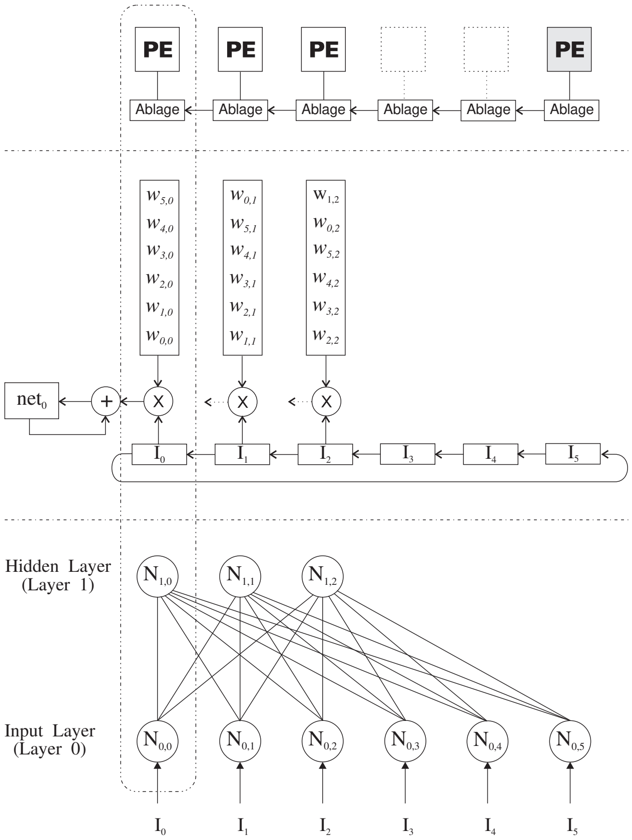

Abbildung 3: Die Vorwärtspropagierung von Layer 0 auf Layer 1. Jeder Prozessor berechnet die Netzeingabe der ihm zugeordneten Hidden-Unit i wie folgt: $net_i^1 = \sum_{j=0}^{N_{nodes}(0)-1} w_{j,i}^{0,1} I_j(n)$. Hierbei bezeichnet $N_{nodes}(0)$ die Anzahl der Neuronen im Layer 0, $w_{j,i}^{0,1}$ stellt das synaptische Gewicht der Verbindung zwischen dem Neuron j aus Layer 0 und dem Neuron i aus Layer 1 dar und I_j benennt das Element j des Eingabemusters n.

Operation, bestehend aus einer Multiplikation, einer Addition und einer Schiebeoperation, wird $\max(N_{nodes}(k-1), N_{nodes}(k))$-mal wiederholt. Die Berechnung der Netzeingabe des i-ten Neurons in der k-ten Schicht des Netzwerkes

$$net_i^k = \sum_{j=0}^{N_{nodes}(k-1)-1} w_{j,i}^{k-1,k}\, x_j^{k-1}$$

dauert somit nur $\max(N_{nodes}(k-1), N_{nodes}(k))$ Schritte, d.h. die Zeit zur Berechnung der Aktivierung des gesamten Netzwerkes ist, bei ausreichend vielen Prozessoren, proportional zur Anzahl der Neuronen des Netzwerkes. Nach Bestimmung der Netzeingabe net_i^k der Neuronen erfolgt die Berechnung der Aktivierungs- und Ausgangswerte a_i^k und o_i^k der Neuronen durch die Auswertung der *sigmoiden* Transferfunktion [4]. Der Ausgabewert $o_i^k = a_i^k = \sigma(net_i^k + \theta_i^k)$ ist hier identisch mit der Aktivierung a_i^k des Neurons. θ_i^k bezeich-

[4] Die *sigmoide* Funktion ist hier $\sigma(x) = \frac{1}{1+e^{-x}}$.

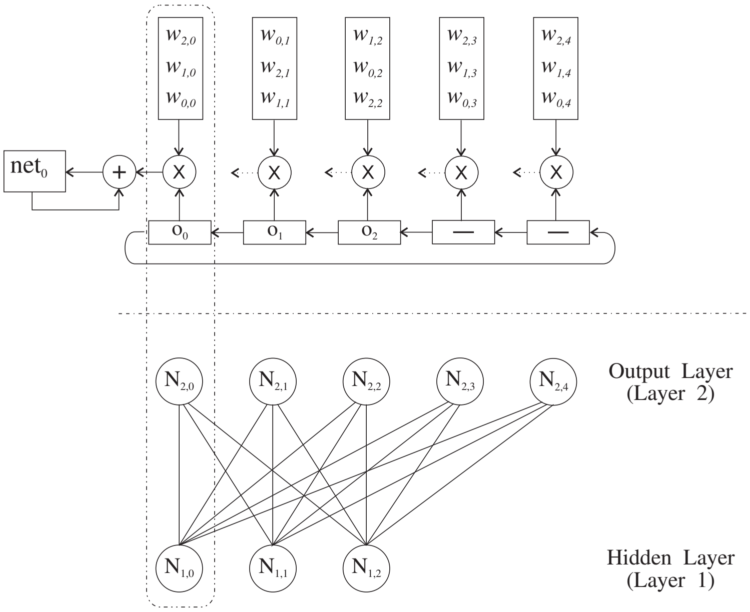

Abbildung 4: Die Vorwärtspropagierung von Layer 1 auf Layer 2. Jeder Prozessor berechnet die Netzeingabe $net_i^2 = \sum_{j=0}^{N_{nodes}(1)-1} w_{j,i}^{1,2}\, o_j^1$ der ihm zugeordneten Output-Unit i.

net den Schwellenwert oder *Offset* des Neurons. Diese Berechnungsschritte werden für die Schichten $1, \ldots, (N_{layers} - 1)$ des Netzwerkes wiederholt. Nach $(N_{layers} - 1)$ Durchläufen stehen die Ausgangswerte der Ausgangsunits fest.

3.2 Die Rückwärtspropagierung

Bei der Propagierung des Fehlers werden zuerst die Fehlerwerte der Neuronen, beginnend mit der Ausgabeschicht, bestimmt. Danach erfolgt die Berechnung der Korrekturwerte für die Verbindungsgewichte und Schwellenwerte. Die Berechnung der Fehlerwerte der Ausgabeneuronen gestaltet sich recht einfach, da sich der Fehler des Neurons i direkt aus der Differenz zwischen dem Zielwert des Ausgabemusters und dem Ausgangswert des Neurons ergibt:

$$\delta_i^{N_{layers}-1} = o_i^{N_{layers}-1} \cdot (1 - o_i^{N_{layers}-1}) \cdot (T_i(n) - o_i^{N_{layers}-1}).$$

$T_i(n)$ ist der Zielwert des Neurons i beim Muster n des Trainingssatzes. Für die Bestimmung des Fehlers der Neuronen in den verdeckten Schichten $k = 1, \ldots, (N_{layers} - 2)$ des Netzwerkes, wird der Fehlerwert der nachfolgenden Neuronen der Schicht $(k+1)$ gewichtet aufsummiert. Es gilt:

$$\delta_i^k = o_i^k \cdot (1 - o_i^k) \cdot \sum_{j=0}^{N_{nodes}(k+1)-1} w_{i,j}^{k,k+1}\, \delta_j^{k+1}.$$

Die Summation der $w_{i,j}^{k,k+1}\, \delta_j^{k+1}$ erfolgt wie in der Arbeitsphase durch Rotation der Elemente in den „Ablagen“ der Prozessoren. Im Gegensatz zur Arbeitsphase werden jedoch nicht die

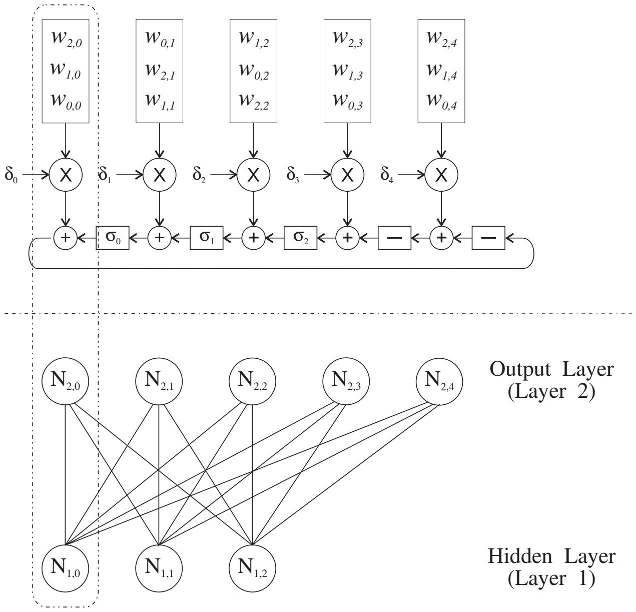

Abbildung 5: Die Berechnung der Fehlerwerte der inneren Neuronen. Jeder Prozessor berechnet den Fehlerwert $\delta_i^1 = o_i^1(1 - o_i^1) \sum_{j=0}^{N_{nodes}(2)-1} w_{i,j}^{1,2}\, \delta_j^2$. einer Hidden-Unit i durch die gewichtete Summation des Fehlers der mit dieser Unit verbundenen Output-Units j.

Eingabewerte rotiert, sondern die bei der Berechnung entstehenden Partialsummen (siehe Abbildung 5). Nach $\max(N_{nodes}(k), N_{nodes}(k+1))$-Durchläufen sind alle Partialsummen berechnet. Nach der Berechnung der Fehlerwerte δ_i^k kann die Fehlerpropagierung für die nächst-tieferliegende Schicht erfolgen. Die Berechnung der Fehlerwerte durch Summation ist nur für die Neuronen aus den inneren Schichten des Netzwerkes erforderlich.

3.3 Die Modifikation der Verbindungsgewichte

Nachdem die Fehlerwerte berechnet sind, können die Verbindungsgewichte und Schwellenwerte für alle Schichten $k = 1, \ldots, (N_{layers} - 1)$ justiert werden. Die Änderung des synaptischen Gewichts $\Delta w_{j,i}^{k-1,k}$ der Verbindung des Neurons j aus der Schicht $(k-1)$ mit dem Neuron i der Schicht k berechnet sich wie folgt:

$$\Delta w_{j,i}^{k-1,k} = \eta \cdot \delta_i^k \cdot o_j^{k-1}.$$

Die Änderung $\Delta\theta_i^k$ des Schwellenwertes θ_i^k des Neurons i ist:

$$\Delta\theta_i^k = \eta \cdot \delta_i^k.$$

η ist dabei der Lernfaktor und o_j^{k-1} ist der Ausgangswert des Neurons j aus der tieferliegenden Schicht des Netzwerkes. Ist $k = 1$, so gilt $o_j^0 = I_j(n)$, d.h. o_j^0 ist das Element j des n-ten Eingabemusters. Die Berechnung der Gewichtsänderungen $\Delta w_{j,i}^{k-1,k}$ geschieht wie in der Arbeitsphase durch die Rotation der Ausgangswerte o_j^{k-1} in den „Ablagen" der Prozessorelemente. Die Abbildungen 6 und 7 zeigen den Vorgang für die verdeckte Schicht und die Ausgabeschicht des Beispielnetzwerkes.

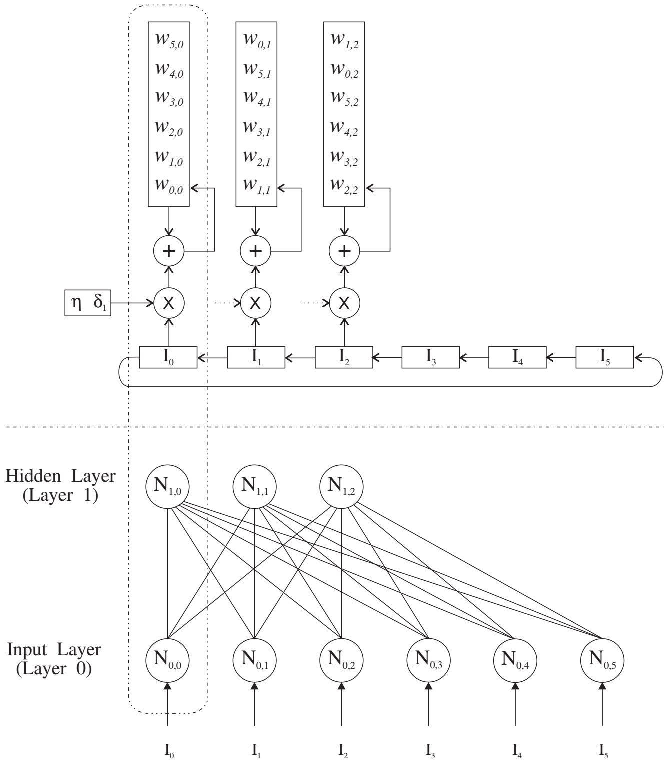

Abbildung 6: Berechnung der Gewichtsänderungen $\Delta w_{j,i}^{0,1} = \eta \cdot \delta_i^1 \cdot I_j$ der synaptischen Gewichte $w_{j,i}^{0,1}$ der Verbindungen der Eingabeneuronen j aus Schicht 0 mit den Neuronen i aus Schicht 1.

4 Parallele Berechnung verschiedener Trainingssätze

Normalerweise benötigt die Simulation eines Netzwerkes weniger Prozessoren als die MP-1216A zur Verfügung stellt. Um möglichst viele Prozessoren auszulasten, werden alle Phasen des Backpropagation-Verfahrens, d.h. die Vorwärts- und die Rückwärtspropagierung, die Berechnung der Gewichtsänderungen sowie die Modifikation der Verbindungsgewichte, auf m Kopien des Netzwerkes für m verschiedene Trainingssätze parallel durchgeführt. In den Netzwerkkopien werden die Verbindungsgewichte nicht nach jeder Lernphase geändert, sondern erst nach der Abarbeitung von $k \cdot m$ Trainingssätzen, wobei k ist eine natürliche Zahl ist. Dazu werden in den Netzwerkkopien die jeweiligen Gewichts- und Schwellenwertänderungen für die verschiedenen Trainingssätze aufsummiert. Nachdem die Berechnung der Trainingssätze abgeschlossen ist, erfolgt die Modifikation der Gewichte und Schwellenwerte durch die Addition der Änderungen Δw bzw. $\Delta \theta$ auf alle Netzwerkkopien. Dies entspricht der Änderung der Verbindungsgewichte nach jeweils $k \cdot m$ Trainingsmustern in einem einzigen Netzwerk.

Die einzelnen Netzwerke werden auf dem Prozessorarray der MasPar entweder nebeneinander oder untereinander angeordnet (Abb. 9). Diese Anordnung minimiert die Kommunikation der Prozessorelemente über den globalen Router der MasPar und wurde aufgrund

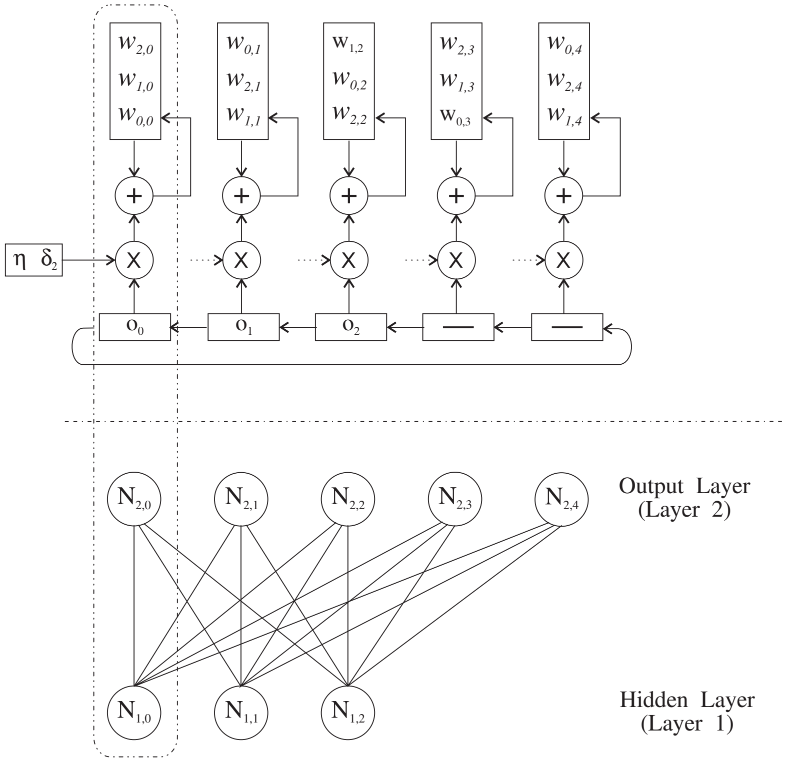

Abbildung 7: Berechnung der Gewichtsänderungen $\Delta w_{j,i}^{1,2} = \eta \cdot \delta_i^2 \cdot o_j^1$ der synaptischen Gewichte $w_{j,i}^{1,2}$ der Verbindungen der Neuronen j aus Schicht 1 mit den Ausgabeneuronen i aus Schicht 2.

von Hardwareproblemen des Routers gewählt. Die Auslastung der Prozessorelemente kann allerdings auf einen Wert von 51.6 % absinken, falls zur Darstellung eines Netzwerkes genau 65 Prozessoren benötigt werden (bei der MasPar MP-1216A).

5 Modifikationen des Verfahrens

Auf der MasPar MP-1216A sind mit den besprochenen Algorithmen Propagierungsgeschwindigkeiten bis zu einigen 100 Millionen Verbindungen pro Sekunde erreichbar. Die Eingabe- und Ausgabemuster müssen bei diesem Verfahren auf dem Subsystem bzw. Frontend gespeichert werden, da die Größe des Speichers der Prozessorelemente der MasPar mit nur 16 kByte im allgemeinen nicht ausreicht um alle Musterpaare aufzunehmen (dies gilt insbesondere für den Fall, in dem ein zusätzliches Prozessorelement benötigt wird um die restlichen Eingabemuster zu speichern). Bei der Speicherung der Muster auf dem Subsystem müssen diese mit einer recht hohen Geschwindigkeit zum Prozessorarray übertragen werden. Es ist jedoch nur unter Schwierigkeiten möglich die gewünschte Datentransferrate zu erreichen, da vom Aufruf einer Funktion zur Datenübertragung bis zum tatsächlichen Beginn der Datenübertragung einige Millisekunden vergehen (die Datenübertragung ist asynchron). Der Zeitbedarf für die Vorwärts- und Rückwärtspropagierung der Netzwerke

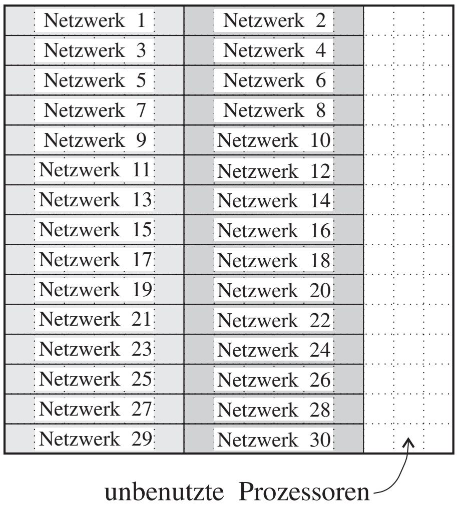

Abbildung 8: Ist die Anzahl der zur Darstellung eines Netzwerkes verwendeten Prozessorelemente kleiner oder gleich der Breite des Prozessorarrays, d.h. $N_{procs} \leq N_{PExprocs}$, so werden die einzelnen Netzwerke linksbündig und ohne Umbruch im Prozessorarray angeordnet.

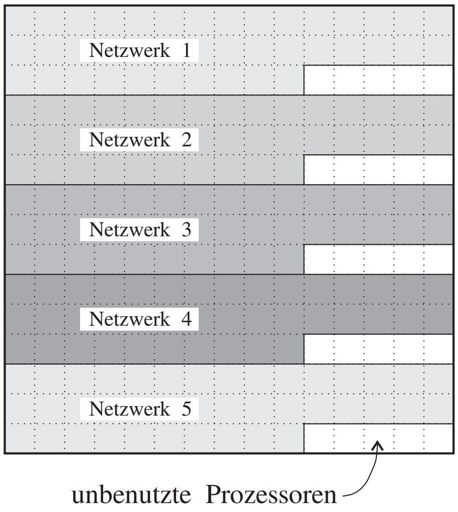

Abbildung 9: Ist die Anzahl der zur Darstellung eines Netzwerkes verwendeten Prozessorelemente größer als die Breite des Prozessorarrays ($N_{procs} > N_{PExprocs}$), werden die Netzwerke linksbündig im Prozessorarray angeordnet.

liegt aber in derselben Größenordnung wie Zeitbedarf für den Aufruf *einer* Funktion zur Datenübertragung. Dies hat einen nicht unerheblichen Aufwand für die Implementierung von effizienten Datenübertragungsroutinen zur Folge. Dieser Aufwand kann vermieden werden, indem man die oben beschriebenen Algorithmen in nur einem Punkt ändert: zur Darstellung eines Netzwerkes werden soviele PEs verwendet wie der maximalen Anzahl der Units in einer Schicht des Netzwerkes, jetzt einschließlich der Eingabeschicht, entspricht. Dadurch ist es möglich auch die Eingabemuster in den PEs zu speichern, da auch die Eingabeschicht des Netzwerks auf die Prozessoren abgebildet wird. Dies hat zwar den Nachteil, daß zur Darstellung der Netzwerke in der Regel mehr Prozessoren benötigt werden, d.h. die Auslastung der Prozessoren sinkt, dafür ist aber kein Mustertransfer während der Trainingsphase nötig.

Die hier besprochenen Algorithmen können in modifizierter Form auch für teilweise vernetzte Netzwerke angewendet werden. Der wesentliche Unterschied liegt in der Art und Weise, wie die Inhalte in den „Ablagen“ der Prozessorelemente rotiert werden. Bei vollvernetzten Netzwerken werden die Werte in den „Ablagen“ der Prozessorelemente immer nach links rotiert, während bei teilweise vernetzten Netzwerken die Inhalte in den „Ablagen“ abwechselnd nach links und nach rechts rotiert werden.

Rechnertyp	Max
VAX 8600	2.0
Convex C-1	3.6
SAIC SIGMA-1	5-8
Butterfly (64)	8
TRW MK V (16)	10
ANZA plus	10
Cray X-MP	50
MasPar MP-1216A [a]	176
CM-2 [b]	180
MasPar MP-1216A [c]	348

Tabelle 1: Die maximale Leistung verschiedener Rechnersysteme in Millionen Verbindungen pro Sekunde (Connections Per Second, CPS).

[a] Implementierung 1
[b] Implementierung nach [Zha90]
[c] Implementierung 2

Rechnertyp	NETtalk	Max
ANZA plus	-	1.5
NeuroTurbo	-	2
Delta	-	2.7
NeuMan	-	8.5
CM-1	2.8	13
WARP	17	17
CM-2 [a]	-	40
MasPar MP-1216A [b]	42	68
MasPar MP-1216A [c]	17.6	129
Sandy/6 [d]	118	141
SX-2	72	180
Sandy/8 [b]	118	567

Tabelle 2: Die Leistung verschiedener Rechnersysteme in Millionen Gewichtsänderungen pro Sekunde (Connection Updates Per Second, CUPS).

[a] Implementierung nach [Zha90]
[b] Implementierung 1
[c] Implementierung 2
[d] Spezielle Hardware zur Simulation von neuronalen Netzwerken

6 Leistungsmessungen

Im folgenden werden die Leistungen verschiedener Netzwerksimulatoren mit der Performance der hier beschriebenen Implementierungen auf der MasPar MP-1216A verglichen. Das NETtalk Netzwerk aus [SR86] ist ein Netzwerk mit 203 Eingangsneuronen, 60 Neuronen in der verdeckten Schicht und 26 Neuronen in der Ausgabeschicht. Die Maximalleistung der MasPar MP-1216A wurde mit einem vollvernetzten 128-128-128 Netzwerk ermittelt. Die Angaben der Gewichtsänderungen pro Sekunde (CUPS) beruhen auf dem Lernverfahren nach [RHW86].

Die Leistungsmessungen der Implementierung 1 (Eingabeunits werden nicht auf Prozessorelemente abgebildet) der MasPar beinhalten aufgrund der oben aufgeführten Schwierigkeiten nicht die Zeiten für den Transfer der Trainingsmuster vom Subsystem zur MasPar. Die Leistungsmessungen der Implementierung 2 (Eingabeunits werden auf Prozessorelemente abgebildet) beinhalten diese Transferzeiten. Die Performance der Implementierung 2 konnte durch Verbesserungen des Compilers und durch weitere Codeoptimierungen noch erhöht werden.

Literatur

[CGC+90] G. Chinn, K. A. Grajski, C. Chen, C. Kuszmaul, and S. Tomboulian. Systolic Array Implementations of Neural Nets on the MasPar MP-1 Massively Parallel Processor. In *Proc. Intl. J. Conf. on Neural Networks, San Diego*, San Diego, CA., 1990.

IEEE TAB Neural Network Commitee.

[Gra90] Kamil A. Grajski.
Neurocomputing using the MasPar MP-1.
Technical report, Ford Aerospace Advanced Development, San Jose, CA 95161, 1990.

[KT89] Hon Keung Kwan and Pang Chung Tsang.
Systolic Implementation of Multi-Layer Feed-Forward Neural Network with Backpropagation Learning Scheme.
In *Proc. Intl. J. Conf. on Neural Networks, Washington, D.C.*, San Diego, CA., 1989. IEEE TAB Neural Network Commitee.

[KYIA90] Hideki Kato, Hideki Yoshizawa, Hiroki Iciki, and Kazuo Asakawa.
A Parallel Neurocomputer Architecture towards Billion Connection Updates Per Second.
In *Proc. Intl. J. Conf. on Neural Networks, Washington, D.C.*, San Diego, CA., 1990. IEEE TAB Neural Network Commitee.

[Mac90] Niels Mache.
Entwurf und Realisierung eines effizienten Simulatorkerns für neuronale Netze.
Studienarbeit 895, Universität Stuttgart, Fakultät Informatik, 1990.

[Mac92] Niels Mache.
Entwicklung eines massiv parallelen Simulatorkerns für neuronale Netze auf der MasPar MP-1.
Master's thesis, Universität Stuttgart, Fakultät Informatik, 1992.

[RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams.
Parallel Distributed Processing, volume 1, chapter 8, Learning Internal Representations by Error Propagation, page 318 ff.
MIT Press, 1986.

[RM86] D.E. Rumelhart and J.L. McClelland.
Parallel Distributed Processing, volume 1.
MIT Press, 1986.

[SR86] T. J. Sejnowsky and C. R. Rosenberg.
NETtalk: A Parallel Network that Learns to Read Aloud.
Technical report, JHU/IEECS-86/01, The Hopkins University, 1986.

[Zha90] X. Zhang et al.
An efficient implementation of the back-propagation algorithm on the Connection Machine CM-2.
In *Advances in Neural Information Processing Systems*, volume 2, pages 801–809, 1990.

[ZMH+92] A. Zell, N. Mache, R. Hübner, M. Schmalzl, G. Mamier, and M. Vogt.
SNNS Stuttgart Neural Network Simulator user manual, version 2.1.
Technical Report 8/92, Institute for Parallel and Distributed High Performance Systems (IPVR), Breitwiesenstr. 20–22, D–7000 Stuttgart 80, Fed. Rep. of Germany, 1992.