(19) Deutsches Patent- und Markenamt

(10) **DE 11 2017 008 040 T5** 2020.07.09

(12) # Veröffentlichung

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2019/053835**
in der deutschen Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: **11 2017 008 040.1**
(86) PCT-Aktenzeichen: **PCT/JP2017/033225**
(86) PCT-Anmeldetag: **14.09.2017**
(87) PCT-Veröffentlichungstag: **21.03.2019**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **09.07.2020**

(51) Int Cl.: ***G06F 17/10*** (2006.01)
G06N 3/063 (2006.01)

(71) Anmelder:
MITSUBISHI ELECTRIC CORPORATION, Tokyo, JP

(74) Vertreter:
Meissner Bolte Patentanwälte Rechtsanwälte Partnerschaft mbB, 80538 München, DE

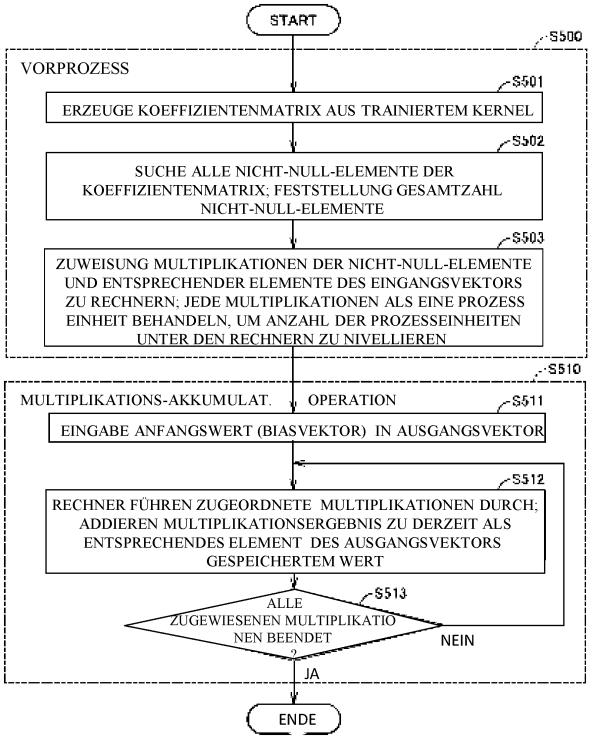
(72) Erfinder:
Tanaka, Susumu, Tokyo, JP; Mori, Masashi, Tokyo, JP; Hashimoto, Kazushige, Amagasaki-shi, Hyogo, JP

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **RECHENOPERATIONSSCHALTUNG, RECHENOPERATIONSVERFAHREN UND PROGRAMM**

(57) Zusammenfassung: Es wird ein Rechenoperationsverfahren für eine Faltungsschicht in einem Gefalteten Neuronalen Netzwerk bereitgestellt. Das Rechenoperationsverfahren beinhaltet Folgendes: Erzeugung einer Koeffizientenmatrix (A) durch Konvertierung eines in der Faltungsschicht verwendeten Kernels so, dass die Koeffizientenmatrix mit einem Eingangsvektor (x) verknüpft wird, der durch die Erweiterung einer Merkmalskarte, die in die Faltungsschicht eingegeben wurde, in eine Spalte erhalten wird; Suche nach Nicht-Null-Elementen, die in der Koeffizientenmatrix enthalten sind; Zuweisen von Multiplikationen der in der Koeffizientenmatrix enthaltenen Nicht-Null-Elemente und entsprechender Elemente des Eingangsvektors an eine Vielzahl von Rechnern (CL), wobei jede der Multiplikationen als eine Prozesseinheit behandelt wird, um die Anzahl der Prozesseinheiten zwischen den Rechnern auszugleichen, wobei alle Rechner in der Lage sind, Prozesse parallel zueinander auszuführen; und sequentielles Ausführen der zugewiesenen Multiplikationen durch die Rechner und sequentielles Addieren der Ergebnisse der Multiplikationen zu entsprechenden Elementen eines Ausgangsvektors (f) durch den Rechner.



Beschreibung

TECHNISCHES GEBIET

[0001] Die vorliegende Erfindung bezieht sich auf eine Rechenoperationsschaltung, ein Rechenoperationsverfahren und ein Programm zur Durchführung des Rechenoperationsverfahrens und wird z.B. für eine Rechenoperation einer Faltungsschicht im Gefalteten Neuronalen Netzwerk verwendet.

STAND DER TECHNIK

[0002] Ein Rechenoperationsverfahren mit der Bezeichnung „Gefaltetes Neuronales Netzwerk (CNN)“ wird häufig in vielen Bereichen eingesetzt, z.B. in der Bildverarbeitung zur Mustererkennung oder ähnlichem, in der Spracherkennung und in der Robotik. Im Allgemeinen besteht das CNN aus einer Faltungsschicht, die eine Faltungsoperation durchführt, einer Pooling-Schicht, die lokale Statistiken berechnet, und einer vollständig verknüpften Schicht. Die Faltungsschicht erzeugt eine Ausgabe-Merkmalskarte auf folgende Weise: Beim Scannen eines Kernels (auch „Filter“ genannt) auf einer Eingabe Merkmalskarte, die auf Pixeln als Einheit basiert, wird wiederholt eine Multiplikations-Akkumulationsoperation zwischen einem entsprechenden Bereich der Eingabe Merkmalskarte und dem Kernel durchgeführt und dann ein endgültiges Ergebnis der Multiplikations-Akkumulationsoperation nichtlinear konvertiert.

[0003] Die Japanische Patentanmeldungs-Offenlegungsschrift JP 2010-134 697 A (Patentdokument 1) beschreibt eine Rechenoperationsschaltung zur Durchführung einer Faltungsoperation im Wege der Parallelverarbeitung. Insbesondere führt die im Patentdokument beschriebene Rechenoperationsschaltung eine parallele Rechenoperation unter Verwendung der jeweiligen Anzahl von Multiplikatoren und Akkumulatoren durch, die der Anzahl der Spalten des Kernels entsprechen.

STAND DER TECHNIK

[0004] Patentdokument 1: Japanische Patentanmeldungs-Offenlegungsschrift JP 2010-134 697 A

KURZBESCHREIBUNG DER ERFINDUNG

Mit der Erfindung zu lösende Probleme

[0005] In der oben beschriebenen Rechenoperationsschaltung des oben beschriebenen Patentdokuments werden die Merkmalskarte und der Kernel unverändert für die Rechenoperation verwendet. Dadurch wird die Anzahl der Wiederholungen der Rechenoperation erhöht, mit der Folge, dass der Prozess langsam und nachteilig wird.

[0006] Um dem entgegenzuwirken, wird zur Verkürzung der Prozesszeit häufig ein Verfahren zur Durchführung einer Rechenoperation nach Erweiterung der Merkmalskarte oder des Kernels auf eine Spalte verwendet. Nach diesem Verfahren ergibt die Faltungsoperation ein Produkt aus einer Koeffizientenmatrix und einem Vektor, d.h. eine Multiplikations-Akkumulationsoperation eines Elements jeder Zeile der Koeffizientenmatrix und eines Elements des Vektors. Zum Beispiel kann eine parallele Rechenoperation durch die Anzahl der Multiplikations-Akkumulationsrechner entsprechend der Anzahl der Zeilen der Koeffizientenmatrix durchgeführt werden.

[0007] Wenn hier die Multiplikations-Akkumulationsoperation einfach von jedem Multiplikations-Akkumulationsrechner durchgeführt wird, wobei 0 in die Elemente der Koeffizientenmatrix eingeschlossen wird, wird Zeit für überflüssige Rechenoperationen verbraucht. Daher wird die Rechenoperation normalerweise mit dem Multiplikations-Akkumulationsrechner durchgeführt, wobei die Null-Elemente der Koeffizientenmatrix ausgeschlossen werden. Die Anzahl der Nicht-Null-Elemente unterscheidet sich jedoch zwischen den Zeilen der Koeffizientenmatrix. Auch wenn die Anzahl der Nicht-Null-Elemente in der gesamten Koeffizientenmatrix gering ist, wird eine gesamte Prozesszeit durch eine Zeile mit der größten Anzahl von Nicht-Null-Elementen bestimmt. Wenn es nur eine Zeile mit einer Vielzahl von Nicht-Null-Elementen gibt, wird dementsprechend die gesamte Prozesszeit durch die Multiplikations-Akkumulationsoperation in dieser Zeile bestimmt, mit dem Ergebnis, dass die gesamte Prozesszeit nicht wie erwartet verkürzt werden kann.

[0008] Die vorliegende Erfindung wurde unter Berücksichtigung der oben beschriebenen Problematik erstellt und hat zum Ziel, eine Rechenoperationsschaltung und ein Rechenoperationsverfahren bereitzustellen, durch

die jeweils die gesamte Prozesszeit verkürzt werden kann, wenn eine Multiplikation einer Koeffizientenmatrix mit 0 in Elementen und einem Vektor durchgeführt wird. Es ist zu beachten, dass jede der Rechenoperationsschaltungen und Rechenoperationsverfahren gemäß der vorliegenden Erfindung für eine Faltungsoperation in einem CNN geeignet ist, aber nicht nur für das CNN, sondern auch für andere Bereiche anwendbar ist.

Mittel zum Lösen der Probleme

[0009] Bei einer Rechenoperationsschaltung gemäß einer Ausführungsform der Erfindung ist eine Koeffizientenmatrix, die Nicht-Null-Elemente und Null-Elemente enthält, mit einem Eingangsvektor von rechts zu multiplizieren und ein Rechenoperationsergebnis als Ausgangsvektor auszugeben. Die Rechenoperationsschaltung weist Folgendes auf: Einen Steuerprozessor; und eine Vielzahl von Rechnern, die jeweils in der Lage sind, einen Prozess parallel zueinander durchzuführen. Der Steuerprozessor weist dem Rechner Multiplikationen der in der Koeffizientenmatrix enthaltenen Nicht-Null-Elemente und entsprechende Elemente des Eingangsvektors zu, wobei jede der Multiplikationen als eine Prozesseinheit behandelt wird, um die Anzahl der Prozesseinheiten im Rechner zu nivellieren. Der Rechner führt die zugeordneten Multiplikationen nacheinander aus und addiert die Ergebnisse der Multiplikationen nacheinander zu den entsprechenden Elementen des Ausgangsvektors.

Effekt der Erfindung

[0010] Nach der oben beschriebenen Ausführungsform werden die Multiplikationen der in der Koeffizientenmatrix enthaltenen Nicht-Null-Elemente und die entsprechenden Elemente des Eingangsvektors dem Rechner zugeordnet, wobei jede der Multiplikationen als eine Prozesseinheit behandelt wird, um die Anzahl der Prozesseinheiten im Rechner zu nivellieren, wodurch die gesamte Prozesszeit verkürzt werden kann.

Figurenliste

Fig. 1 ist ein Flussdiagramm, das einen Rechenoperationsprozess durch ein CNN zeigt;

Fig. 2 illustriert eine Faltungsoperation;

Fig. 3 zeigt die Erweiterung einer Merkmalskarte und eines Kernels;

Fig. 4 ist ein Blockdiagramm, das eine beispielhafte Konfiguration eines Parallelcomputers zeigt;

Fig. 5 ist ein Flussdiagramm, das einen Überblick über den Ablauf der Faltungsoperation zeigt;

Fig. 6 ist ein Flussdiagramm, das Details des Ablaufs der Faltungsoperation zeigt;

Fig. 7 ist ein Flussdiagramm, das eine beispielhafte Vorgehensweise bei der Durchführung einer Einheits-Multiplikations-Akkumulationsoperation in jedem Rechner des Parallelrechners von **Fig. 4** zeigt;

Fig. 8 veranschaulicht einen Effekt einer ersten Ausführungsform und

Fig. 9 ist ein Flussdiagramm, das den Ablauf einer Faltungsoperation nach einer zweiten Ausführungsform zeigt.

BESCHREIBUNG DER AUSFÜHRUNGSFORMEN

[0011] Im Folgenden wird jede Ausführungsform unter Bezug auf die Zeichnungen beschrieben. Dabei ist zu beachten, dass die gleichen oder entsprechende Bestandteile die gleichen Bezugszeichen erhalten und nicht wiederholt beschrieben werden.

Ausführungsform 1

CNN

[0012] Zunächst wird ein CNN kurz beschrieben. **Fig. 1** ist ein Flussdiagramm, das einen Rechenoperationsprozess des CNN zeigt.

[0013] Wie in **Fig. 1** gezeigt, enthält das CNN eine Eingangsschicht **S201**, Faltungsschichten **S202**, **S204**, Pooling-Schichten **S203**, **S205**, eine vollständig verknüpfte Schicht **S206** und eine Ausgangsschicht **S207**.

[0014] Die Eingabeschicht **S201** empfängt die Eingabe der zu verarbeitenden Daten, wie z.B. Bilddaten. Die Ausgabeschicht **S207** gibt nach der Datenverarbeitung ein Endergebnis aus. Zur einfacheren Beschreibung

wird in **Fig. 1** eine Kombination aus Faltungsschicht und Pooling-Schicht zweimal wiederholt (**S202**, **S203**; **S204**, **S205**), kann aber auch mehrmals wiederholt werden.

[0015] Die Dateneingabe in die Faltungsschicht wird als „Eingabe Merkmalskarte“, die Datenausgabe aus der Faltungsschicht als „Ausgabe-Merkmalskarte“ bezeichnet. Jede der Faltungsschichten **S202**, **S204** führt wiederholt eine Multiplikations-Akkumulationsoperation eines entsprechenden Bereichs der Eingabe Merkmalskarte und einen Kernel (auch als „Filter“ bezeichnet) durch, während der Kernel auf der Eingabe Merkmalskarte basierend auf Pixel als Einheit gescannt wird, und konvertiert ein endgültiges Ergebnis der Multiplikations-Akkumulationsoperation nichtlinear, wodurch die Ausgabe-Merkmalskarte erzeugt wird. Ein Element (auch als „Gewicht“ bezeichnet) des Kernels wird durch Training im Voraus bestimmt. Einzelheiten der Faltungsoperation werden später mit Bezug auf **Fig. 2** beschrieben.

[0016] Jede der Pooling-Schichten **S203**, **S205** führt eine Operation durch, um Elemente einer lokalen Domäne der Ausgabe-Merkmalskarte in einem Element zu sammeln, um die räumliche Größe der Merkmalskarte zu reduzieren. Jede der Pooling-Schichten **S203**, **S205** nimmt den maximalen Wert der lokalen Domäne oder mittelt z.B. die in der lokalen Domäne enthaltenen Elemente.

[0017] Eine oder mehrere vollständig verknüpfte Schichten **S206** sind neben der Ausgabeschicht **S207** vorgesehen. Jedes Neuron der vollständig verknüpften Schicht(en) **S206** hat eine Verbindung zu allen Neuronen einer benachbarten Schicht.

Faltungsoperation

[0018] **Fig. 2** veranschaulicht die Faltungsoperation. Wie in **Fig. 2** dargestellt, werden die Ausgangsdaten **102** durch eine Faltungsoperation aus einem Kernel **101** und Eingabedaten **100** als Eingabe Merkmalskarte erzeugt. Jedem Element der Ausgangsdaten **102** wird ein Bias hinzugefügt, und es wird auch eine aktivierende Funktion darauf angewendet, wodurch die Ausgangs-Merkmalskarte erzeugt wird. Als aktivierende Funktion wird eine nichtlineare Funktion wie z.B. ReLU (Rectified Linear Unit) verwendet.

[0019] Der Einfachheit halber wird im Beispiel von **Fig. 2** die Eingabedaten-Größe auf (7, 7) und die Kernel-Größe auf (3, 3) gesetzt. Zur Einstellung einer Ausgangsdatengröße kann die Umgebung **104** der Eingabedaten **100** mit festen Daten (z.B. 0) gefüllt werden. Dies wird als „Padding“ bezeichnet. Ein Padding mit einer Breite von 1 und einem Wert von 0 wird auf Eingabedaten **100** von **Fig. 2** angewendet.

[0020] In der Faltungsoperation werden beim Verschieben des Kernel **101** in einem bestimmten Intervall auf Eingabedaten **100** einschließlich der Padding-Anteile Elemente des Kernel **101** mit entsprechenden Elementen der Eingabedaten **100** multipliziert und eine Summe daraus berechnet. Das heißt, es wird eine Multiplikations-Akkumulationsoperation durchgeführt. Ein Ergebnis der Multiplikations-Akkumulationsoperation wird als entsprechendes Element der Ausgangsdaten **102** gespeichert. Das Intervall, in dem Kernel **101** verschoben wird, wird als „schreiten“ bezeichnet. Im Fall von **Fig. 2** ist der Schritt **1**.

[0021] Insbesondere wenn Kernel **101** so angeordnet ist, dass er einem Rahmen **103** entspricht, der durch eine dicke durchgezogene Linie in **Fig. 2** angezeigt wird, wird ein Ergebnis der Multiplikations-Akkumulationsoperation von „**30**“ als ein positionsmäßig entsprechendes Element **106** der Ausgangsdaten **102** gespeichert. Wenn Kernel **101** so angeordnet ist, dass er einem Rahmen **105** entspricht, der durch eine dicke gestrichelte Linie in **Fig. 2** angezeigt wird, wird das Ergebnis der Multiplikations-Akkumulationsoperation von „13“ als positionsmäßig entsprechendes Element **107** der Ausgangsdaten **102** gespeichert.

Erweiterung von Merkmalskarte und Kernel.

[0022] **Fig. 3** veranschaulicht die Erweiterung der Merkmalskarte und des Kernels. Im Falle der vorliegenden Ausführungsform wird die Merkmalskarte zur Verkürzung der Prozesszeit der Faltungsoperation zu einer Spalte erweitert, indem die jeweiligen Reihen der Merkmalskarte miteinander verbunden werden.

[0023] Konkret wird unter Bezugnahme auf **Fig. 2** und **Fig. 3** ein Eingangsvektor **110** aus **Fig. 3** erzeugt, indem die jeweiligen Zeilen der Eingabedaten **100** aus **Fig. 2** miteinander verbunden werden. Die Anzahl der Elemente des Eingangsvektors **110** entsprechend Eingabedaten **100** beträgt 7×7=49. Auch die Zeilen der Ausgangsdaten **102** aus **Fig. 2** werden miteinander verbunden, um sich zu einer Spalte zu erweitern. Die Anzahl der Elemente des Ausgangsvektors, die den Ausgangsdaten **102** entsprechen, beträgt ebenfalls **49**.

[0024] Kernel **101** aus **Fig. 2** wird zu einer Matrix erweitert, um einen Ausgangsvektor zu erzeugen, der den Ausgangsdaten **102** aus **Fig. 2** entspricht, wenn er mit dem Eingangsvektor **110** von rechts multipliziert wird. Entsprechend wird eine Koeffizientenmatrix **111** erzeugt. Die Koeffizientenmatrix **111** hat eine erste bis neunundvierzigste Zeile und hat somit **49** Zeilen. Die Koeffizientenmatrix **111** hat eine erste bis neunundvierzigste Spalte und hat daher **49** Spalten. Es ist zu beachten, dass in der Koeffizientenmatrix **111**, die in **Fig. 3** dargestellt ist, Elemente in leeren Quadraten **0** darstellen.

[0025] Die erste Zeile der Koeffizientenmatrix **111** ist (3, 2, 0, 0, 0, 0, 0, 1, 3, 0, ..., 0) und entspricht dem Fall, dass Kernel **101** von **Fig. 2** so angeordnet ist, dass er dem Rahmen **105** entspricht, der durch die dicke gestrichelte Linie auf der Merkmalskarte **100** angezeigt wird. Durch die Durchführung der Multiplikations-Akkumulationsoperation der ersten Zeile der Koeffizientenmatrix **111** und des Eingangsvektors **110** werden die Daten „13“, die als positionsentsprechendes Element **107** der Ausgangsdaten **102** von **Fig. 2** zu speichern sind, erzeugt.

[0026] In ähnlicher Weise ist die neunte Zeile der Koeffizientenmatrix **111** (3, 2, 1, 0, 0, 0, 0, 1, 3, 2, 0, 0, 0, 0, 2, 1, 3, 0, ..., 0) und entspricht dem Fall, dass Kernel **101** von **Fig. 2** so angeordnet ist, dass er dem Rahmen **103** entspricht, der durch die dicke durchgezogene Linie auf der Merkmalskarte **100** angezeigt wird. Durch die Durchführung der Multiplikations-Akkumulationsoperation der neunten Zeile der Koeffizientenmatrix **111** und des Eingangsvektors **110** werden die Daten „**30**“, die als positionsentsprechendes Element **106** der Ausgangsdaten **102** von **Fig. 2** zu speichern sind, erzeugt.

[0027] Wenn in **Fig. 2** kein Padding angewendet wird, ist der Eingangsvektor **110** entsprechend Eingabedaten **100** unverändert und hat **49** Elemente. Da die Datengröße der Ausgangsdaten **102** (5, 5) ist, beträgt die Anzahl der Elemente eines Ausgangsvektors, der den Ausgangsdaten **102** entspricht, 5×5=25. Außerdem ist die Anzahl der Zeilen der Koeffizientenmatrix **111**, die dem Kernel **101** entspricht, **25**, und die Anzahl der Spalten **49**.

[0028] Im Allgemeinen wird eine in der Faltungsoperation durchgeführte Matrixoperation durch eine Formel (1) ausgedrückt. Das heißt, einen Ausgangsvektor f der Faltungsoperation erhält man durch Multiplikation der Koeffizientenmatrix A mit dem Eingangsvektor x von rechts und durch Addition eines Biasvektors b zum Ergebnis der Rechenoperation. Eine Besonderheit der Koeffizientenmatrix A besteht darin, dass die Koeffizientenmatrix A eine vergleichsweise große Anzahl von Elementen enthält, die jeweils den Wert 0 haben.

$$f = A \cdot x + b$$

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \qquad (1)$$

[0029] In dieser Beschreibung werden die Elemente des Ausgangsvektors f als f_1, ..., f_n angegeben. Das i-te Element des Ausgangsvektors f wird als f_i oder f(i) angegeben. Die Elemente des Eingangsvektors x werden als x_1, ..., x_m angegeben. Das j-te Element des Eingangsvektors x wird als x_j oder x(j) angegeben. Die Elemente des Biasvektors b werden als b_1, ..., b_n angegeben. Das Element des i-ten Biasvektors b wird als bi oder b(i) angegeben. Außerdem besteht die Koeffizientenmatrix A aus n Zeilen der ersten bis n-ten Zeile und m Spalten der ersten bis m-ten Spalte. Ein Element der i-ten Zeile und der j-ten Spalte der Koeffizientenmatrix A wird als A_{ij} oder A(i, j) angegeben.

Konfiguration des Parallelrechners

[0030] Die durch die obige Formel (1) dargestellte Matrix-Operation kann von einem Parallelcomputer mit mehreren Rechnern ausgeführt werden. Im Folgenden wird eine beispielhafte Konfiguration eines Allzweck-Parallelcomputers beschrieben. Im Gegensatz zum Beispiel in **Fig. 4** kann die durch die obige Formel (1) dargestellte Matrix-Operation von einem ASIC (Application Specific Integrated Circuit) ausgeführt werden, der eine Vielzahl von Rechnern enthält, die jeweils in der Lage sind, einen Prozess parallel zueinander auszuführen.

[0031] **Fig. 4** ist ein Blockdiagramm, das eine beispielhafte Konfiguration des Parallelcomputers zeigt. Unter Bezugnahme auf **Fig. 4** enthält der Parallelcomputer **120** Folgendes: Mehrere Verarbeitungseinheiten **121A**, **121B**, ..., die jeweils mehrere Rechner **CL0**, **CL1**, ... enthalten; einen L2-Cache-Speicher (Level 2 Cache-Speicher) **125**; und einen dedizierten Speicher **126**.

[0032] Jede Verarbeitungseinheit **121** (**121A**, **121B**, ...) enthält Folgendes: Die Vielzahl von Rechner **CL0**, **CL1**, ...; Registergruppen **R0**, **R1** ... entsprechend dem jeweiligen Rechner; und einen L1-Cache-Speicher (Level 1 Cache-Speicher) **122**. Die Rechner **CL** (**CL0**, **CL1**, ...), die in der gleichen Verarbeitungseinheit **121** enthalten sind, teilen sich den L1-Cache-Speicher **122**.

[0033] In der oben beschriebenen Konfiguration kann die Vielzahl der Verarbeitungseinheiten **121A**, **121B**, ... ein Programm parallel zueinander ausführen. Weiterhin kann die Vielzahl der Rechner **CL0**, **CL1**, ... jeder Verarbeitungseinheit **121** ein Programm parallel zueinander ausführen. Es ist zu beachten, dass das Programm über ein Netzwerk bereitgestellt werden kann, oder durch ein Speichermedium, das das Programm in einer nicht-transitorischen Art und Weise speichert, indem ein magnetisches oder optisches Verfahren, ein Halbleiterspeicher oder ähnliches verwendet wird.

[0034] Der Parallelrechner **120** kann über eine Hochgeschwindigkeitsschnittstelle an eine CPU (Zentrale Verarbeitungseinheit) **130** angeschlossen sein. Die CPU **130** steuert das gesamte Programm. In diesem Fall kann die Datenübertragung durch direkten Speicherzugriff zwischen einem Speicher **131** für die CPU **130** und einem dedizierten Speicher **126** des Parallelcomputers **120** zugelassen werden. Im Gegensatz zu der Konfiguration in **Fig. 4** können die CPU **130** für die Steuerung und der Speicher **131** auch im den Parallelcomputer **120** eingebaut sein.

Übersicht über den Ablauf der Faltungsoperation

[0035] Im Folgenden wird ein Überblick über einen Ablauf der Faltungsoperation beschrieben, insbesondere ein Überblick über einen Ablauf der durch die Formel (1) dargestellten Matrixoperation.

[0036] **Fig. 5** ist ein Flussdiagramm, das den Ablauf der Faltungsoperation im Überblick zeigt. Es wird davon ausgegangen, dass jedes Element des für die Faltungsoperation verwendeten Kernels bereits trainiert wurde. Wie in **Fig. 5** dargestellt, kann ein Faltungsoperationsprozess unterteilt werden in einen Vorprozess **S500**, der zu Beginn nur einmal ausgeführt wird, und eine Multiplikations-Akkumulationsoperation **S510**, die als Antwort auf Eingabedaten wiederholt wird. Der Vorprozess kann von einer Allzweck-CPU (z.B. CPU **130** in **Fig. 4**) durchgeführt werden. Andererseits wird die Multiplikations-Akkumulationsoperation hauptsächlich von dem Parallelrechner **120** aus **Fig. 4** durchgeführt und im Allgemeinen von der CPU **130** gesteuert.

[0037] In der Vorverarbeitungsphase erzeugt zunächst in einem Schritt **S501**, wie in **Fig. 2** und **Fig. 3** dargestellt, ein Prozessor wie die CPU aus dem trainierten Kernel die Koeffizientenmatrix A. Die erzeugte Koeffizientenmatrix A wird in einem Speicher abgelegt.

[0038] In einem nächsten Schritt **S502** sucht der Prozessor nach allen Nicht-Null-Elementen der Koeffizientenmatrix A. Die Koeffizientenmatrix A wird in einem Speicher abgelegt. Das Suchergebnis wird im Speicher gespeichert. Dementsprechend wird die Gesamtzahl der in der Koeffizientenmatrix A enthaltenen Nicht-Null-Elemente gefunden. Es wird davon ausgegangen, dass die Eingabedaten in eine Spalte expandiert und entsprechend in den Eingangsvektor umgewandelt werden.

[0039] In einem nächsten Schritt **S503** weist der Prozessor dem Rechner **CL** Multiplikationen der gesuchten Nicht-Null-Elemente und entsprechende Elemente des Eingangsvektors x zu, wobei jede der Multiplikationen als eine Prozesseinheit behandelt wird, um die Anzahl der Prozesseinheiten unter den Rechnern **CL**, die im Parallelrechner **120** enthalten sind, zu nivellieren. Dementsprechend können die Prozesszeiten zwischen den Rechnern im Wesentlichen gleich gemacht werden. Auf diese Weise wird die Vorverarbeitungsphase beendet.

[0040] In der nächsten Multiplikations-Akkumulationsoperation wird zunächst in einem Schritt **S511** der Biasvektor b als Anfangswert in den Ausgangsvektor f eingegeben. Es ist zu beachten, dass der Biasvektor b am Ende der Multiplikations-Akkumulationsoperation zum Ausgangsvektor f addiert werden kann.

[0041] Im nächsten Schritt **S512** führt jeder Rechner **CL** des Parallelrechners **120** nacheinander die zugeordnete Multiplikation durch. Der Rechner **CL** addiert ein Multiplikationsergebnis zu einem Wert, der derzeit als entsprechendes Element des Ausgangsvektors f gespeichert ist. Das heißt, das Multiplikationsergebnis wird sequentiell zum entsprechenden Element des Ausgangsvektors f addiert. Schritt **S512** wird wiederholt, bis alle zugewiesenen Multiplikationen beendet sind (bis in Schritt **S513** JA bestimmt wird).

[0042] Es ist zu beachten, dass in jedem der Schritte **S512**, **S513** berücksichtigt werden kann, dass die Multiplikations-Akkumulationsoperationen jeder Zeile der Koeffizientenmatrix A und des Eingangsvektors x zur

Ausführung in Einheits-Multiplikations-Akkumulationsoperationen aufgeteilt werden können. Hier besteht jede der Einheits-Multiplikations-Akkumulationsoperationen aus Folgendem: Einer Multiplikation eines Nicht-Null-Elements der Koeffizientenmatrix A und eines entsprechenden Elements des Eingangsvektors; und einer Addition eines Multiplikationsergebnisses zu einem entsprechenden Element des Ausgangsvektors f.

[0043] Im Folgenden wird ein kurzes spezifisches Beispiel dafür beschrieben. Es wird z.B. angenommen, dass in der Koeffizientenmatrix **A** der Formel (1) n=m und nur diagonale Elemente Nicht-Null-Elemente sind. Außerdem wird angenommen, dass die Gesamtzahl der Rechner **CL** insgesamt n ist. In diesem Fall führt der i-te ($1 \leq i \leq n$) Rechner **CL** eine Multiplikation von A_{ii} und x_i durch und addiert das Multiplikationsergebnis zum Bias-Wert bi, der aktuell als Element f_i des Ausgangsvektors f gespeichert ist.

[0044] Als weiteres Beispiel wird angenommen, dass in der Koeffizientenmatrix **A** der Formel (1) nur die Elemente A_{11} bis A_{1m} der ersten Zeile Nicht-Null-Elemente sind. Darüber hinaus wird angenommen, dass die Gesamtzahl der Rechner **CL** m ist. In diesem Fall führt der i-te ($1 \leq i \leq m$) Rechner **CL** eine Rechenoperation von $A_{1i} \cdot x_i$ durch und addiert das Ergebnis der Rechenoperation zu dem Wert, der derzeit als erstes Element f_1 des Ausgangsvektors f gespeichert ist. Da die Additionsoperationen des Rechners **CL** miteinander in Konflikt stehen, wird in diesem Fall eine exklusive Steuerung durchgeführt. Beispielsweise addiert der erste Rechner **CL** das Ergebnis der Rechenoperation von $A_{11} \cdot x_1$ zum Anfangswert b_1 des Elements f_1 des Ausgangsvektors f. Nach Ende dieser Additionsoperation addiert der zweite Rechner **CL** ein Ergebnis der Rechenoperation von $A_{12} \cdot x_2$ zu $b_1 + A_{11} \cdot x_1$, das derzeit als Element f_1 des Ausgangsvektors f gespeichert ist. Anschließend wird die Additionsoperation in der gleichen Weise sequentiell wiederholt.

Details des Verfahrens der Faltungsoperation

[0045] **Fig. 6** ist ein Flussdiagramm, das den Ablauf der Faltungsoperation im Detail zeigt.

[0046] Unter Bezugnahme auf **Fig. 6** entspricht der Schritt **S101** bis zum Schritt **S108** dem Vorprozess von Schritt **S500** aus **Fig. 5**.

[0047] Zunächst werden die Variablen in Schritt **S101** initialisiert. Konkret initialisiert der Prozessor sowohl eine Zeilenrichtungsvariable i als auch eine Spaltenrichtungsvariable j auf 1 und initialisiert die Anzahl k der Nicht-Null-Elemente (d.h. die Anzahl der Einheits-Multiplikations-Akkumulationsoperationen) auf 0.

[0048] Im nächsten Schritt **S102** wird bestimmt, ob ein Element ein Nicht-Null-Element ist oder nicht. Konkret prüft der Prozessor den Wert eines Elements A(i, j) der Koeffizientenmatrix **A**. Wenn der Wert **0** ist, ist das Element A(i, j) ein Element, das nicht für die Multiplikations-Akkumulationsoperation vorgesehen ist. Daher fährt der Prozess mit Schritt **S105** fort, um die Zeilenrichtungsvariable i hochzuzählen. Wenn der Wert des Elements A(i, j) jedoch nicht 0 ist, ist das Element A(i, j) ein Element, das für die Multiplikations-Akkumulationsoperation vorgesehen ist. Daher setzt der Prozessor den Prozess zum Schritt **S103** fort, um das Nicht-Null-Element zu speichern.

[0049] In Schritt **S103** wird ein Zeiger auf das gesuchte Nicht-Null-Element gespeichert. Insbesondere speichert der Prozessor zur Speicherung des Zeigers auf das Element A(i, j), das für die Multiplikations-Akkumulationsoperation vorgesehen ist, die Zeilenrichtungsvariable i in einem Zeilenzahl-Speicherarray $A_{ROW}(k)$ und die Spaltenrichtungsvariable j in einem Spaltenzahl-Speicherarray $A_{COL}(k)$.

[0050] Im nächsten Schritt **S104** wird die Anzahl k der Nicht-Null-Elemente hochgezählt. Insbesondere inkrementiert der Prozessor die Variable k, die die Anzahl der Nicht-Null-Element anzeigt.

[0051] Im nächsten Schritt **S105** inkrementiert der Prozessor die Zeilenrichtungsvariable i, um die nächste Zeile zu prüfen.

[0052] Im nächsten Schritt **S106** wird, um den Prozess in die nächste Spalte zu überführen, wenn ein Prozess für eine Spalte abgeschlossen ist, bestimmt, ob der Prozess für eine Spalte der Koeffizientenmatrix abgeschlossen ist oder nicht. Insbesondere durch den Vergleich der Zeilenrichtungsvariablen i mit der Anzahl n von Zeilen der Koeffizientenmatrix bestimmt der Prozessor, ob der Prozess für die aktuelle Spalte abgeschlossen ist oder nicht. Wenn die Zeilenrichtungsvariable i größer als die Anzahl n von Zeilen ist, stellt der Prozessor fest, dass der Prozess für die aktuelle Spalte abgeschlossen ist oder nicht. Um den Prozess für die nächste Spalte durchzuführen, setzt der Prozessor den Prozess in Schritt **S107** zur Aktualisierung der Spaltenrichtungsvariablen j fort. Wenn andererseits die Zeilenrichtungsvariable i nicht größer als die Anzahl n von Zeilen ist, stellt der

Prozessor fest, dass der Prozess für die gegenwärtige Spalte noch nicht abgeschlossen ist. Um den Prozess für die nächste Zeile durchzuführen, setzt der Prozessor den Prozess in Schritt **S102** fort, um zu bestimmen, ob das Element ein Nicht-Null-Element ist oder nicht.

[0053] In Schritt **S107** werden die Variablen für den Prozess für die nächste Spalte aktualisiert. Insbesondere initialisiert der Prozessor die Zeilenrichtungsvariable i auf 1, um den Prozess ab der ersten Zeile der nächsten Spalte durchzuführen. Außerdem inkrementiert der Prozessor die Spaltenrichtungsvariable j.

[0054] Im nächsten Schritt **S108** wird ermittelt, ob der Prozess für alle Spalten abgeschlossen ist. Um zu bestimmen, ob der Prozess für alle Spalten abgeschlossen ist oder nicht, bestimmt der Prozessor durch Vergleich der Spaltenrichtungsvariablen j mit der Anzahl m der Spalten der Koeffizientenmatrix A, ob der Prozess für die gesamte Matrix abgeschlossen ist oder nicht. Wenn die Spaltenrichtungsvariable j größer ist als die Anzahl m der Spalten, bestimmt der Prozessor, dass der Prozess für die gesamte Matrix abgeschlossen ist, und setzt den Prozess mit Schritt **S109** zur Initialisierung der Variable f (entsprechend dem Ausgangsvektor) der Multiplikations-Akkumulationsoperation fort. Wenn andererseits die Spaltenrichtungsvariable j nicht größer ist als die Anzahl m der Spalten, stellt der Prozessor fest, dass eine nicht verarbeitete Spalte übrig bleibt, und setzt den Prozess mit Schritt **S102** fort, um zu bestimmen, ob das Element ein Nicht-Null-Element ist oder nicht.

[0055] Die nachfolgenden Schritte **S109** bis **S112** entsprechen der Multiplikations-Akkumulationsoperation **S510** aus **Fig. 5**. Diese Schritte werden hauptsächlich von dem Parallelrechner **120** aus **Fig. 4**, einem dedizierten ASIC, der in der Lage ist, parallele Rechenoperationen oder ähnliches durchzuführen, durchgeführt.

[0056] Zunächst wird in Schritt **S109** jede Variable, die für die Multiplikations-Akkumulationsoperation verwendet wird, initialisiert. Im Einzelnen initialisiert der Prozessor (z.B. CPU **130** in **Fig. 4**), der die allgemeine Operation steuert, jeweils eine Indexvariable o des Zeilenzahl-Speicherarrays und des Spaltenzahl-Speicherarrays auf 0. Ferner initialisiert der Prozessor die Ausgangsvektoren f(1) bis f(n) zur Ausgabe von Ergebnissen der Multiplikations-Akkumulationsoperation an die Elemente b(1) bis b(n) des Biasvektors b.

[0057] Im nächsten Schritt **S110** wird die Gesamtzahl k der Einheits-Multiplikations-Akkumulationsoperationen sequentiell mit n Multiplikations-Akkumulationsrechnern durchgeführt. Insbesondere werden der Zeilenzahl-Speicherarray $A_{ROW}(p)$ und der Spaltenzahl-Speicherarray $A_{COL}(p)$ als Zeiger auf die Koeffizientenmatrix A verwendet, und der p-te Rechner **CL** führt die durch die folgende Formel (2) angegebene Einheits-Multiplikations-Akkumulationsoperation durch:

$$f\left(A_{ROW}(p)\right) = f\left(A_{ROW}(p)\right) + A\left(A_{ROW}(p)\right), A_{COL}(p) \cdot x\left(A_{COL}(p)\right) \qquad (2)$$

[0058] Da die Einheits-Multiplikations-Akkumulationsoperationen mit den n Rechnern parallel durchgeführt werden, hat die Variable p in der obigen Formel (2) n Werte in einem Bereich von p=o bis p=o+n-1. Da die Multiplikations-Akkumulationsoperationen nicht so durchgeführt werden, dass sie die Anzahl k der Nicht-Null-Elemente überschreiten, wird eine Multiplikations-Akkumulationsoperation nur dann durchgeführt, wenn p<k erfüllt ist.

[0059] Im nächsten Schritt **S111** wird die Variable für die Multiplikations-Akkumulationsoperationen hochgezählt. Insbesondere erhöht der Prozessor für die Steuerung die Indexvariable o der Zeilenzahl-Speicheranordnung AROW(p) und der Spaltenwert-Speicheranordnung ACOL(p) um die Zahl n der Rechner, um die nächsten n Multiplikations-Akkumulationsoperationen vorzubereiten.

[0060] Im nächsten Schritt **S112** wird bestimmt, ob alle Einheits-Multiplikations-Akkumulationsoperationen abgeschlossen sind. Durch Vergleich der Indexvariablen o jedes Arrays zur Speicherung der Zeilenzahl und des Arrays zur Speicherung der Spaltenzahl mit der Anzahl k von Nicht-Null-Elementen bestimmt der Prozessor zur Steuerung, ob die Multiplikations-Akkumulationsoperationen für alle Nicht-Null-Elemente abgeschlossen sind. Wenn die Indexvariable o jeder der Zeilenzahl-Speicheranordnung und der Spaltenzahl-Speicheranordnung größer oder gleich der Anzahl k von Nicht-Null-Elementen ist, bestimmt der Prozessor zur Steuerung, dass alle Multiplikations-Akkumulationsoperationen abgeschlossen sind, und beendet den Prozess der Multiplikations-Akkumulationsoperationen. Wenn andererseits die Indexvariable o jeder der Zeilenzahl-Speicherarrays und der Spaltenzahl-Speicherarrays nicht größer oder gleich der Anzahl k der Nicht-Null-Elemente ist, fährt der Prozessor für die Steuerung mit dem Prozess zu Schritt **S110** zur Durchführung der Multiplikations-Akkumulationsoperationen fort, um die restlichen Multiplikations-Akkumulationsoperationen durchzuführen.

[0061] Es ist zu beachten, dass bei der oben beschriebenen Prozedur der Rechenoperation die Prüfung auf die Nicht-Null-Elemente der Koeffizientenmatrix A in der Reihenfolge der Zeile und der Spalte durchgeführt wird, aber auch in der Reihenfolge der Spalte und der Zeile durchgeführt werden kann. Außerdem ist die Gesamtzahl der Rechner **CL** gleich der Anzahl n der Zeilen, kann aber auch gleich der Anzahl m der Spalten sein oder auf eine beliebige Anzahl gesetzt werden.

[0062] **Fig. 7** ist ein Flussdiagramm, das ein beispielhaftes Verfahren zur Durchführung der Einheits-Multiplikations-Akkumulationsoperation in jedem Rechner des Parallelrechners von **Fig. 4** zeigt. Es ist zu beachten, dass bei dem unten beschriebenen Verfahren ein erstes Register und ein zweites Register in der Registergruppe R entsprechend jedem Rechner **CL** enthalten sind.

[0063] In einem Schritt **S301** liest Rechner **CL** ein entsprechendes Element $A(A_{ROW}(p), A_{COL}(p))$ der Koeffizientenmatrix aus dediziertem Speicher **126**, **L1** Cache-Speicher **122** oder **L2** Cache-Speicher **125** aus und speichert es in sein entsprechendes erstes Register.

[0064] In einem nächsten Schritt **S302** liest der Rechner **CL** ein entsprechendes Element $x(A_{COL}(p))$ des Eingangsvektors aus dedizierter Speicher **126**, **L1** Cache-Speicher **122** oder **L2** Cache-Speicher **125** aus und speichert es in sein entsprechendes zweites Register. Es ist zu beachten, dass Schritt **S302** auch gleichzeitig mit Schritt **S301** oder vor Schritt **S301** durchgeführt werden kann.

[0065] In einem nächsten Schritt **S303** multipliziert der Rechner **CL** den im ersten Register gespeicherten Wert mit dem im zweiten Register gespeicherten Wert und speichert z.B. ein Multiplikationsergebnis im ersten Register.

[0066] Die nachfolgenden Schritte werden ausgeführt, wenn auf das entsprechende Element $f(A_{ROW}(p))$ des Ausgangsvektors zugegriffen werden kann, d.h. wenn kein Konflikt vorliegt (JA in Schritt **S304**).

[0067] Zunächst liest der Rechner **CL** in einem Schritt **S305** $fs(A_{ROW}(p))$ aus dem dedizierten Speicher **126**, **L1** Cache-Speicher **122** oder **L2** Cache-Speicher **125** und speichert es in das zweite Register.

[0068] In einem nächsten Schritt **S306** werden der im ersten Register gespeicherte Wert (d.h. das Ergebnis der Rechenoperation aus Schritt **S303**) und der im zweiten Register gespeicherte Wert addiert und das Additionsergebnis z.B. im ersten Register gespeichert.

[0069] In einem nächsten Schritt **S307** speichert der Rechner **CL** den im ersten Register gespeicherten Wert (d.h. das Ergebnis der Rechenoperation aus Schritt **S306**) in eine entsprechende Adresse des L1-Cache-Speichers **122**. Auf diese Weise wird die Einheits-Multiplikations-Akkumulationsoperation abgeschlossen.

Spezifisches Beispiel einer Faltungsoperation

[0070] Im Folgenden wird anhand exemplarischer Zahlenwerte die Vorgehensweise der Faltungsoperation von **Fig. 6** näher beschrieben. Konkret werden Koeffizientenmatrix **A**, Eingangsvektor x und Biasvektor b wie in der folgenden Formel (3) eingestellt:

$$f = A \cdot x + b = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix} \quad (3)$$

[0071] Wenn jede Zeile der oben beschriebenen Matrixoperation einzeln von einem entsprechenden Rechner-CL ausgeführt wird, werden vier Rechenoperationen, die in der folgenden Formel (4) dargestellt sind, jeweils vier Rechnern zugeordnet.

$$\left.\begin{aligned} f(1) &= 1\cdot 1+1\cdot 2+1\cdot 3+4 \\ f(2) &= 1\cdot 2+1\cdot 3+3 \\ f(3) &= 1\cdot 3+2 \\ f(4) &= 1\cdot 4+1 \end{aligned}\right\} \quad (4)$$

[0072] Daher ist in diesem Fall die Anzahl der Rechenoperationen des ersten Rechners am größten, während die Anzahl der Rechenoperationen des dritten und vierten Rechners jeweils am kleinsten ist. Die gesamte Prozesszeit wird durch eine Rechenoperationszeit des ersten Rechners bestimmt.

[0073] Wenn andererseits die beispielhaften Zahlenwerte der Formel (3) auf die Schritte **S101** bis **S108** von **Fig. 6** der vorliegenden Ausführungsform angewendet werden, sind die Zeilenzahl-Speicheranordnung $A_{ROW}(p)$ und die Spaltenzahl-Speicheranordnung $A_{COL}(p)$ wie in der folgenden Tabelle 1 beschrieben:

Tabelle 1: Werte von $A_{ROW}(p)$ und $A_{COL}(p)$

p	0	1	2	3	4	5	6
A_{ROW}	1	1	2	1	2	3	4
A_{COL}	1	2	2	3	3	3	4

[0074] Der Index p der obigen Tabelle 1 wird in der nachfolgenden Koeffizientenmatrix A zur besseren Übersichtlichkeit hochgestellt, wie in der folgenden Formel (5) angegeben. Wie in der Formel (5) angegeben, beträgt die Gesamtzahl der Nicht-Null-Elemente, d.h. die Gesamtzahl der Einheits-Multiplikations-Akkumulationsoperationen, 7.

$$f = A\cdot x + b$$
$$= \begin{pmatrix} {}^{0}1 & {}^{1}1 & {}^{3}1 & 0 \\ 0 & {}^{2}1 & {}^{4}1 & 0 \\ 0 & 0 & {}^{5}1 & 0 \\ 0 & 0 & 0 & {}^{6}1 \end{pmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix} \quad (5)$$

[0075] Basierend auf dem Zeilenzahl-Speicherarray $A_{ROW}(p)$ und dem Spaltenzahl-Speicherarray $A_{COL}(p)$ werden die in der Koeffizientenmatrix A durchzuführenden Einheits-Multiplikations-Akkumulationsoperationen wie in den folgenden Formeln (6.1) bis (6.7) ausgedrückt:

$$f(1) = f(1) + A(1,1)\cdot x(1) \quad (6.1)$$

$$f(1) = f(1) + A(1,2)\cdot x(2) \quad (6.2)$$

$$f(2) = f(2) + A(2,2)\cdot x(2) \quad (6.3)$$

$$f(1) = f(1) + A(1,3)\cdot x(3) \quad (6.4)$$

$$f(2) = f(2) + A(2,3)\cdot x(3) \quad (6.5)$$

$$f(3) = f(3) + A(3,3)\cdot x(3) \quad (6.6)$$

$$f(4) = f(4) + A(4,)\cdot x(4) \quad (6.7)$$

[0076] Da die Anzahl n der Rechners **CL 4** ist, werden die durch die Formeln (6.1) bis (6.4) angegebenen Einheits-Multiplikations-Akkumulationsoperationen jeweils im ersten Schritt durch den nullten bis dritten Rechner **CL0** bis **CL3** durchgeführt. In diesem Fall wird für den Zugriff auf das entsprechende Element f(1) des Ausgangsvektors jeder Einheits-Multiplikations-Akkumulationsoperation in der Formel (6.1), der Formel (6.2) und der Formel (6.4) die ausschließliche Kontrolle durchgeführt.

[0077] In einem nächsten Schritt werden die Einheits-Multiplikations-Akkumulationsoperationen der Formeln (6.5) bis (6.7) jeweils durch den nullten bis zweiten Rechner **CL0** bis **CL2** durchgeführt.

Wirkung

[0078] Wie oben beschrieben, werden gemäß der ersten Ausführungsform die Nicht-Null-Elemente der Koeffizientenmatrix A gesucht und die Einheits-Multiplikations-Akkumulationsoperationen für die Nicht-Null-Elemente auf Basis des Suchergebnisses dem Rechner zugeordnet und von diesem durchgeführt. Entsprechend können die Anzahl der Prozesse für die Einheits-Multiplikations-Akkumulationsoperationen unter den Rechnern ausgeglichen werden, wodurch die Multiplikations-Akkumulationsoperationen durch die Vielzahl der Rechner effizient durchgeführt werden können.

[0079] **Fig. 8** veranschaulicht einen Effekt der ersten Ausführungsform. **Fig. 8 (A)** zeigt ein Vergleichsbeispiel und **Fig. 8 (B)** zeigt den Fall der vorliegenden Ausführungsform.

[0080] **Fig. 8 (A)** zeigt einen Fall, bei dem Multiplikations-Akkumulationsoperationen der jeweiligen Zeilen der Koeffizientenmatrix A und des Eingangsvektors x individuell dem Rechner **CL** zugeordnet werden. In diesem Fall, auch wenn nur die Nicht-Null-Elemente berechnet werden, wird, wenn die Anzahl der Nicht-Null-Elemente zwischen den Zeilen der Koeffizientenmatrix unterschiedlich ist, die gesamte Rechenoperationsdauer durch eine Rechenoperation im Rechner **CL** bestimmt, die der Zeile mit der größten Anzahl von Nicht-Null-Elementen entspricht.

[0081] Im Fall der vorliegenden Ausführungsform aus **Fig. 8 (B)** werden die Einheits-Multiplikations-Akkumulationsoperationen dem Rechner im Wesentlichen gleich zugeordnet. Das heißt, Bereiche der Einheits-Multiplikations-Akkumulationsoperationen, die vom Rechner (**1**) und vom Rechner (**2**) im Fall von **Fig. 8 (A)** durchgeführt werden, werden anderen Rechnern zugewiesen. Entsprechend kann die gesamte Prozesszeit reduziert werden.

Ausführungsform 2

[0082] Im Schritt **S110** des in **Fig. 6** gezeigten Flussdiagramms greifen n Rechner **CL** gleichzeitig auf die Koeffizientenmatrix A mit den n Zeilen und den m Spalten über die Zeilenzahl-Speicheranordnung $A_{ROW}(p)$ und die Spaltenzahl-Speicheranordnung $A_{COL}(p)$ zu. Entsprechend hoch wird die Prozesslast bei einer großen Koeffizientenmatrix A. Um dem entgegenzuwirken, wird in der zweiten Ausführungsform anstelle des Zugriffs auf die Koeffizientenmatrix A mit den n Zeilen und den m Spalten über die Zeilenzahl-Speicheranordnung $A_{ROW}(p)$ und die Spaltenzahl-Speicheranordnung $A_{COL}(p)$ ein Koeffizientenfeld A' (auch als „Koeffizientenvektor A' “ bezeichnet) mit Ausnahme der Nicht-Null-Elemente neu definiert und n Rechner **CL** greifen auf das Koeffizientenfeld A' zu. Damit kann ein intensiver Zugriff auf die Koeffizientenmatrix A mit den n Zeilen und den m Spalten verhindert werden.

[0083] **Fig. 9** ist ein Flussdiagramm, das den Ablauf einer Faltungsoperation nach der zweiten Ausführungsform zeigt. Die Schritte **S401** bis **S412** der **Fig. 9** entsprechen jeweils den Schritten **S101** bis **S112** der **Fig. 6**. Allerdings unterscheiden sich Bereiche der Abläufe in Schritt **S403** und Schritt **S412** davon. Im Folgenden werden hauptsächlich die Schritte **S403** und **S412** mit unterschiedlichen Prozessen beschrieben, wobei die Schritte mit den gleichen Prozessen wie in **Fig. 6** nicht wiederholt beschrieben werden.

[0084] Unter Bezugnahme auf **Fig. 9** werden in Schritt **S403** die Werte der in der Koeffizientenmatrix A enthaltenen Nicht-Null-Elemente sowie deren Zeilen- und Spaltennummern gespeichert. Um insbesondere ein Element A(i, j) zu speichern, das für eine Multiplikations-Akkumulationsoperation vorgesehen ist, speichert der Prozessor eine Zeilenrichtungsvariable i in der Zeilenzahl-Speicheranordnung $A_{ROW}(k)$, eine Spaltenrichtungsvariable j in der Spaltenzahl-Speicheranordnung $A_{COL}(k)$ und das Element A(i, j) der Koeffizientenmatrix A in dem Koeffizientenfeld A'(k).

[0085] Im Folgenden wird insbesondere der Fall der beispielhaften Zahlenwerte in der obigen Formel (3) beschrieben. In diesem Fall wird das Koeffizientenfeld A' wie in der folgenden Tabelle 2 beschrieben angegeben.

Tabelle 2: Wert von A'(p)

p	0	1	2	3	4	5	6
A'	1	1	1	1	1	1	1

[0086] In Schritt **S410** werden k Einheits-Multiplikations-Akkumulationsoperationen insgesamt sequentiell von n Rechnern **CL** ausgeführt. Insbesondere werden der Zeilenzahl-Speicherbereich $A_{ROW}(p)$ und der Spaltenzahl-Speicherbereich $A_{COL}(p)$ als Zeiger auf den Ausgangsvektor f bzw. den Eingangsvektor **x** verwendet, und der p-te Rechner **CL** führt eine Einheits-Multiplikations-Akkumulationsoperation durch, die durch die folgende Formel (7) angegeben wird:

$$f\left(A_{ROW}(p)\right) = f\left(A_{ROW}(p)\right) + A'(p) \cdot x\left(A_{COL}(p)\right) \qquad (7)$$

[0087] Da die Einheits-Multiplikations-Akkumulationsoperationen mit den n-Rechnern parallel durchgeführt werden, nimmt die Variable p der Formel (7) n Werte in einem Bereich von p=o bis p=o+n-1 an. Da die Multiplikations-Akkumulationsoperationen durchgeführt werden, ohne die Anzahl k der Nicht-Null-Elemente zu überschreiten, wird eine Multiplikations-Akkumulationsoperation nur dann durchgeführt, wenn p<k erfüllt ist.

[0088] Wie oben beschrieben, kann nach der zweiten Ausführungsform der gleiche Effekt wie in der ersten Ausführungsform gezeigt werden, und ein intensiver Zugriff auf die Koeffizientenmatrix A mit den n Zeilen und den m Spalten kann verhindert werden.

[0089] Die hier beschriebenen Ausführungsformen sind in jeder Hinsicht illustrativ und nicht einschränkend. Der Umfang der vorliegenden Erfindung wird nicht durch die oben beschriebenen Ausführungsformen definiert und soll alle Änderungen umfassen.

Bezugszeichenliste

100	Eingabedaten (Eingabe Feature Map)
101	Kernel
102	Ausgangsdaten
110, x	Eingangsvektor
111, A	Koeffizientenmatrix
120	Parallelcomputer
121	Verarbeitungseinheit
122	L1 Cache-Speicher
125	L2-Cache-Speicher
126	dedizierter Speicher
130	CPU
131	Speicher
A'	Koeffizientenfeld (Koeffizientenvektor)
CL	Rechner
R	Registergruppe
b	Biasvektor
f	Ausgangsvektor

ZITATE ENTHALTEN IN DER BESCHREIBUNG

Diese Liste der vom Anmelder aufgeführten Dokumente wurde automatisiert erzeugt und ist ausschließlich zur besseren Information des Lesers aufgenommen. Die Liste ist nicht Bestandteil der deutschen Patent- bzw. Gebrauchsmusteranmeldung. Das DPMA übernimmt keinerlei Haftung für etwaige Fehler oder Auslassungen.

Zitierte Patentliteratur

- JP 2010134697 A [0003, 0004]

Patentansprüche

1. Rechenoperationsschaltung, um eine Koeffizientenmatrix, die Nicht-Null-Elemente und Null-Elemente enthält, mit einem Eingangsvektor von rechts zu multiplizieren und um das Rechenoperationsergebnis als Ausgangsvektor auszugeben, wobei die Rechenoperationsschaltung Folgendes aufweist:
Einen Steuerprozessor; und
eine Vielzahl von Rechnern, die jeweils in der Lage sind, Prozesse parallel zueinander auszuführen, wobei
der Steuerprozessor den Rechnern Multiplikationen der in der Koeffizientenmatrix enthaltenen Nicht-Null-Elemente und entsprechende Elemente des Eingangsvektors zuweist, wobei jede der Multiplikationen als eine Prozesseinheit behandelt wird, um die Anzahl der Prozesseinheiten unter den Rechnern zu nivellieren, und wobei
die Rechner nacheinander die zugewiesenen Multiplikationen ausführen und die Ergebnisse der Multiplikationen nacheinander zu den entsprechenden Elementen des Ausgangsvektors addieren.

2. Rechenoperationsschaltung nach Anspruch 1, wobei
die Rechenoperationsschaltung eine Rechenoperation einer Faltungsschicht in einem Gefalteten Neuronalen Netzwerk durchführt,
der Eingangsvektor durch die Erweiterung einer Merkmalskarte, die in die Faltungsschicht eingegeben wird, in eine Spalte erhalten wird, und wobei
die Koeffizientenmatrix einem Kernel entspricht, der in der Faltungsschicht verwendet wird.

3. Rechenoperationsschaltung nach Anspruch 1 oder 2, die weiterhin Folgendes aufweist:
Einen Koeffizientenvektor, um nur die aus der Koeffizientenmatrix extrahierten Nicht-Null-Elemente zu speichern, wobei
die Rechner die Multiplikationen durchführten, indem sie entsprechende Nicht-Null-Elemente verwendet, die aus dem Koeffizientenvektor extrahiert werden.

4. Rechenoperationsverfahren für eine Faltungsschicht in einem Gefalteten Neuronalen Netzwerk, wobei das Rechenoperationsverfahren folgende Schritte aufweist:
Erzeugen einer Koeffizientenmatrix durch Umwandeln eines in der Faltungsschicht verwendeten Kernels, so dass die Koeffizientenmatrix mit einem Eingangsvektor verbunden ist, der durch Expandieren einer Merkmalskarte, die in die Faltungsschicht eingegeben wird, in eine Spalte erhalten wird;
Suche nach Nicht-Null-Elementen, die in der Koeffizientenmatrix enthalten sind;
Zuweisen von Multiplikationen der in der Koeffizientenmatrix enthaltenen Nicht-Null-Elemente und entsprechender Elemente des Eingangsvektors an eine Vielzahl von Rechnern, wobei jede der Multiplikationen als eine Prozesseinheit behandelt wird, um die Anzahl der Prozesseinheiten zwischen den Rechnern zu nivellieren, wobei alle Rechner in der Lage sind, Prozesse parallel zueinander auszuführen; und
sequentielles Ausführen der zugewiesenen Multiplikationen durch die Rechner und sequentielles Addieren der Ergebnisse der Multiplikationen durch die Rechner zu den entsprechenden Elementen eines Ausgangsvektors.

5. Rechenoperationsverfahren nach Anspruch 4, wobei ferner ein Biasvektor als Anfangswert des Ausgangsvektors eingegeben wird.

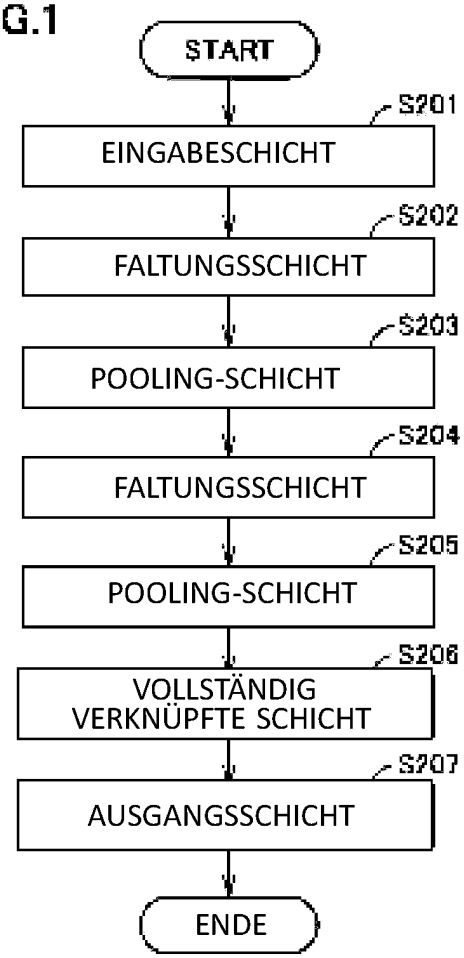
6. Rechenoperationsverfahren nach Anspruch 4 oder 5, wobei ferner folgende Schritte durchgeführt werden:
Extrahieren nur der Nicht-Null-Elemente, die in der Koeffizientenmatrix enthalten sind; und
Speichern der Nicht-Null-Elemente als Koeffizientenvektor, wobei
die Rechner die Multiplikationen durchführen, indem sie entsprechende, vom Koeffizientenvektor extrahierte Nicht-Null-Elemente verwenden.

7. Programm, das einen Computer veranlasst, das in einem der Ansprüche 4 bis 6 aufgeführte Rechenoperationsverfahren durchzuführen.

Es folgen 9 Seiten Zeichnungen

Anhängende Zeichnungen

FIG.1



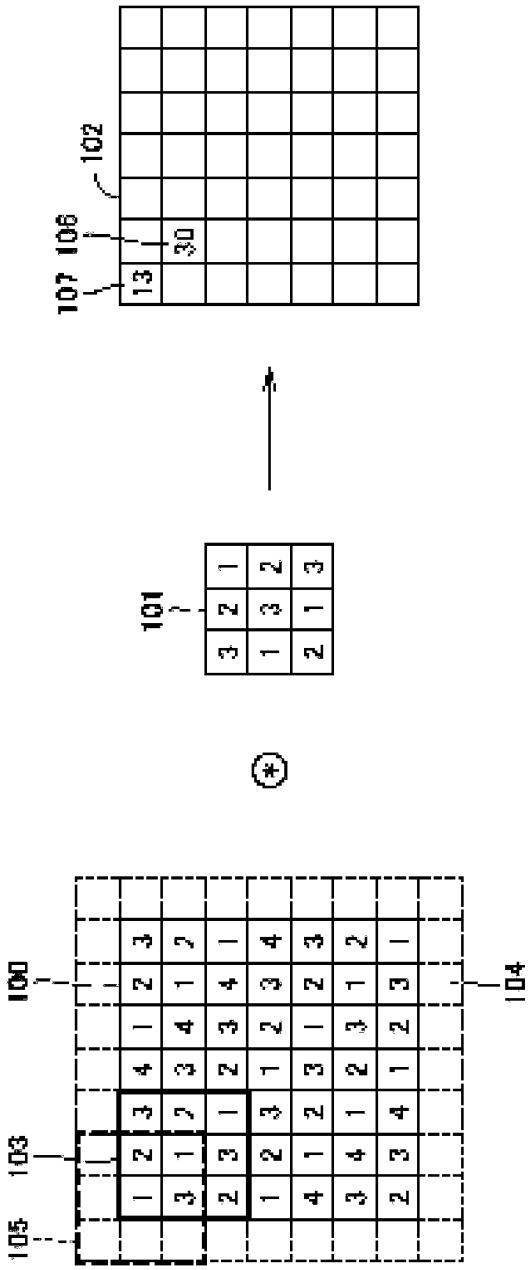


FIG.2

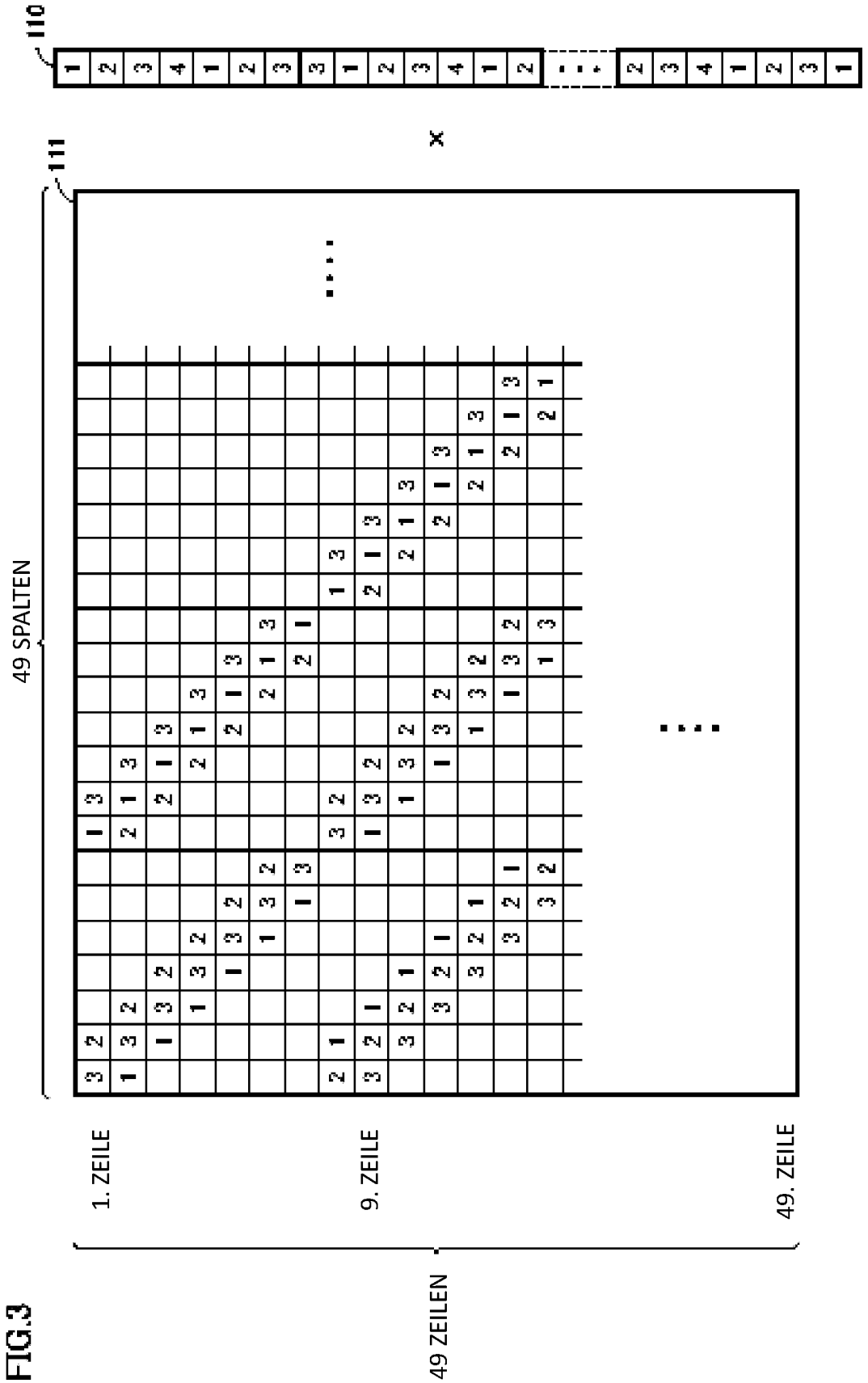


FIG.3

FIG.4

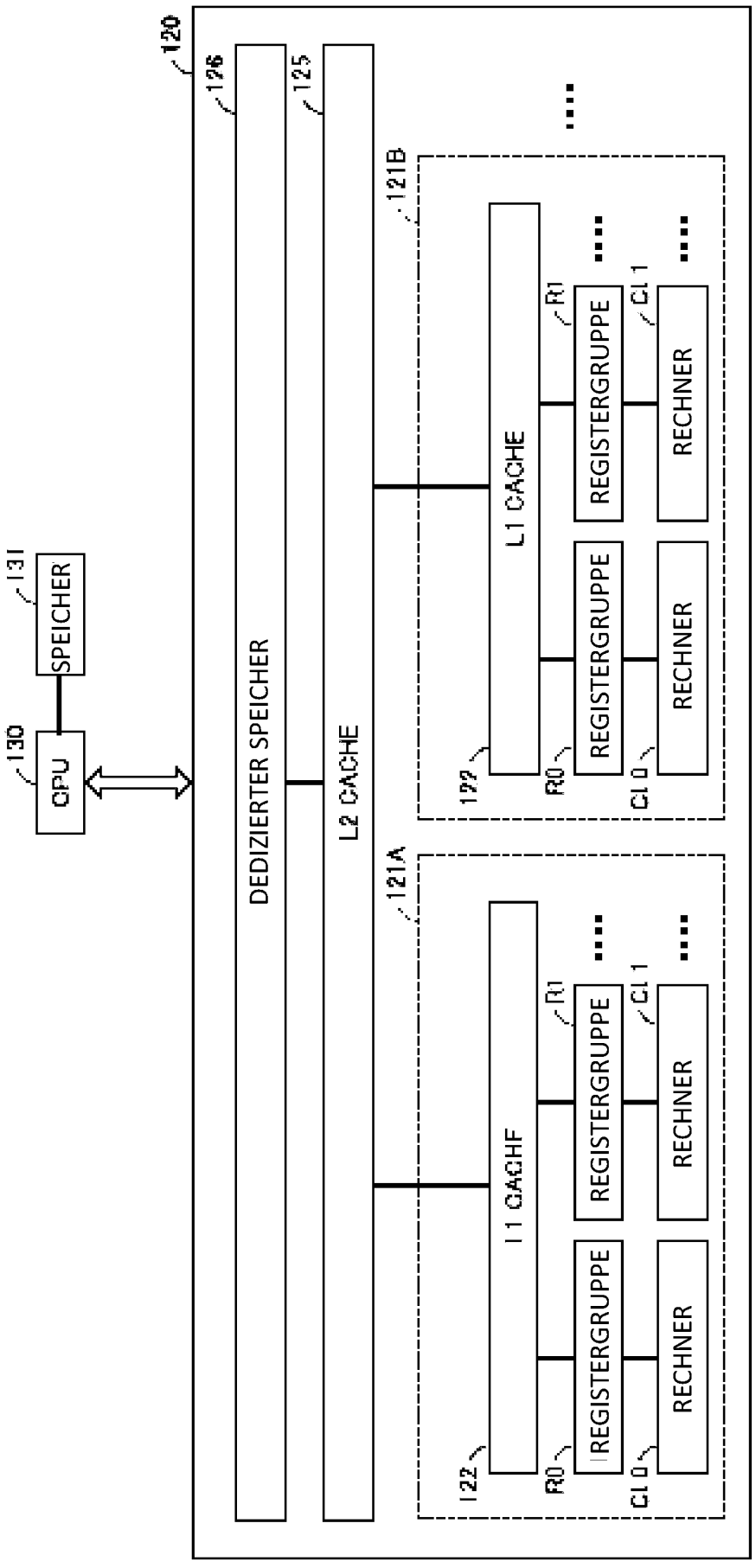

120
126
125
121B
121A
131
SPEICHER
130
CPU
DEDIZIERTER SPEICHER
L2 CACHE
L1 CACHE
122
R0
R1
C1 0
C1 1
REGISTERGRUPPE
RECHNER

FIG.5

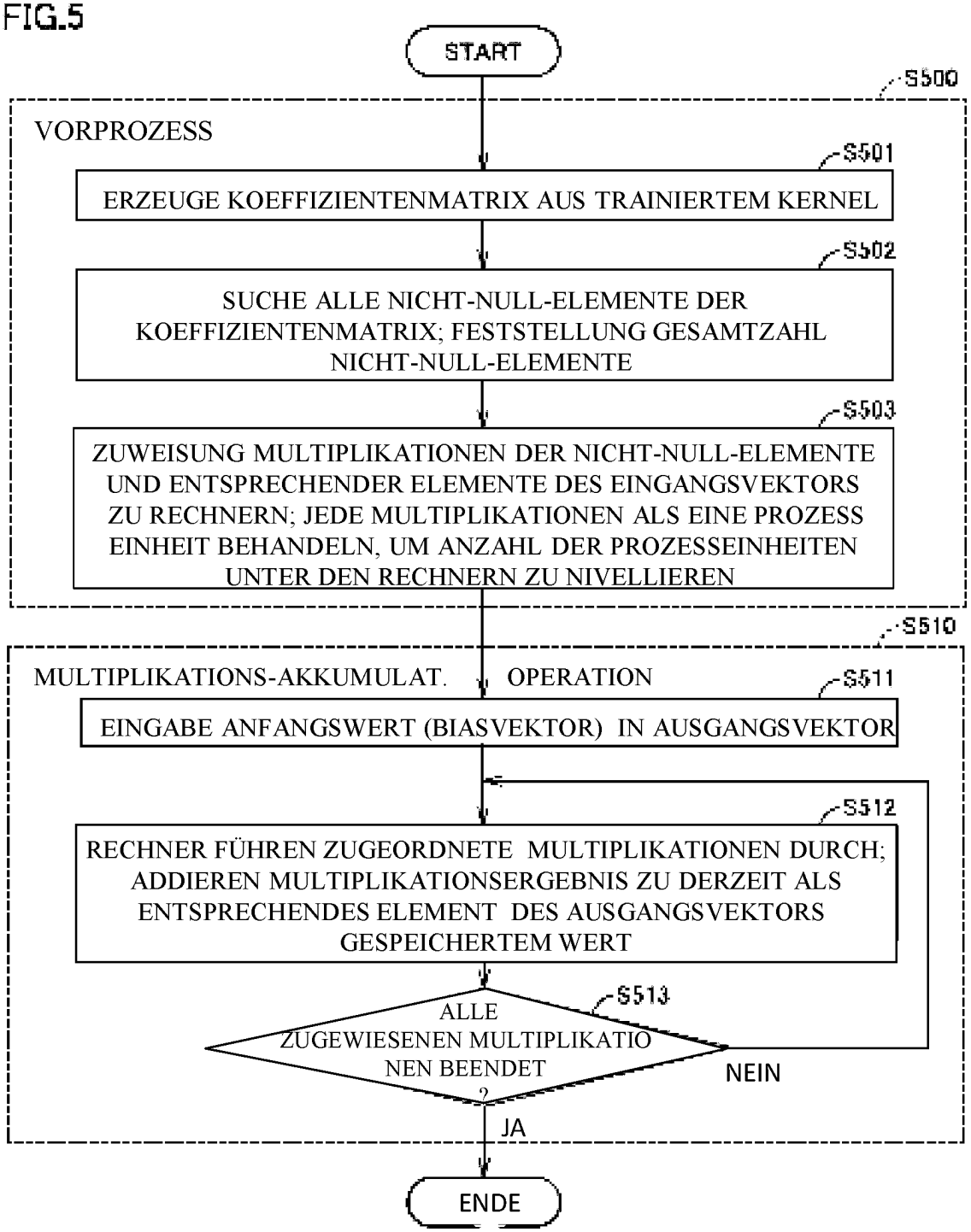

START
S500
VORPROZESS
S501
ERZEUGE KOEFFIZIENTENMATRIX AUS TRAINIERTEM KERNEL
S502
SUCHE ALLE NICHT-NULL-ELEMENTE DER KOEFFIZIENTENMATRIX; FESTSTELLUNG GESAMTZAHL NICHT-NULL-ELEMENTE
S503
ZUWEISUNG MULTIPLIKATIONEN DER NICHT-NULL-ELEMENTE UND ENTSPRECHENDER ELEMENTE DES EINGANGSVEKTORS ZU RECHNERN; JEDE MULTIPLIKATIONEN ALS EINE PROZESS EINHEIT BEHANDELN, UM ANZAHL DER PROZESSEINHEITEN UNTER DEN RECHNERN ZU NIVELLIEREN
S510
MULTIPLIKATIONS-AKKUMULAT. OPERATION
S511
EINGABE ANFANGSWERT (BIASVEKTOR) IN AUSGANGSVEKTOR
S512
RECHNER FÜHREN ZUGEORDNETE MULTIPLIKATIONEN DURCH; ADDIEREN MULTIPLIKATIONSERGEBNIS ZU DERZEIT ALS ENTSPRECHENDES ELEMENT DES AUSGANGSVEKTORS GESPEICHERTEM WERT
S513
ALLE ZUGEWIESENEN MULTIPLIKATIONEN BEENDET ?
NEIN
JA
ENDE

FIG.6

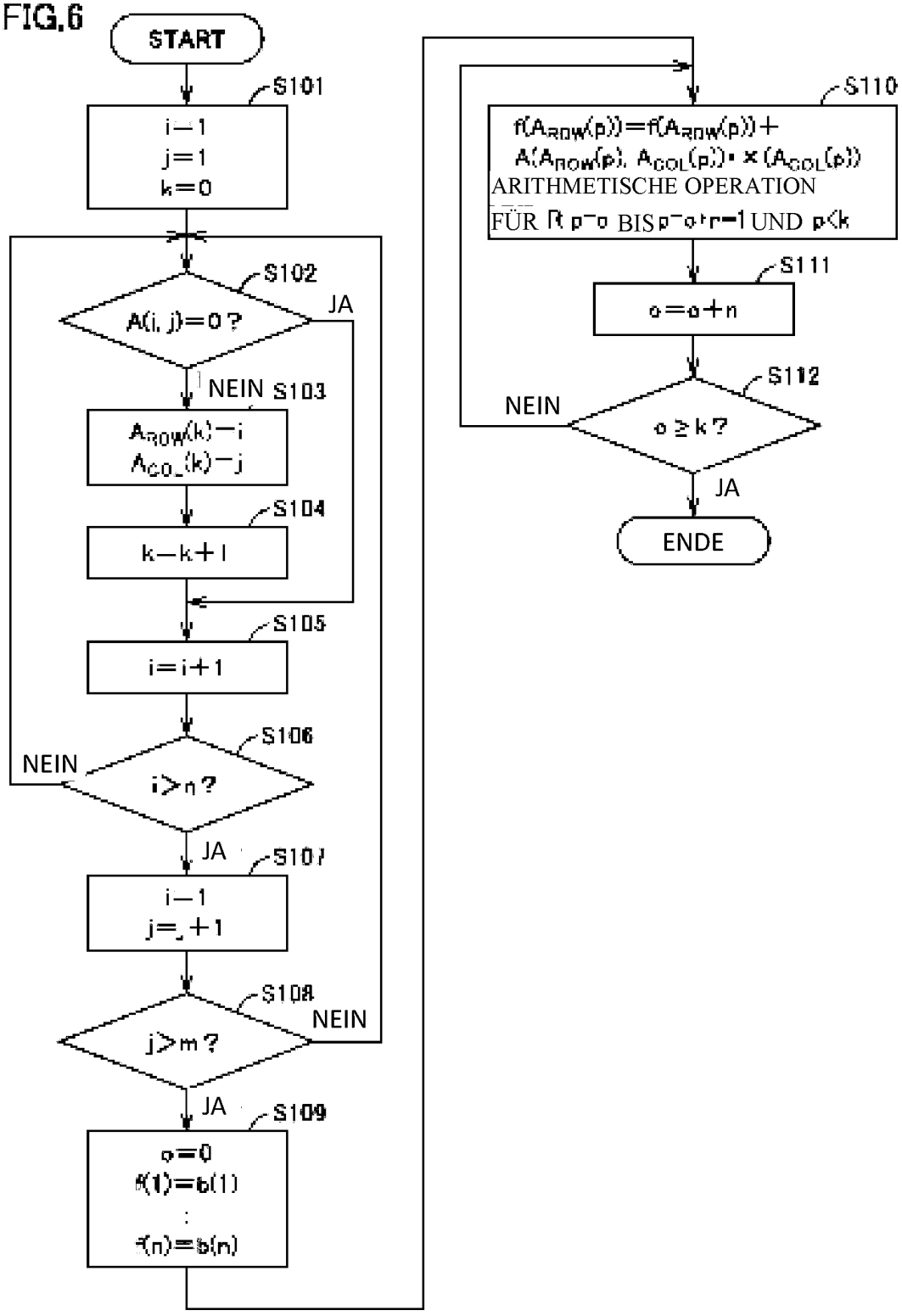

START
S101
i=1
j=1
k=0
S102
A(i, j)=0 ?
JA
NEIN
S103
$A_{ROW}(k)=i$
$A_{COL}(k)=j$
S104
k=k+1
S105
i=i+1
S106
NEIN
i>n ?
JA
S107
i=1
j=j+1
S108
NEIN
j>m ?
JA
S109
o=0
b(1)=b(1)
:
b(n)=b(n)
S110
$f(A_{ROW}(p))=f(A_{ROW}(p))+$
$A(A_{ROW}(p), A_{COL}(p)) \times (A_{COL}(p))$
ARITHMETISCHE OPERATION
FÜR p=o BIS p=o+r=1 UND p<k
S111
o=o+n
S112
NEIN
o ≥ k ?
JA
ENDE

FIG.7

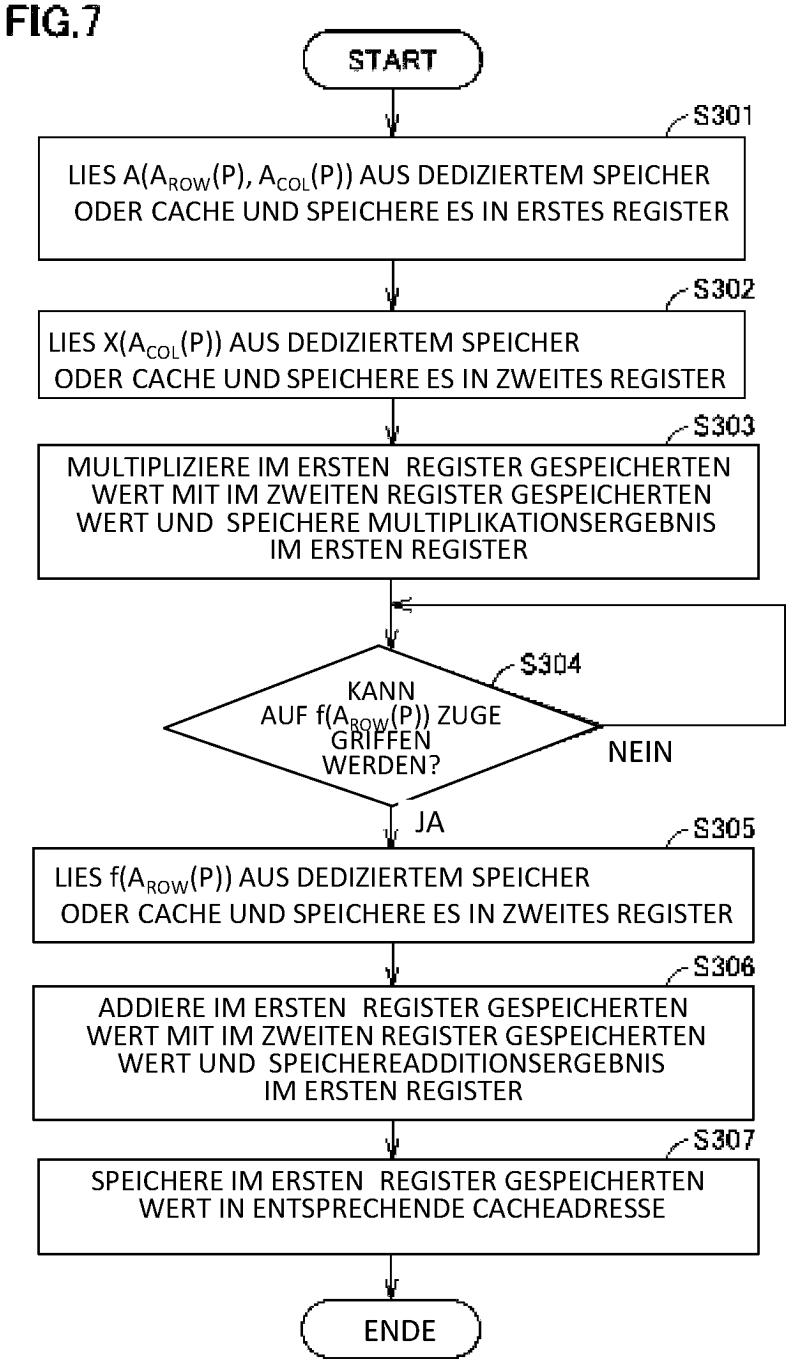
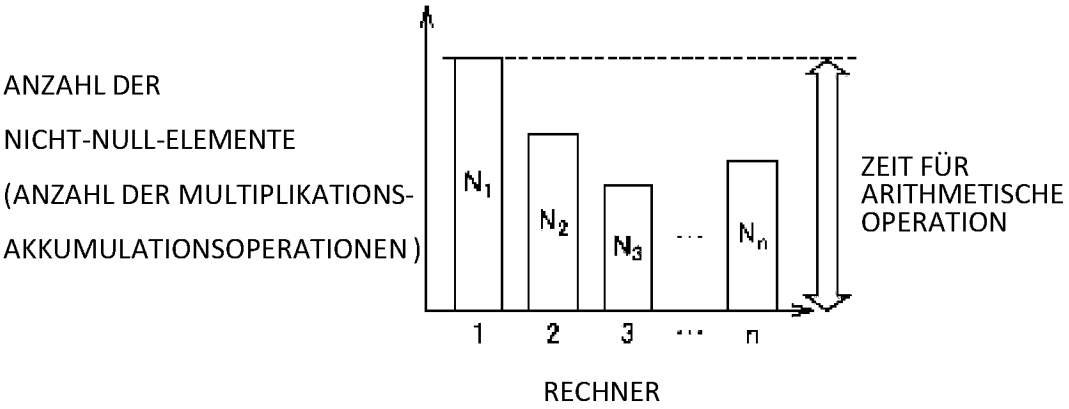

START
S301
LIES A(A_{ROW}(P), A_{COL}(P)) AUS DEDIZIERTEM SPEICHER ODER CACHE UND SPEICHERE ES IN ERSTES REGISTER
S302
LIES X(A_{COL}(P)) AUS DEDIZIERTEM SPEICHER ODER CACHE UND SPEICHERE ES IN ZWEITES REGISTER
S303
MULTIPLIZIERE IM ERSTEN REGISTER GESPEICHERTEN WERT MIT IM ZWEITEN REGISTER GESPEICHERTEN WERT UND SPEICHERE MULTIPLIKATIONSERGEBNIS IM ERSTEN REGISTER
S304
KANN AUF f(A_{ROW}(P)) ZUGE GRIFFEN WERDEN?
NEIN
JA
S305
LIES f(A_{ROW}(P)) AUS DEDIZIERTEM SPEICHER ODER CACHE UND SPEICHERE ES IN ZWEITES REGISTER
S306
ADDIERE IM ERSTEN REGISTER GESPEICHERTEN WERT MIT IM ZWEITEN REGISTER GESPEICHERTEN WERT UND SPEICHEREADDITIONSERGEBNIS IM ERSTEN REGISTER
S307
SPEICHERE IM ERSTEN REGISTER GESPEICHERTEN WERT IN ENTSPRECHENDE CACHEADRESSE
ENDE

FIG.8

(A) VERGLEICHSBEISPIEL



(B) VORLIEGENDE AUSFÜHRUNGSFORM

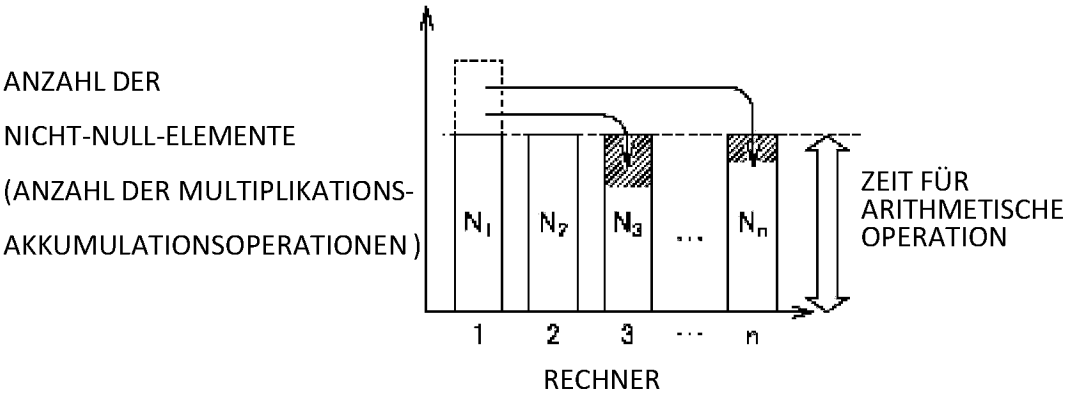


FIG.9